Integrated Modeling for Road Condition Prediction

Installation and Administration Guide

www.its.dot.gov/index.htm
October 2019
FHWA-JPO-18-746

U.S. Department of Transportation

Produced under contract DTFH61-16-D-00053, Operations IV
U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology,
Intelligent Transportation Systems Joint Program Office
and
Federal Highway Administration, Office of Operations

Notice

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA-JPO-18-746		

4. Title and Subtitle	5. Report Date
Integrated Modeling for Road Condition Prediction Installation and Administration Guide	October 2019
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
J. Kyle Garrett, Bryan Krueger, Aaron Cherney (Synesis Partners, LLC); Hani Mahmassani (Northwestern University); Jiaqi Ma (University of Cincinnati); Michelle Neuner (Leidos)	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Leidos 11951 Freedom Drive Reston, VA 20190	11. Contract or Grant No. DTFH61-16-D-00053L, Task 693JJ318F000084

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Federal Highway Administration U.S. Department of Transportation 1200 New Jersey Avenue, SE Washington D.C., 20590	Installation and Administration Guide
	14. Sponsoring Agency Code HOIT

15. Supplementary Notes

The Government Task Managers: Mr. Gabriel Guevara and Mr. Jawad Paracha.

16. Abstract

Transportation systems management and operations is at a critical point in their development due to an explosion in data availability and analytics. Intelligent transportation systems gathering data about weather and traffic conditions coupled with the imminent deployment of connected vehicles will bring an increase in data availability to power traffic and road condition predictions. This convergence of opportunities has led the Federal Highway Administration's Road Weather Management Program to initiate research into integrated modeling for road condition prediction (IMRCP) to investigate and capture that potential.

The product of this IMRCP research is a prototype system and demonstration deployment that provides a framework for the integration of road condition monitoring and forecast data to support decisions by travelers, transportation operators, and maintenance providers. The system collects and integrates environmental and transportation operations data, collects forecast weather data when available, initiates road weather and traffic forecasts, generates advisories and warnings, and provides the results to other applications and systems.

The purpose of this Installation and Administration Guide is to describe the installation, configuration, operations, and maintenance of the IMRCP system.

17. Keywords	18. Distribution Statement
road weather management, road condition prediction	No restrictions

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	68	

Table of Contents

Chapter 1. Introduction 1

Background 1
Purpose 1
Document Overview 2

Chapter 2. Installation 3

Server Environment 3
Integrated Modeling for Road Condition Prediction Server 4
Integrated Modeling for Road Condition Prediction Components 4
Traffic Estimation and Prediction System Server 6
Traffic Estimation and Prediction System Components 6

Chapter 3. Model Localization and Configuration 9

Atmospheric Weather Model 9
Road Network Model 9
Traffic Detectors 10
Traffic Estimation and Prediction System Traffic Demand Model 12
Machine Learning-based Prediction Traffic Model 13
Configuration Files 16
Road Weather Model 17
Hydrological Model 18

Chapter 4. Application Configuration 21

Configuration Files 21
Config.json 21
Context.xml 21
Web.xml 21
Polygons (Localization Required) 21
Segments (Localization Required) 22
log4j2.properties 22
Detector Locations (Localization Required) 22
Routes (Localization Required) 22
Unit Conversions 22
Source Units (Localization Required) 22
Advanced Hydrologic Prediction Service (Localization Required) 22
Advanced Hydrologic Prediction Service Locations (Localization Required) 23
Advanced Hydrologic Prediction Service Stages (Localization Required) 23
Machine Learning-Based Prediction Detector Metadata (Localization Required) 23
Downstream Links (Localization Required) 23
Machine Learning-Based Prediction LinkId Map (Localization Required) 23
Traffic Management Center LinkId Map (Localization Required) 24

Machine Learning-Based Prediction Links Downstream Detectors (Localization Required) 24
Stormwatch Locations (Location-Specific) 24
Stormwatch Devices (Location-Specific) 24
Stormwatch Stages (Location-Specific) 24
Alerts 24
Chapter 5. Operations 25
LOG FILE 25
SYSTEM STARTUP AND SHUTDOWN 25
Chapter 6. Maintenance 27
BACKUP 27
USER ACCOUNTS 27
Chapter 7. References 29
Appendix A. Log File Example 31
Appendix B. config.json Definitions 33
Appendix C. Dynasmart-X Real-Time Dynamic Traffic Assignment System 47
Appendix D. Glossary 57

List of Tables

Table 1. Machine learning-based prediction variables. 14
Table 2. Road weather model observation data sources. 17
Table 3. Road weather model forecast data sources. 18
Table 4. imrcp.system.Directory. 33
Table 5. imrcp.ImrcpBlock 33
Table 6. imrcp.store.Store. 34
Table 7. imrcp.store.WeatherStore. 34
Table 8. RapNcf. 35
Table 9. imrcp.store.NcfWrapper. 35
Table 10. imrcp.store.MRMSNcfWrapper.mrms. 35
Table 11. imrcp.store.MRMSNcfWrapper. 35
Table 12. imrcp.collect.RemoteGrid. 35
Table 13. imrcp.collect.MRMS. 36
Table 14. imrcp.collect.KCScoutDetectors. 36
Table 15. imrcp.collect.KCScoutIncidents. 36
Table 16. IncidentDbWrapper. 36
Table 17. imrcp.collect.RAPPcCat. 36
Table 18. imrcp.collect.MRMSPcCat. 37
Table 19. imrcp.geosrv.SegmentShps. 37
Table 20. imrcp.geosrv.NED. 37
Table 21. imrcp.forecast.treps.RealTimeDetector. 37
Table 22. imrcp.forecast.treps.RealTimeIncident. 37
Table 23. imrcp.forecast.treps.RealTimeWorkzone. 37
Table 24. imrcp.forecast.treps.RealTimeWeather. 37
Table 25. SourceUnits. 38
Table 26. imrcp.forecast.mdss.Metro. 38
Table 27. DoMetroWrapper. 38
Table 28. imrcp.subs.Subscription. 38
Table 29. imrcp.alert.Alerts. 39
Table 30. imrcp.alert.AlertsStore. 39
Table 31. imrcp.system.Units. 39
Table 32. imrcp.forecast.treps.OutputManager. 40
Table 33. TrepsFtp. 40
Table 34. imrcp.forecast.treps.TrepsCollect. 40
Table 35. imrcp.collect.AHPS. 41
Table 36. imrcp.collect.SubSurfaceTemp. 41
Table 37. imrcp.collect.CAP. 41
Table 38. imrcp.store.CAPStore. 41
Table 39. Imrcp.geosrv.Polygons. 42
Table 40. imrcp.collect.StormWatch. 42
Table 41. imrcp.route.Routes. 42
Table 42. route. 42

Table 43. rangerules_<obstypeid>. 42
Table 44. imrcp.alert.Notifications. 43
Table 45. imrcp.store.PresentationCache. 43
Table 46. NotificationServlet. 43
Table 47. imrcp.geosrv.SensorLocations. 43
Table 48. imrcp.forecast.mlp.MLPBlock. 44
Table 49. imrcp.forecast.mlp.MLPPredict. 44
Table 50. imrcp.forecast.mlp.MLPUpdate. 44
Table 51. imrcpBlocks that need configuration for localization. 45
Table 52. Environment variables for TAO. 48
Table 53. ACE_TAO visual studio setting. 50

List of Figures

Figure 1. Integrated Modeling for Road Condition Prediction deployment diagram. 3
Figure 2. Integrated Modeling for Road Condition Prediction components. 6
Figure 3. Traffic Estimation and Prediction System installation process flowchart. 7
Figure 4. Grade calculation. 10
Figure 5. Detector configuration data relationships. 11
Figure 6. Day-to-day learning framework for online origin-destination demand estimation and prediction. 13
Figure 7. Windows environment variables window. 47
Figure 8. ActivePerl version. 49
Figure 9. MFC generation process. 49
Figure 10. MFC solution window. 50
Figure 11. Library files generated. 51
Figure 12. Initial graphical user interface of DYNASMART-X. 53
Figure 13. DYNASMART mode selection window. 53
Figure 14. DYNASMART naming service window. 54
Figure 15. DYNASMART open services windows. 54
Figure 16. Completed DYNASMART initialization. 55

Chapter 1. Introduction

Background

Transportation system management and operations (TSMO) is at the cusp of a revolution, spurred by the explosion in data from different sources and the increasing sophistication of models using these data. 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 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, 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 strategies in support of operational decisions continues to be limited.

Recent efforts to incorporate forecast weather conditions into traffic predictions under the U.S. Department of Transportation (USDOT) Traffic Estimation and Prediction System (TrEPS) project have shown considerable promise. The utility of traffic predictions can, however, be further enhanced by augmenting the forecast weather conditions with known and likely capacity constraints, such as work zones and incidents. Factoring in reported conditions from environmental sensor stations, vehicle fleets, and citizen-reported conditions will further enhance predictions. 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 has initiated an investigation into and development of Integrated Modeling for Road Condition Prediction (IMRCP). This effort includes an initial survey of available and imminent weather, hydrological, traffic, and related transportation management models; development of a concept of operations (ConOps); and development of fundamental system requirements. Follow-on efforts will develop a system architecture and system design, implement a foundational system, and deploy the system with an operating transportation agency to evaluate its effectiveness.

Purpose

As described in the IMRCP ConOps, the purpose of IMRCP is to integrate weather and traffic data sources and predictive methods to effectively predict road and travel conditions. The first step in the study surveyed the existing field of predictive models in road weather, traffic, and related disciplines. The ConOps then developed the case for and a description of an integrated model for predicting road conditions that incorporates transportation and non-transportation data, deterministic and probabilistic data, and measured and reported data. 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 Installation and Administration Guide is to describe the installation, configuration, operations, and maintenance of the IMRCP system. The system, as packaged, includes a default configuration for demonstration deployment in the Kansas City metropolitan area. While the system will install and run in this configuration, localization of the data sources, data collectors, road network model(s), and traffic model(s) will be needed for application in other geographic contexts. The localization will likely require development of new software components specific to the applicable data sources, collectors, and traffic models. Configuration settings and modules for which this may be the case are noted as such in this document.

Document Overview

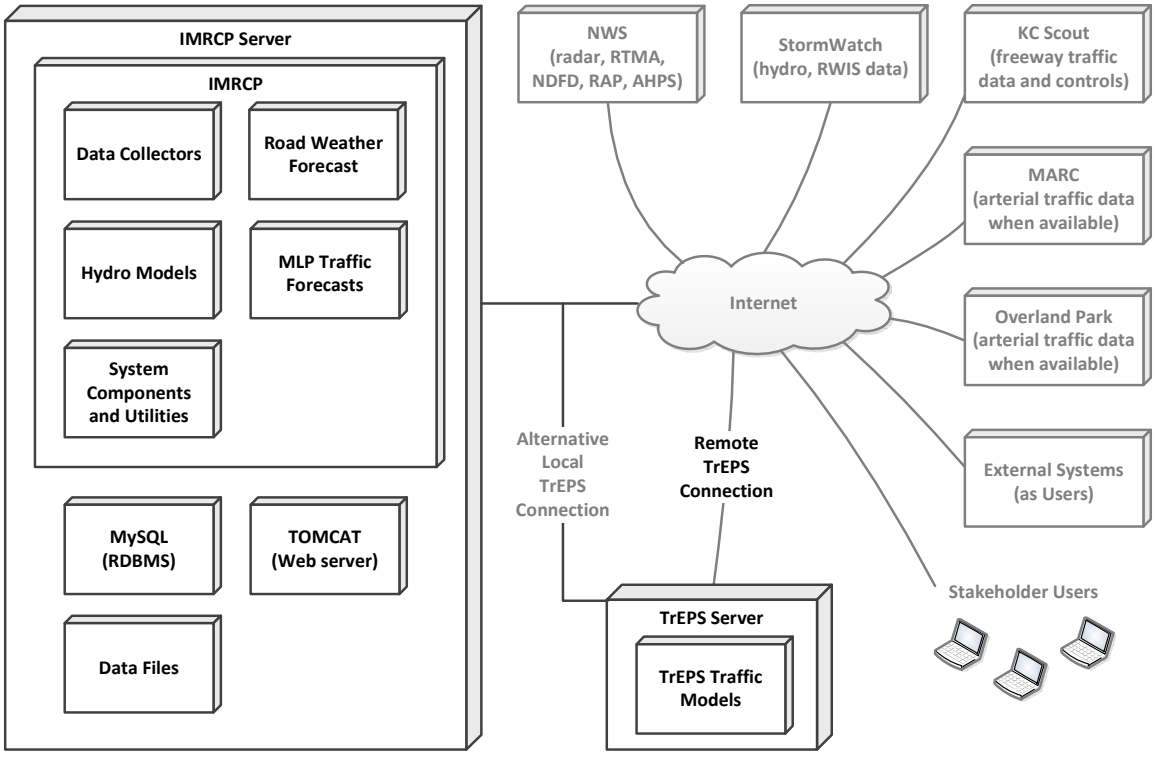
This document provides guidance on the administration and installation of the IMRCP system:

- Chapter 2 describes the server environment as well as installation of IMRCP components.
- Chapter 3 explains the configuration file used for the IMRCP system.
- Chapter 4 illustrates the configuration of the road network, traffic demand, road weather, hydrological, Bayes traffic, and speed statistics models.
- Chapter 5 outlines the continuous operation requirements for the IMRCP system.
- Chapter 6 details the maintenance required for the IMRCP system.
- Chapter 7 lists the references used in this guide.
- Appendix A provides an example excerpt of the IMRCP system log.
- Appendix B documents the names, types, and definitions of parameters in the IMRCP system configuration file.
- Appendix C details the installation of the DYNASMART-X components for the TrEPS model.
- Appendix D lists the acronyms used throughout this document.

Chapter 2. Installation

Server Environment

The Integrated Modeling for Road Condition Prediction (IMRCP) system package contains all of the IMRCP system components, other than the Traffic Estimation and Prediction System (TrEPS), as deployed for the Kansas City demonstration site. IMRCP and TrEPS share underlying road network configuration information, but are implemented separately for each deployment area. Data exchange between the core IMRCP and TrEPS components consists of operational data files from IMRCP sent to TrEPS and traffic estimation/prediction results from TrEPS read back into IMRCP for the user interface and data archives. Figure 1 shows the deployment of the IMRCP system. A similar deployment scheme would be expected for cases using traffic models other than TrEPS.



Source: FHWA, 2018.

Figure 1. Integrated Modeling for Road Condition Prediction deployment diagram.

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

Integrated Modeling for Road Condition Prediction Server

As described in the *Integrated Modeling for Road Condition Prediction System Design Description* (FHWA, 2019), the IMRCP server is used to provision all of the core system components.

The application/web server used in the demonstration deployment includes 64 64-bit processors, 1 terabyte (TB) of memory, and 9 TB RAID5 solid state disk (SSD) storage. The system is run on the 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 IMRCP:

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

Integrated Modeling for Road Condition Prediction Components

As shown in Figure 1, the IMRCP server itself contains four server packages to support the functional components. These server packages are a relational database management system (RDBMS), a web application server, the file system, and the IMRCP server package. A more complete view of the functional components and dependencies supporting data acquisition, forecasting, and user functions is shown in Figure 2. The IMRCP System Design Description (FHWA, 2019) provides a complete architectural view of the as-built system.

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

1. First, an RDBMS must be selected and installed. For the demonstration deployment, MariaDB was used. The process of installing an 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 are 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 demonstration deployment. Once the web application server is installed, move the SDK/deploy/IMRCP/ directory into its web applications directory.

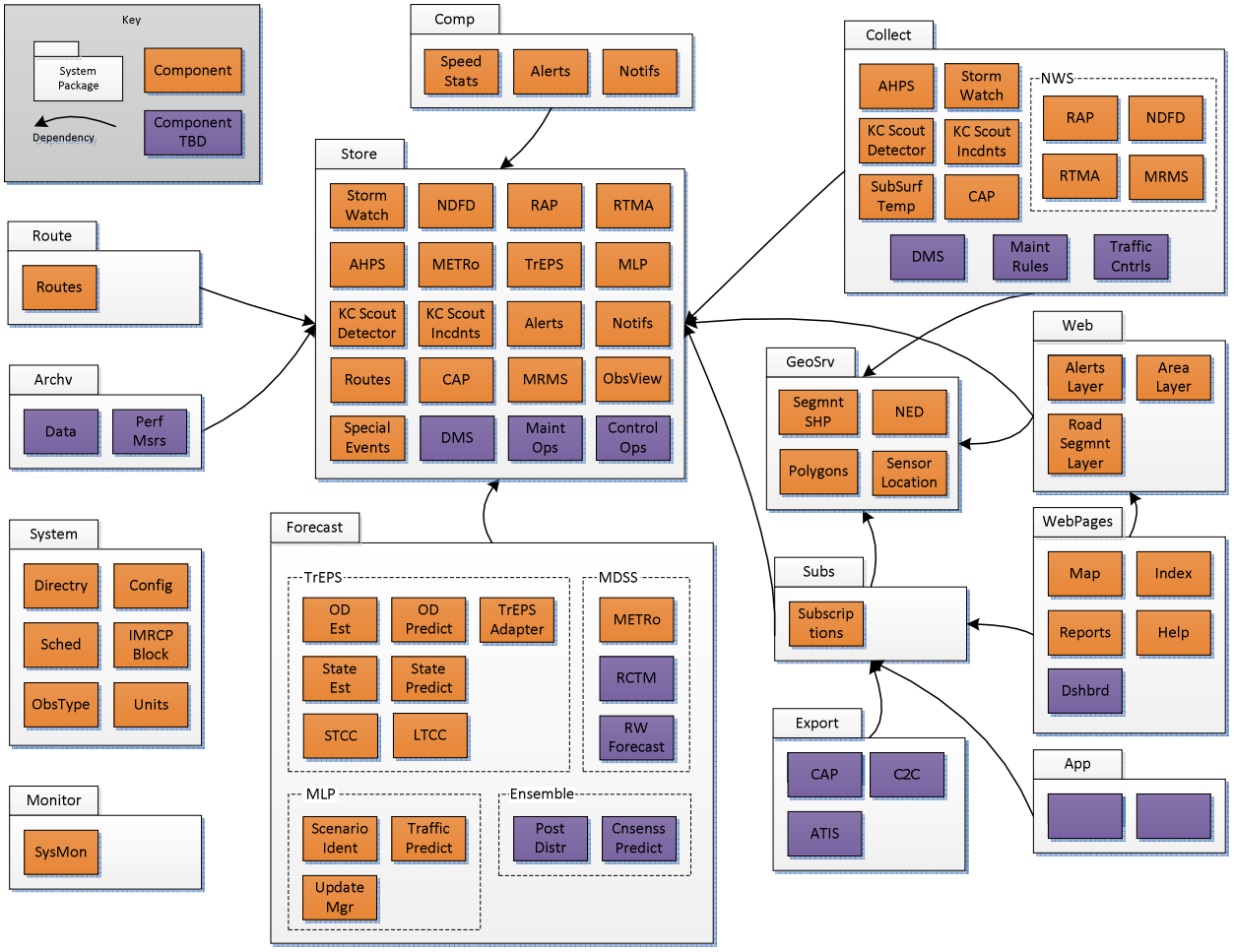
3. The machine learning-based prediction (MLP) traffic forecasts are processed and calculated using R, a free software environment used for statistical computing. The documentation to install R can be found at https://www.r-projgect.org (accessed September 2019). R v3.6.1 was used for demonstration deployment. Once R is installed, a few libraries need to be installed as well. To install the libraries for R, execute the following commands in the R console:

   > install.packages("forecast").
   > install.packages("neuralnet").
   > install.packages("markovchain").
   > install.packages("tictoc").

4. 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.

5. 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 uses generic file paths for many of the components. These can be found in the file by searching for "<desired file path>" and "<desired temp file path>." In 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 run 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.

6. Another configuration item that needs to be set is the map tile service. Mapbox was the service used during 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.



Source: FHWA, 2019.

Figure 2. Integrated Modeling for Road Condition Prediction components.

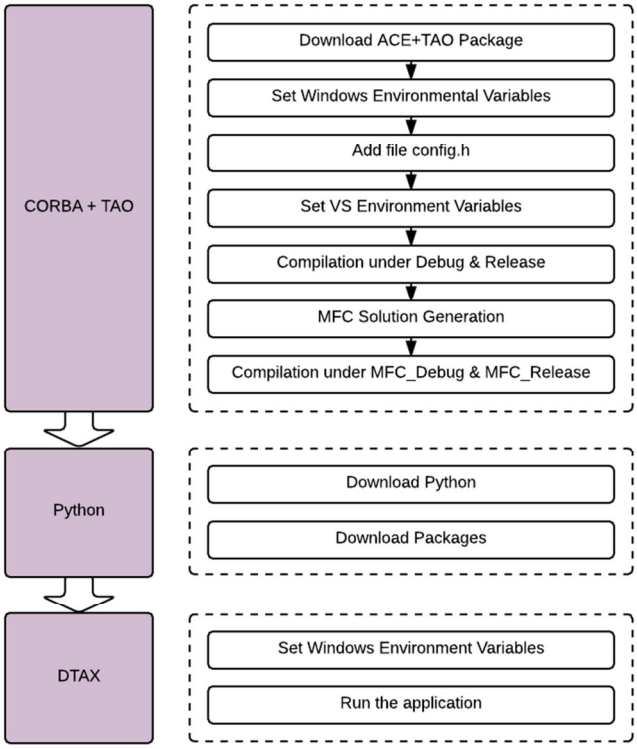
Traffic Estimation and Prediction System Server

The minimum server requirements for installing DYNASMART-X are:

- Platform: Windows XP and later versions.
- Memory: totally 8 gigabyte (GB) random access memory or above on one or multiple computers are recommended.
- Hard Drive Space: minimal 20 GB.
- Recommended display: small fonts, 1024 by 768 screen resolution.

Traffic Estimation and Prediction System Components

Installing DYNASMART-X on a server requires several dependent steps. The overall installation flowchart is shown in Figure 3.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 3. Traffic Estimation and Prediction System 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 Microsoft 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:

- Operating system is Windows 7.
- Machine is 64-bit.
- Microsoft 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. Model Localization and Configuration

Atmospheric Weather Model

Atmospheric weather data and forecasts are provided for Integrated Modeling for Road Condition Prediction (IMRCP) by National Oceanic and Atmospheric Administration (NOAA)/National Weather Service (NWS) sources. As described in the IMRCP System Design Description (FHWA, 2019), the particular NOAA data sources were selected based on the data and their spatio-temporal extents needed to support the road weather condition and traffic models. The default IMRCP atmospheric weather data collection and processing acquire data for the entire contiguous continental United States and do not require any customization for IMRCP deployment within the continental United States.

Road Network Model

The road network model is foundational to the traffic and road weather models. It describes the physical and logical arrangement of roadway segments across the road network. Although each type of behavioral model could conceivably use its own road network, a common road network is used in the IMRCP architecture to assure consistency across the disciplines.

Road network models can be tedious to build, with thousands of segments, links, and nodes. In general, the models can be derived from data in geographical information systems and map databases. For IMRCP demonstration deployment, the OpenStreetMap (OSM) database provided the underlying road network geometry in a set of XML files. The XML files use latitude and longitude coordinates (World Geodetic System 1984 datum) to describe the geometry of the roadway links between the network nodes (intersections, junctions, and splits) at the ends of those links.

The road network model for the greater Kansas City deployment was extracted for IMRCP from the OSM database as XML files. This OSM sub-model included segments designated within the OSM model as motorway, trunk, and primary highways. Some adjustments to the resulting sub-model were made to assure that all road network decision points (splits, junctions, and intersections) were specifically included. As described below, the model was then enhanced beyond the OSM model to adjust the road segment model to accommodate the specific IMRCP traffic and road weather prediction models. A similar extraction and extension process would be used in other IMRCP deployments, with the specifics depending on the quality of the original underlying shape files (or XML) and on the structure and detail in the traffic and weather models.

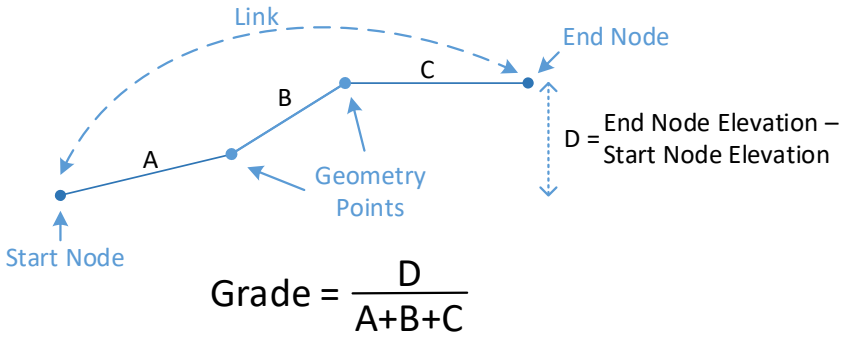
The Traffic Estimation and Prediction System (TrEPS) road network model for the Kansas City deployment was generated from shapefile (SHP) files originating with the Mid-America Regional Council (MARC) planning model that included both highways and arterials across the metropolitan area. The TrEPS model was extracted from the MARC planning model and represents an area of about 10 mi east to west and 7 mi north to south in the southern part of the metropolitan area along a congested interstate highway corridor. The MARC model was then modified to accommodate TrEPS modeling needs. The resulting TrEPS road network model then replaced the corresponding segment definitions in the broader Kansas City road model originally extracted from the OSM database.

The machine learning-based prediction (MLP) model for traffic uses the road network built up from the OSM data and the modified segment definitions needed for TrEPS. This assures that traffic condition forecasts from TrEPS and MLP will apply to identical segment definitions.

Road weather modeling recognizes that some road network segment definitions for traffic purposes still need to be further divided for weather condition forecasting. At least one segment is generated per link from the SHP or XML 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., OSM). Each bridge needs to be its own segment and is identified as such in the segment file, as described in the IMRCP System Design Description (FHWA, 2019).

Links, segments, and nodes are stored in the database with unique system identifiers consistent with the IMRCP identifier 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) were used to look up the elevation at the midpoint of each segment for the Kansas City network. 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.



Source: FHWA, 2019.

Figure 4. Grade calculation.

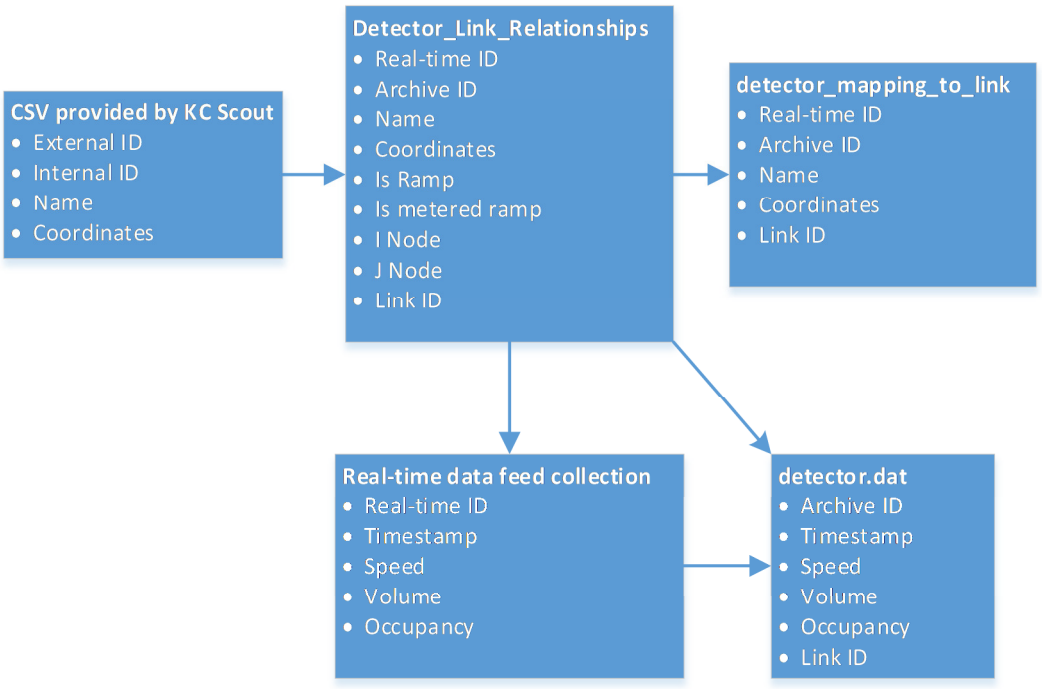
Traffic Detectors

Traffic detectors (i.e., vehicle detection stations) provide real-time and archived speed, volume, and occupancy data to the IMRCP system. These data are used in the offline and online calibration of the traffic models. Detector configuration data are used to assign the archived and real-time detector data to particular roadway segments in the network model.

Source: FHWA, 2019.

Figure 5 illustrates the relationships between the detector configuration files.

[1] The U.S. Geological Survey's "National Map Elevation" Web page is available online at: <https://nationalmap.gov/elevation.html> and was accessed on April 11, 2019.



Source: FHWA, 2019.

Figure 5. Detector configuration data relationships.

For the demonstration deployment, Kansas City Scout provided a comma-separated value (CSV) file that included the geometric coordinates, name, external identifier (used in real-time feed), internal identifier (used in 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 Kansas City Scout. Deployment, configuration, and collection of traffic detector data in any other deployment location will require development of new system components specific to local data sources.

Detector_Link_Relationship.csv lists all of the detectors in the IMRCP study area. The geometric coordinates, name, real-time identifier (Kansas City Scout external identifier), and archive identifier (Kansas City Scout internal identifier) for each of these detectors are included in the Detector_Link_Relationship.csv file. Keeping these metadata accurate and update-to-date is essential to maintaining the quality of the traffic forecasts. 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 based on the physical location of the sensor and inferred location of detection. The I-node, J-node, and link identifiers from the Northwestern University Transportation Center (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 identifier, archive identifier, 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 for the real-time feed. The data are collected from Kansas City Scout's TransSuite real-time data portal as an XML file every 1 min. The data collected from the real-time data portal are from the previous 30 sec. The

real-time data are 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. An example of this real-time file can be found in the IMRCP System Design Description (FHWA, 2019). detector.dat is updated every 1 min and includes the archive identifier, timestamp of each observation, volume, speed, occupancy, and link identifier for each detector for each of the previous 15 min. 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-sec 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.

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 identifier, longitude, latitude, link identifier, upstream node, and downstream node.

Traffic Estimation and Prediction System Traffic Demand Model

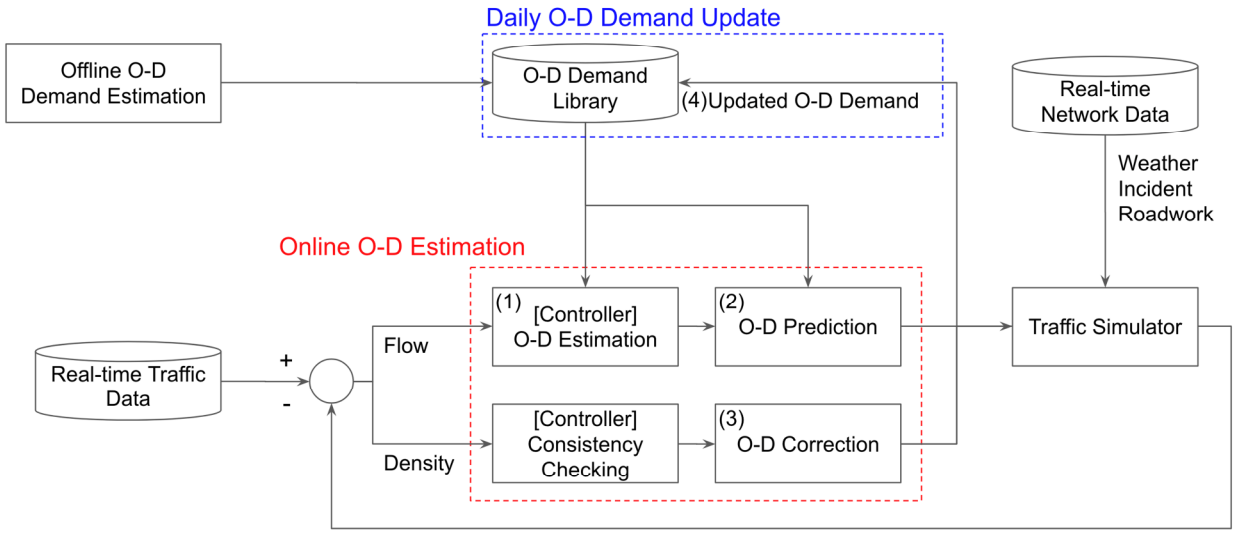
The TrEPS/DYNASMART model is well established and documented in its demonstrations of the impacts of weather on traffic. Its application as part of the IMRCP demonstration for the Kansas City area is described in the IMRCP System Design Description (FHWA, 2019). Detailed descriptions of its calibration for weather applications are beyond the scope of this guide, but can be obtained from prior such applications[2]. This section provides a generalized description of the TrEPS calibration and forecast process, given a prior offline calibration.

TrEPS initiates the traffic simulation with a demand library that includes multiple demand scenarios (e.g., weekdays, weekends, and weather-affected). At the beginning of each day, a priori demand selected with basic information is inserted as input demand to the simulation. As the online system runs, the demand information is updated according to the discrepancy between simulated traffic states and unfolding real-time traffic states measured by traffic detectors. The corrected demand is stored in the demand library on a daily basis and is used to improve a priori demand quality for the next coming days. A framework of the TrEPS online demand learning process is described in Figure 6.

Within Figure 6, at part (1), a priori input from the origin-destination (O-D) demand library enters the O-D estimation module. In this step, flow deviation measured on observed links is converted to the structural deviation of O-D demand, and this parameter is fed to the O-D demand prediction module to adjust the O-D demand matrix. In part (2), the input of O-D demand over the traffic prediction horizon is prepared. The limitation of O-D prediction is that the observation period is too short in comparison to the prediction horizon, and the adjustment parameter remains constant in the prediction period. To devise a time-varying and trend-sensitive intervention in O-D prediction, the predictive O-D adjustment module was developed. The long-term consistency checking (LTCC) module adjusts input demand to keep the

[2] For example, Mahmassani, H.S., Dong, J., Kim, J., Chen, R., and Park, B. 2009. *Incorporating Weather Impacts in Traffic Estimation and Prediction Systems.* Report Number FHWA-JPO-09-065. Washington, DC: FHWA.

estimated state comparable to the unfolding real-time state based on the density discrepancy between the simulation and the observation. In part (3), the O-D adjustment module takes the time series of this adjustment for a rolling horizon from the LTCC module. Then, a time-varying O-D correction is applied during the prediction period. The set of O-D demand matrices, corrected for 24 simulation hours in the running system, is archived in the O-D demand library as an updated a priori demand in part (4). It is designed to learn the most recent demand pattern and use it for next-day simulation in a continuous manner.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 6. Day-to-day learning framework for online origin-destination demand estimation and prediction.

Machine Learning-based Prediction Traffic Model

A machine learning-based prediction (MLP) traffic model has been developed for IMRCP to provide a traffic model based as much as possible on observed traffic and road condition data. The intent has been to reduce the effort required to build and initialize the online model. The Kansas City deployment has provided opportunities to conform the model to multiple road condition data sources, such that the MLP model can conceivably be adapted to any road network model with similar data sets.

The MLP traffic model estimates traffic characteristics on highway segments based on their operational and weather characteristics. As shown in Figure 2, the key components are the data store of historical and current traffic states, the Scenario Identifier (which determines which prediction algorithm to use based on available information), the Update Manager (which updates files needed by the Traffic Predictor), and the Traffic Predictor (which evaluates an anticipated state based on the machine learning). The Scenario Identifier and Traffic Predictor are implemented as scripts run in R. Due to high processor demand, using Rserve to host the MLP processing on a different server than the main IMRCP system server helps ensure that other application requests and processes can continue without resource contention.

The Scenario Identifier uses the metadata files to determine which prediction algorithm to use for each highway segment modeled in the system. Prediction algorithms are based off of the location of traffic detectors relative to the segments. There are three classifications of segments that relate to three different prediction algorithms—segments that have a traffic detector, segments that are downstream of a traffic detector, and segments that are not located near a traffic detector.

The Update Manager updates the files used by the Traffic Predictor on a regular schedule. These files contain rolling windows of the system variables, shown in Table 1, needed by the Traffic Predictor. One set of files contains the long time series of traffic detector speeds for the last month. Another set of files contains a more detailed account of system variables for the last 26 hr for each segment. Each time the Update Manager updates its files, it calls the Scenario Identifier and Traffic Predictor to process the new set of files.

The Traffic Predictor is the main processing component of MLP. A multithreaded approach was implemented to be able to process all of the data within the desired timeframe. Using the information produced by the Scenario Identifier and Update Manager, a speed prediction is calculated for every highway segment. Each highway segment is processed individually; however, if there are not sufficient data for a given segment, a speed prediction will not be produced.

Table 1. Machine learning-based prediction variables.

Node Group	Variable	States	State Definitions
	Link Id		
	Timestamp	YYYY-MM-DD hh:mm	5-min interval timestamps from 01/01/2015 to 12/31/2017
Group 1: Network Environment	Precipitation	1=Clear 2=Light rain 3=Moderate rain 4=Heavy rain 5=Light snow 6=Moderate snow 7=Heavy snow 8=Hail	00 mm/h < 2.5 mm/h 2.5 to 7.6 mm/h ≥ 7.6 mm/h < 1 mm/hr 1 to 2.5 mm/hr > 2.5 mm/hr If hails
	Visibility	1=Clear visibility 2=Reduced visibility 3=Low visibility	Visibility >3300 ft 330 to 3300 ft < 330ft
	Direction	1=Eastbound 2=Southbound 3=Westbound 4=Northbound	
	Temperature (F)	(numeric)	
	Wind speed (mph)	(numeric)	
	Day of week	1=Weekend 2=Weekday	Saturday, Sunday Monday to Friday

Table 1. Machine learning-based prediction variables. (continued)

Node Group	Variable	States	State Definitions
Group 1: Network Environment (cont'd)	Time of day	1=Morning 2=AM peak 3=Off-peak 4=PM peak 5=Night	1AM to 6AM (5 hr) 6AM to 10 AM (4hr) 10AM to 4PM (6hr) 4PM to 8PM (4hr) 8PM to 1AM (5 hr)
	Number of lanes	(integers)	
	Speed limit (mph)	(numeric)	
	Curve on the segment	0=No freeway curve 1=Freeway curve	
	High occupancy vehicle lane	0=No high occupancy vehicle lane 1=High occupancy vehicle lane	
Group 2: Freeway Characteristics	Pavement condition	1=Good 2=At risk 3=Distressed	
	Number of on-ramps	(integers)	
	Number of off-ramps	(integers)	
	Incident on link	0=No incident 1=Incident	
	Incident downstream	0=No incident 1=Incident	
	Number of lanes closed by incidents on link	(integers)	0 if no incident on link (combined number of lanes closed by incidents and number of lanes closed by work zones on link)
	Number of lanes closed by incidents downstream	(integers)	0 if no incident downstream (combined number of lanes closed by incidents and number of lanes closed by work zones on link)
Group 3: External Event	Work zone on link	0=No work zone 1=Work zone	
	Work zone downstream	0=No work zone 1=Work zone	
	Special Events*	1=Special event 0=No special event	Other information related to special event (e.g., types)*
Group 4: Traffic Condition	Flow (veh/h/ln)	(numeric)	
	Speed (mph)	(numeric)	
	Occupancy (percent)	(numeric)	
	Density (percent)	(numeric)	

*Special events were found to not have a significant impact on MLP traffic results for the Kansas City study area.
Source: FHWA, 2019.

Considering that the proposed MLP model is a data-driven method, it can be applied to any road network with a wide variety of external conditions. Archived data over multiple years (2 years at a minimum, depending on the occurrence frequency of weather and other external events) of weather conditions, incident, work zone, and traffic detectors (i.e., speed, volume) are required to recalibrate the model. Although the Kansas City deployment used R, the calibration of the model can be achieved by other similar statistical software. Deployers can make use of the provided R scripts for model calibration and validation with some modifications.

With a calibrated MLP model and real-time data feeds, the traffic prediction can be generated by MLP to capture a dynamic change in traffic conditions. Key steps of the data collection and calibration process adopted in the Kansas City testbed are described in more detail in the IMRCP System Design Description (FHWA, 2019). Note that, while it is recommended to recalibrate the MLP model for any new deployment site, the existing calibrated model can be applied with some accuracy directly to any site when reconfigured to that site's road model. The existing model can also serve as a starting point of the calibration process such that the recalibration at new sites may require fewer data.

The core of MLP is a Markov-based time series model that predicts traffic network conditions by integrating archived and real-time data under various external conditions, including weather, work zones, incidents, and special events. Two-year archived data are used to calibrate the model, and real-time data are required for real-time prediction. The MLP package downloads the archived data for calibration and updates, and uses real-time data, where available, for the prediction process. Online or historical private sector data (e.g., INRIX, HERE) may be applied in the case of lacking detector data.

For traffic with external conditions, a Markov stochastic process is used to estimate the probability that one state transits to another state after a given time period. Two-year archived traffic condition data are required to be categorized into different levels of congestion. The same 2-year system variables (e.g., weather, roadwork, incidents, and traffic control strategies) are clustered into several scenarios based on traffic conditions. The probabilities of transition between traffic states under different external conditions are computed from the observed historical transition frequencies.

Time series models are proposed to predict traffic speeds under normal traffic conditions. Real-time traffic condition feeds and data for 6 hr before the current timestamp are needed to predict the speeds of the next prediction interval in a short-term time series model. The past 2 weeks of traffic condition data are needed to capture the peak-hour traffic patterns and recurring congestion pattern in a long-term time series. The short-term and long-term time series models are combined to forecast the traffic conditions of normal cases to consider weekly/daily trends, and the outputs are further adjusted with the Markov processes for external conditions.

Configuration Files

objects.RData

This file contains the trained road network objects used by R.

markovchains

This file contains markov chains for the road network used by R.

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 "Integrated Modeling for Road Condition Prediction Model Analysis" (Leidos, 2015a), 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.[3] METRo computes the road temperature, subsurface temperature, pavement state, liquid depth, and snow/ice depth.

METRo needs the recent history of the observations, the station configurations, and weather forecast data.

Observations are collected from the sources listed in Table 2.

Table 2. 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 a road weather information system station

Source: FHWA, 2019.
METRo = Model of the Environment and Temperature of Roads.
RTMA = Real-Time Model Assessment

Subsurface temperature data are collected from an environmental sensor station as identified in the system configuration detailed in Table 36 of Appendix B. The subsurface temperature module downloads data from a file transfer protocol site and stores the values on the server. If the file does not contain any data or does not exist, the configured initial value is used for the initial run of METRo. Providing an accurate initial subsurface temperature value will help METRo converge faster.

METRo is configured so that the first 1 hr of the forecast is 1 hr before the current time. METRo requires at least 20 min of data to calibrate its calculations. The 1-hr time shift is necessary to accommodate the minimum amount of data required (the first interval of atmospheric forecast data greater than 20 min) and to synchronize its results for presentation. The forecast data for METRo are collected from the data sources listed in Table 3.

[3] Crevier, L.P., and Y. Delage. 2001. "METRo: A New Model for Road-Condition Forecasting in Canada," *Journal of Applied Meteorology* 40: 2026-2037. Downloaded on April 11, 2019 at: <http://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%282001%29040%3C2026%3AMANMFR%3E2.0.CO%3B2>.

Table 3. Road weather model forecast data sources.

Observation Type	Hour 1 Source	Hours 2 to 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

Source: FHWA, 2019.
METRo = Model of the Environment and Temperature of Roads.
MRMS = Multi-Radar/Multi-Sensor System
NDFD = National Digital Forecast Database
RAP = Rapid Refresh
RTMA =Real-Time Model Assessment

METRo refers to each evaluated segment as a "station." The following station variables are collected from the segment file:

- If the segment is a bridge.
- Latitude.
- Longitude.
- Treatment type (currently untreated for all cases).

These data are sent to the METRo library functions (adapted from a National Center for Atmospheric Research method) using Java Native Interface. All forecast and observation arrays get interpolated from every 1 hr to every 30 sec. 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 min and is executed at each segment/site. METRo is configured for the IMRCP to consider a rain reservoir level greater than 0.2 mm to cause wet pavement.

Hydrological Model

The hydrological model for IMRCP uses data collected from 25 Advanced Hydrological Prediction System (AHPS) stations in the study area— including 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. AHPS station metadata can be found by downloading (as of September 20, 2019) the most recent observation file at: <https://water.weather.gov/ahps/download.php?data=tgz_obs>. After unzipping the .tgz file, all of the stations can be found in the .dbf file and filtered to meet the specific deployment. The specific stations to be monitored within the modeled deployment area are defined in the AHPS Locations and Stages files, as described in Chapter 4.

A CSV file was created using the inundation mapping available (as of September 20, 2019) at: <https://water.weather.gov/ahps/inundation.php> for each of the three specific stations listed above. 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 to 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 1 hr.

For the selected stations for a deployment, the flood stage data are collected every time the NWS updates their observation and forecast files. If the observed or forecasted flood stage is greater than any of the station's thresholds, a flood stage observation is created by the IMRCP system.

Chapter 4. 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 Appendix B tables is the six-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 Appendix B. config.json Definitions.

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 in this file.

Web.xml

This file contains configuration parameters for the Tomcat web application. Its path is application root/WEB-INF/web.xml. It contains the ImrcpBlocks 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 contain the geometric definitions for Federal Information Processing Standard (FIPS) geocodes. These definitions are used for the National Weather Service (NWS) alerts area data layer. For the Kansas City study area, only the FIPS geocodes in and near the area are contained in the configuration file. County boundaries with FIPS codes can be downloaded from the United States Census Bureau's Cartographic Boundary Files at: <https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.2018.html> as accessed on September 20, 2019. This file would require changes for other Integrated Modeling for Road Condition Prediction (IMRCP) locations by determining which geocodes are needed for the deployment and formatting them into the Polygon file format found in the IMRCP System Design Description (FHWA, 2019).

Segments (Localization Required)

The Segments configuration file contains segment definitions based off of the road network model provided by the Northwestern University Transportation Center (NUTC) for the Kansas City study area. OpenStreetMap (OSM) segment definitions were used for the rest of the Kansas City Metro area. Changes must be made to this configuration file for other IMRCP locations. Segment definitions, regardless of the source, must be formatted into the Segment file format found in the IMRCP System Design Description (FHWA, 2019).

log4j2.properties

This configuration file contains the configuration for the logger log4j2. Documentation for the format of this file can be found (as of September 20, 2019) at: <https://logging.apache.org/log4j/2.x/manual/configuration.html>.

Detector Locations (Localization Required)

The Kansas City study area Detector Locations file contains the mappings for the Kansas City Scout detectors to links in the road network model. Once traffic detectors are mapped to segments in the road network, those mappings and detector metadata can be formatted into the Detector Locations file format found in the IMRCP System Design Description (FHWA, 2019). This configuration file contains data specific to the IMRCP 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 routes are specific to the location of the IMRCP deployment, this file must be modified for each location case. Travel times for routes are currently implemented using Traffic Estimation and Prediction System (TrEPS); therefore, if a different traffic model is implemented, another module for travel times must be created to generate travel times using that model. The Routes file format can be found in the IMRCP System Design Description (FHWA, 2019).

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 configuration file must be modified for each of the locations. The Source Units file format can be found in the IMRCP System Design Description (FHWA, 2019).

Advanced Hydrologic Prediction Service (Localization Required)

The Advanced Hydrologic Prediction Service (AHPS) configuration file uses the inundation maps provided by NWS to map road flood depths to study area segments using their latitudes and longitudes. Inundation

maps can be viewed (as of September 20, 2019) at: <https://water.weather.gov/ahps/inundation.php>. NWS does not provide inundation maps for all AHPS stations. Using the inundation map for a specific AHPS station, entries in the AHPS file can be created. The AHPS file format can be found in the IMRCP System Design Description (FHWA, 2019). Different AHPS stations must be used in deployment locations; therefore, the AHPS configuration file must be modified to match the inundation mappings of those locations.

Advanced Hydrologic Prediction Service Locations (Localization Required)

The AHPS Locations file contains the name, identifier, and location of the AHPS stations provided by NWS. Different AHPS stations would be used in deployment locations; therefore, this configuration file must be modified to contain those stations. The AHPS Locations file format can be found in the IMRCP System Design Description (FHWA, 2019).

Advanced Hydrologic Prediction Service Stages (Localization Required)

The AHPS Stages file contains the action, flood, moderate, and major flood stages defined for each station by NWS. Different AHPS stations would be used in deployment locations; therefore, this configuration file must be modified to contain the stages for those stations. The stage metadata can be found in the .dbf file described in the Hydrological Model section of Chapter 3. The AHPS Stages file format can be found in the IMRCP System Design Description (FHWA, 2019).

Machine Learning-Based Prediction Detector Metadata (Localization Required)

The machine learning-based prediction (MLP) Detector Metadata file contains metadata for the road segment(s) mapped to that traffic detector, including the detector identifier, direction of travel, road name, number of on-ramps, number of off-ramps, number of lanes, if the segment is curved, speed limit, if the segment is a ramp, if the segment contains an high occupancy vehicle lane, and the pavement condition. As segment definitions will be different based on the deployment location, this file must be modified for each location case. The MLP Detector Metadata file format can be found in the IMRCP System Design Description (FHWA, 2019).

Downstream Links (Localization Required)

The Downstream Links file should contain an entry for each segment mapped to a traffic detector. Each entry includes the segment identifier, the link identifier that the segment is a part of, and a list of link identifiers that are downstream of the traffic detector. As segment and traffic detector definitions will be different based on the deployment location, this file must be modified for each location case. The Downstream Links file format can be found in the IMRCP System Design Description (FHWA, 2019).

Machine Learning-Based Prediction LinkId Map (Localization Required)

The MLP LinkId Map contains a mapping from MLP link identifiers to IMRCP segment identifiers. As segment definitions will be different based on the deployment location, this file must be modified for each location case. The MLP LinkId Map file format can be found in the IMRCP System Design Description (FHWA, 2019).

Traffic Management Center LinkId Map (Localization Required)

The Traffic Management Center (TMC) LinkId Map contains a mapping from MLP link identifiers to TMC segment codes. As segment definitions will be different based on the deployment location, this file must be modified for each location case. The TMC LinkID Map file format can be found in the IMRCP System Design Description (FHWA, 2019).

Machine Learning-Based Prediction Links Downstream Detectors (Localization Required)

The MLP Links Downstream Detectors file contains an entry for each highway segment traffic detector. The entries list the detector identifier and a list of MLP link identifiers that are downstream from that traffic detector. As traffic detector and link definitions will be different based on the deployment location, this file must be modified for each location case. The MLP Links Downstream Detectors file format can be found in the IMRCP System Design Description (FHWA, 2019).

Stormwatch Locations (Location-Specific)

The Stormwatch Locations file contains an entry for each Stormwatch Station in the Kansas City area. Each entry includes the station name, Stormwatch identifier, IMRCP identifier, latitude, and longitude. The Stormwatch system is specific to the Kansas City area; therefore, it would not be used in other deployment locations.

Stormwatch Devices (Location-Specific)

The Stormwatch Devices file contains an entry for each Stormwatch Device in the Kansas City area. Each entry includes the observation type measured by the device, Stormwatch Station identifier, and Stormwatch Device identifier. The Stormwatch system is specific to the Kansas City area; therefore, it would not be used in other deployment locations.

Stormwatch Stages (Location-Specific)

The Stormwatch Stages file contains the action, flood, moderate, and major flood stages defined for each Stormwatch Station that has a device that measures flood stage and provides the necessary metadata. The Stormwatch system is specific to the Kansas City area; therefore, it would not be used in other deployment locations.

Alerts

The configuration of alert rules can be found in Appendix B under the imrcp.comp.Alerts section. Default rules are included in the example config.json file. Custom rules can be defined following the rules format in that file.

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 Integrated Modeling for Road Condition Prediction (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 Appendix A.

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 administrative privileges have access to additional system functions specified in the system. 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

Federal Highway Administration (FHWA). 2019. *Integrated Modeling for Road Condition Prediction System Design Description.* Report Number FHWA-JPO-18-727. Washington, DC: FHWA.

Leidos. 2015a. "Integrated Modeling for Road Condition Prediction Model Analysis." Unpublished working paper developed under Federal Highway Administration contract DTFH61-12-D-00050, Task Order 5022, Integrated Modeling for Road Condition Prediction, May 10.

Leidos. 2015b. "Integrated Modeling for Road Condition Prediction Concept of Operations." Unpublished working paper developed under Federal Highway Administration contract DTFH61-12-D-00050, Task Order 5022, Integrated Modeling for Road Condition Prediction, November 25.

Leidos. 2016. "Integrated Modeling for Road Condition Prediction System Requirements." Unpublished working paper developed under Federal Highway Administration contract DTFH61-12-D-00050, Task Order 5022, Integrated Modeling for Road Condition Prediction, January 25.

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

Table 4. 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
prefer	string array	List of six-character contributor identifiers used to set display preference for obs types that come from different contributors
<list of contributors>	int	For each contributor identifier in the "prefer" section above, there will be a key and value pair where the key is the contributor identifier and the value is the preference. A smaller value means a higher preference

Source: FHWA, 2019.

Table 5. imrcp.ImrcpBlock.

Key Name	Type	Purpose
box	string array	Bounding box of the study area. The array consists of groups of four strings that represent rectangles that make up the study area. The format of the four strings is left, bottom, right, and 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
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
test	string	"True" if the block should be ran in test mode, otherwise "False"

Source: FHWA, 2019.

All configuration objects that represent ImrcpBlocks inherit these values. The values can be overridden in block-specific configuration.

Table 6. imrcp.store.Store.

Key Name	Type	Purpose
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 cache 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
haspres	string	"True" if this store has a presentation cache, otherwise "False"
contrib	string	Six-character contributor identifier
lrutime	int	Time, in milliseconds, used to determine if a file should be removed from the lru cache. If the file's last used time plus this time is less than the current time, the file is removed
fileoffset	int	If the store contains multiple files per collection period, this is the offset time, in milliseconds, that separates each file

Source: FHWA, 2019.

All ImrcpBlocks that inherit the Store class can specify these configuration values. Instances of Store include KCScoutDetectorsStore, KCScoutIncidentsStore, RTMAStore, RAPStore, RadarStore, RadarPrecipStore, NDFDTdStore, NDFDTempStore, NDFDSkyStore, NDFDWspdStore, AHPSStore, StormWatchStore, RadarPresentationCache, RTMAPresentationCache, RAPPcCatPresentationCache, NDFDTempPresentationCache, RAPPcCatStore, MRMSPcCatStore, MetroStore, TrepsStore, TrvlTimeStore, CAPStore, WeatherAlertsStore, MetroAlertsStore, AHPSAlertsStore, StormwatchAlertsStore, WeatherNotificationsStore, IncidentNotificationsStore, MetroNotificationsStore, AHPSNotificationsStore, StormwatchNotificationsStore, and MLPStore.

Table 7. imrcp.store.WeatherStore.

Key Name	Type	Purpose
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
filepattern	string	Regular expression used to detect the correct files in the directory
filesperperiod	int	The number of files collected each collection period for this store
fileend	string	The format string used to create the ending of a file name
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

Source: FHWA, 2019.

All ImrcpBlocks that inherit the WeatherStore class can specify these configuration values. Instances of WeatherStore include RTMAStore, RAPStore, RadarStore, RadarPrecipStore, NDFDTdStore, NDFDTempStore, NDFDSkyStore, NDFDWspdStore, and MetroStore.

Table 8. RapNcf.

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

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 9. imrcp.store.NcfWrapper.

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

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 10. imrcp.store.MRMSNcfWrapper.mrms.

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

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 11. imrcp.store.MRMSNcfWrapper.

Key Name	Type	Purpose
raintemp	string	Cutoff temperature, in K, used to determine if precipitation is rain
snowtemp	string	Cutoff temperature, in K, used to determine if precipitation is snow

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 12. imrcp.collect.RemoteGrid.

Key Name	Type	Purpose
url	string	Base URL used to collect the data file
initperiods	int	The number of past collection periods to attempt to initially download
filesperperiod	int	The number of files collected each collection period
fileend	string	The format string used to create the ending of a file name
fileoffset	int	If the block collects multiple files per collection period, this is the

		offset time, in milliseconds, that separates each file

Source: FHWA, 2019.

All ImrcpBlocks that inherit the RemoteGrid class can specify these configuration values. Instances of RemoteGrid include RTMA, RAP, Radar, RadarPrecip, PrecipFlag, NDFDSky, NDFDTd, NDFDTemp, and NDFDWspd.

Table 13. imrcp.collect.MRMS.

Key Name	Type	Purpose
memtime	int	The time, in milliseconds, used to determine if a downloaded file should be loaded into the current files cache

Source: FHWA, 2019.

All ImrcpBlocks that inherit the Multi-Radar/Multi-Sensor System (MRMS) class can specify these configuration values. Instances of MRMS include Radar, RadarPrecip, and PrecipFlag.

Table 14. imrcp.collect.KCScoutDetectors.

Key Name	Type	Purpose
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names
freq	int	How often a new observation file is created in milliseconds
pw	String array	List of encoded passwords to cycle through for the Kansas City Scout site
user	String array	List of usernames to cycle through for the Kansas City Scout site. Must be in corresponding order compared to the pw configuration item

Source: FHWA, 2019.

Table 15. imrcp.collect.KCScoutIncidents.

Key Name	Type	Purpose
dir	string	Directory used to save files

Source: FHWA, 2019.

Table 16. 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

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 17. imrcp.collect.RAPPcCat.

Key Name	Type	Purpose
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names

Source: FHWA, 2019.

Table 18. imrcp.collect.MRMSPcCat.

Key Name	Type	Purpose
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 19. 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
mlpmap	String	The file that contains the MLP link identifier mapping

Source: FHWA, 2019.
MLP = machine learning-based prediction.

Table 20. imrcp.geosrv.NED.

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

Source: FHWA, 2019.
NED = National Elevation Database.

Table 21. imrcp.forecast.treps.RealTimeDetector.

Key Name	Type	Purpose
output	string	Path of the output file
rtoffset	int	The realtime offset in milliseconds
minfile	int	The number of minutes to put in a file

Source: FHWA, 2019.

Table 22. imrcp.forecast.treps.RealTimeIncident.

Key Name	Type	Purpose
output	string	Path of the output file

Source: FHWA, 2019.

Table 23. imrcp.forecast.treps.RealTimeWorkzone.

Key Name	Type	Purpose
output	string	Path of the output file
rate	Int	The capacity reduction rate

Source: FHWA, 2019.

Table 24. imrcp.forecast.treps.RealTimeWeather.

Key Name	Type	Purpose
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

Source: FHWA, 2019.

Table 25. SourceUnits.

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

Source: FHWA, 2019.

Table 26. imrcp.forecast.mdss.Metro.

Key Name	Type	Purpose
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
details	string	String used to create the SimpleDateFormat that is used to generate metro detail file names

Source: FHWA, 2019.
METRo = Model of the Environment and Temperature of Roads.

Table 27. DoMetroWrapper.

Key Name	Type	Purpose
roadmin	Int	Minimum road temperature allowed in C
roadmax	Int	Maximum road temperature allowed in C
ssmin	Int	Minimum subsurface temperature allowed in C
ssmax	Int	Maximum subsurface 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 meters per second
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	Threshold of the rain reservoir in mm to determine pavement state
snowcut	String	Threshold of the snow/ice reservoir in millimeters to determine pavement state

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 28. imrcp.subs.Subscription.

Key Name	Type	Purpose
subsRoot	String	Base directory where subscription information is saved

Source: FHWA, 2019.

Table 29. 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 identifiers
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 identifier, minimum value, maximum value, and units. Minimum and maximum values can be a double, an empty string if there is not a minimum or maximum 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 identifiers 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

Source: FHWA, 2019.

All ImrcpBlocks that inherit the Alerts class can specify these configuration values. Instances of alerts include WeatherAlerts, TrafficAlerts, MetroAlerts, AHPSAlerts, and StormwatchAlerts.

Table 30. imrcp.alert.AlertsStore.

Key Name	Type	Purpose
recover	string	Path to the file used to save alerts that are in memory if the system is shutdown

Source: FHWA, 2019.

All ImrcpBlocks that inherit the AlertsStore class can specify these configuration values. Instances of AlertsStore include WeatherAlertsStore, TrafficAlertsStore, MetroAlertsStore, SpeedStatsAlertsStore, AHPSAlertsStore, StormwatchAlertsStore, WeatherNotificationsStore, IncidentNotificationsStore, MetroNotificationsStore, AHPSNotificationsStore, and StormwatchNotificationsStore.

Table 31. imrcp.system.Units.

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

Source: FHWA, 2019.

Table 32. imrcp.forecast.treps.OutputManager.

Key Name	Type	Purpose
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

Source: FHWA, 2019.
TrEPS = Traffic Estimation and Prediction System.

Table 33. 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

Source: FHWA, 2019.
ftp = file transfer protocol.
ip = internet provider.
TrEPS = Traffic Estimation and Prediction System.

This does not inherit ImrcpBlock.

Table 34. imrcp.forecast.treps.TrepsCollect.

Key Name	Type	Purpose
files	string array	Array of files to check if they are ready to be collected from the TrEPS ftp

Source: FHWA, 2019.
ftp = file transfer protocol.
TrEPS = traffic estimation and prediction system.

All ImrcpBlocks that inherit the TrepsCollect class can specify these configuration values. Instances of TrepsCollect include DefaultTrepsCollect and VehTrepsCollect.

Table 35. imrcp.collect.AHPS.

Key Name	Type	Purpose
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 contain 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
field	String	Name of the field in the dbf file that contains the desired data
search	String	Name of the file to search for to check for the updated timestamp
metadata	string	Path of the AHPS stages metadata file

Source: FHWA, 2019.
AHPS = Advanced Hydrologic Prediction Service.

Table 36. imrcp.collect.SubSurfaceTemp.

Key Name	Type	Purpose
file	String	Path to the file where observations are written
id	Int	The site identifier
user	String	User name for the ftp site
pw	String	Password for the ftp site
ftp	String	URL of the ftp site
init	String	The subsurface value to initially use if a value is not in the file

Source: FHWA, 2019.
ftp = file transfer protocol.

Table 37. imrcp.collect.CAP.

Key Name	Type	Purpose
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

Source: FHWA, 2019.
CAP = Common Alerting Protocol

Table 38. imrcp.store.CAPStore.

Key Name	Type	Purpose
fips	string	The file that contains the fips codes and other polygon definitions used by CAP

Source: FHWA, 2019.
CAP = Common Alerting Protocol

Table 39. Imrcp.geosrv.Polygons.

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

Source: FHWA, 2019.

Table 40. 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
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
stages	string	Path to the file that contains the stormwatch stages metadata
timeout	string	Time, in milliseconds, for the connection to timeout

Source: FHWA, 2019.

Table 41. imrcp.route.Routes.

Key Name	Type	Purpose
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
timeout	string	Time to wait, in milliseconds, while checking if TrEPS files have been updated

Source: FHWA, 2019.
TrEPS = Traffic Estimation and Prediction System.

Table 42. route.

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

Source: FHWA, 2019.

This does not inherit ImrcpBlock.

Table 43. rangerules_<obstypeid>.

Key Name	Type	Purpose
nan	string	The value that represents NaN or missing values
delete	string array	The range of numbers to ignore. Ranges are described by two strings, the first being the lower limit (inclusive), and the second being the upper limit (exclusive). There can be multiple delete ranges in the array
ranges	string array	The “buckets” that the values get placed in to group the data. Ranges are described by two strings, the first being the lower limit (inclusive), and the second being the upper limit (exclusive). There can be multiple ranges in the array

Source: FHWA, 2019.

This does not inherit ImrcpBlock. There must be a rangerules entry for each obs type used to create a Presentation cache. Obstypeid that need a configuration entry right now are RDR0, PCCAT, TAIR, SPDWND.

Table 44. 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 that have the same notification
file	string	Path to the temporary file that contains the current notifications

Source: FHWA, 2019.

All ImrcpBlocks that inherit the Notifications class can specify these configuration values. Instances of Notifications include WeatherNotificaitons, IncidentNotifications, MetroNotifications, AHPSNotifications, and StormwatchNotifications.

Table 45. imrcp.store.PresentationCache.

Key Name	Type	Purpose
invert	String	“True” if the y-axis is inverted; otherwise, it is “False”
tol	string	The tolerance value used by Orthogons

Source: FHWA, 2019.

All ImrcpBlocks that inherit the PresentationCache class can specify these configuration values. Instances of PresentationCache include RTMAPresentationCache, RadarPresentationCache, RAPPcCatPresentationCache, MRMSPcCatPresentationCache, NDFDTempPresentationCache, and NDFDWspdPresentationCache.

Table 46. NotificationServlet.

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

Source: FHWA, 2019.

Table 47. imrcp.geosrv.SensorLocations.

Key Name	Type	Purpose
file	string	Path of the file that contains the metadata for the sensors
mapvalue	int	Identifier used for map presentation

Source: FHWA, 2019.

Table 48. imrcp.forecast.mlp.MLPBlock.

Key Name	Type	Purpose
rdata	string	Path to the R data file
robj	string	Path to the R object file
markov	string	Path to the markov chains file
host	string	IP address of the server hosting the R processing
threads	int	The number of threads to use for R processing
metadata	string	Path to the MLP Detector metadata file
downstream	string	Path to the Downstream Links metadata file
linkmap	string	Path to the Link metadata file
tmcmap	string	Path to the tmc mapping metadata file
tmclocdir	string	Path to the local tmc directory
tmchostdir	string	Path to the host tmc directory
raintemp	string	Cutoff temperature, in K, used to determine if precipitation is rain
snowtemp	string	Cutoff temperature, in K, used to determine if precipitation is snow
locdir	string	Path to the local directory of data files
hostdir	string	Path to the host directory of data files
loctsdir	string	Path to the local directory of long time series files
hosttsdir	string	Path to the host directory of long time series files

Source: FHWA, 2019.
IP = internet provider.
MLP = machine learning-based prediction.
TMC = traffic management center.

All ImrcpBlocks that inherit the MLPBlockclass can specify these configuration values. Instances of MLPBlock include MLPUpdate and MLPPredict.

Table 49. imrcp.forecast.mlp.MLPPredict.

Key Name	Type	Purpose
file	string	String used to create the SimpleDateFormat that is used to generate destination file names

Source: FHWA, 2019.

Table 50. imrcp.forecast.mlp.MLPUpdate.

Key Name	Type	Purpose
inrixdata	string	Path of the INRIX data file

Source: FHWA, 2019.

Table 51. imrcpBlocks that need configuration for localization.

Instance Name	Classes
NED	imrcp.geosrv.NED
KCScoutDetectorLocations	imrcp.geosrv.SensorLocations, imrcp.geosrv.KCScoutDetectorLocations
StormwatchLocation	imrcp.geosrv.SensorLocations, imrcp.geosrv.StormwatchLocations
AHPSLocations	imrcp.geosrv.SensorLocations, imrcp.geosrv.AHPSLocations
SegmentShps	imrcp.geosrv.SegmentShps
Polygons	imrcp.geosrv.Polygons
AHPS	imrcp.collect.AHPS
StormWatch	imrcp.collect.StormWatch
RTMA	imrcp.collect.RemoteGrid, imrcp.collect.RTMA
NDFDTd	imrcp.collect.RemoteGrid, imrcp.collect.NDFD
NDFDTemp	imrcp.collect.RemoteGrid, imrcp.collect.NDFD
NDFDSky	imrcp.collect.RemoteGrid, imrcp.collect.NDFD
NDFDWspd	imrcp.collect.RemoteGrid, imrcp.collect.NDFD
RAP	imrcp.collect.RemoteGrid, imrcp.collect.RAP
Radar	imrcp.collect.RemoteGrid, imrcp.collect.MRMS
RadarPrecip	imrcp.collect.RemoteGrid, imrcp.collect.MRMS
PrecipFlag	imrcp.collect.RemoteGrid, imrcp.collect.MRMS
KCScoutDetectors	imrcp.collect.KCScoutDetectors
KCScoutIncidents	imrcp.collect.KCScoutIncidents
CollectRetry	imrcp.collect.CollectRetry
SubSurfaceTemp	imrcp.collect.SubSurfaceTemp
RAPPcCat	imrcp.collect.RAPPcCat
MRMSPcCat	imrcp.collect.MRMSPcCat
CAP	imrcp.collect.CAP
KCScoutDetectorsStore	imrcp.store.Store, imrcp.store.KCScoutDetectorsStore
KCScoutIncidentsStore	imrcp.store.Store, imrcp.store.KCScout.IncidentStore
RTMAStore	imrcp.store.Store, imrcp.store.WeatherStore
RAPStore	imrcp.store.Store, imrcp.store.WeatherStore, imrcp.store.RAPStore
RadarStore	imrcp.store.Store, imrcp.store.WeatherStore, imrcp.store.MRMSStore
RadarPrecipStore	imrcp.store.Store, imrcp.store.WeatherStore, imrcp.store.MRMSStore
NDFDTdStore	imrcp.store.Store, imrcp.store.WeatherStore
NDFDTempStore	imrcp.store.Store, imrcp.store.WeatherStore
NDFDSkyStore	imrcp.store.Store, imrcp.store.WeatherStore
NDFDWspdStore	imrcp.store.Store, imrcp.store.WeatherStore
AHPSStore	imrcp.store.Store, imrcp.store.AHPSStore
StormWatchStore	imrcp.store.Store, imrcp.store.StormWatchStore
Metro	imrcp.forecast.mdss.Metro
RadarPresentationCache	imrcp.store.Store, imrcp.store.PresentationCache
RTMAPresentationCache	imrcp.store.Store, imrcp.store.PresentationCache
RAPPcCatPresentationCache	imrcp.store.Store, imrcp.store.PresentationCache
MRMSPcCatPresentationCache	imrcp.store.Store, imrcp.store.PresentationCache

Table 51. imrcpBlocks that need configuration for localization.(continued)

Instance Name	Classes
NDFDTempPresentationCache	imrcp.store.Store, imrcp.store.PresentationCache
ObsView	imrcp.store.ObsView
RAPPcCatStore	imrcp.store.Store, imrcp.store.PcCatStore
MRMSPcCatStore	imrcp.store.Store, imrcp.store.PcCatStore
MetroStore	imrcp.store.Store, imrcp.store.WeatherStore, imrcp.store.MetroStore
DefaultTrepsCollect	imrcp.forecast.treps.TrepsCollect
VehTrepsCollect	imrcp.forecast.treps.TrepsCollect
OutputManager	imrcp.forecast.treps.OutputManager
RealTimeDetector	imrcp.forecast.treps.RealTimeDetector
RealTimeWeather	imrcp.forecast.treps.RealTimeWeather
RealTimeIncident	imrcp.forecast.treps.RealTimeIncident
RealTimeWorkzone	imrcp.forecast.treps.Workzone
TrepStore	imrcp.store.Store, imrcp.store.TrepsStore
TrvlTimeStore	imrcp.store.Store, imrcp.store.TrvlTimeStore
CAPStore	imrcp.store.Store, imrcp.store.CAPStore
Routes	imrcp.route.Routes
Subscriptions	imrcp.subs.Subscriptions
WeatherAlertsStore	imrcp.store.Store, imrcp.store.AlertsStore
WeatherAlerts	imrcp.alert.Alerts
MetroAlertsStore	imrcp.store.Store, imrcp.store.AlertsStore
MetroAlerts	imrcp.alert.Alerts
AHPSAlerts	imrcp.alert.Alerts
AHPSAlertsStore	imrcp.store.Store, imrcp.store.AlertsStore
StormwatchAlertsStore	imrcp.store.Store, imrcp.store.AlertsStore
StormwatchAlerts	imrcp.alert.Alerts
WeatherNotifications	imrcp.alert.Notifications
IncidentsNotifications	imrcp.alert.Notifications
MetroNotifications	imrcp.alert.Notifications
AHPSNotifications	imrcp.alert.Notifications
StormwatchNotifications	imrcp.alert.Notifications
WeatherNotificationsStore	imrcp.store.Store, imrcp.store.AlertsStore
IncidentNotificationsStore	imrcp.store.Store, imrcp.store.AlertsStore
MetroNotificationsStore	imrcp.store.Store, imrcp.store.AlertsStore
AHPSNotificationsStore	imrcp.store.Store, imrcp.store.AlertsStore
StormwatchNotificationsStore	imrcp.store.Store, imrcp.store.AlertsStore
MLPStore	imrcp.store.Store, imrcp.store.MLPStore
MLPPredict	imrcp.forecast.mlp.MLPBlock, imrcp.forecast.mlp.MLPPredict
MLPUpdate	imrcp.forecast.mlp.MLPBlock, imrcp.forecast.mlp.MLPUpdate

Source: FHWA, 2019.

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

Environment Settings

Installation of TAO

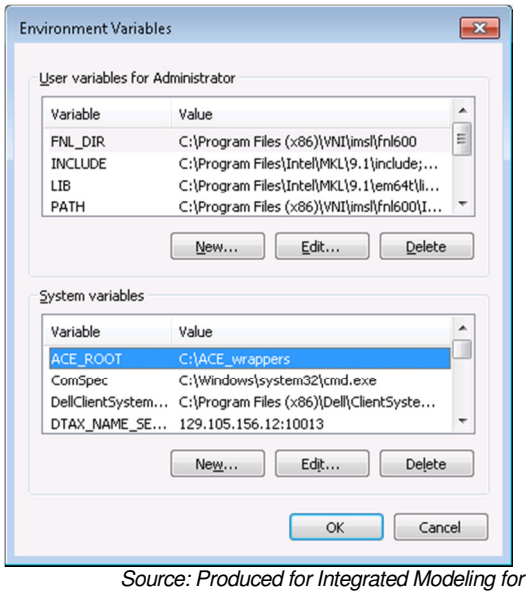
DYNASMART-X is a distributed Traffic Estimation and Prediction System software tool that consists of several different functional modules. All the components within it are brought together through Common Object Request Broker Architecture (CORBA), which is a protocol/specification/standard that allows 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++.

To run DYNASMART-X successfully on a machine, TAO needs to be installed before installing DYNASMART-X.

1. Download ACE+TAO package (Version 6.0.1) from the 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 (Figure 7).

   Right click "Computer" → "Properties" → "Advanced system settings" → "Environment Variables."



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 7. Windows environment variables window.

Under “System Variables,” click “New” and add the following variables with their corresponding values.

Table 52. Environment variables for TAO.

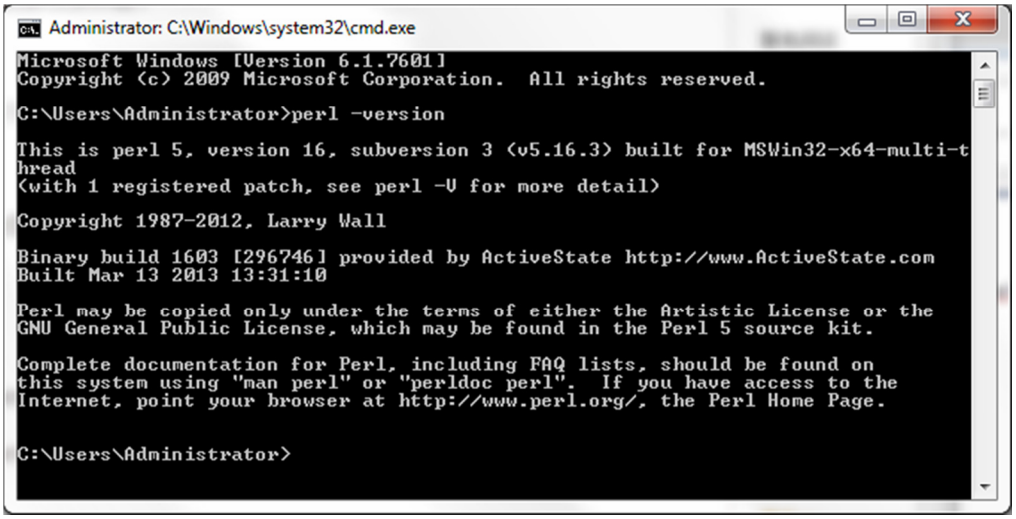
Variable Name	Variable Value
ACE_ROOT	C:\ACE_wrappers
TAO_ROOT	%ACE_ROOT%\TAO
PATH (add values on the right)	%ACE_ROOT%; %ACE_ROOT%\TAO; %ACE_ROOT%\bin; %ACE_ROOT%\lib; %ACE_ROOT%\TAO\orbsvcs

Source: FHWA, 2019.
Remarks: when adding variables into PATH, there should be no spaces between variable names (i.e., it should look like: %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. 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 the “x64” platform. After building is completed, close Visual Studio.
8. Generate ACE’s Microsoft Foundation Class (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 the Command window. Type “Perl -version” in the Command window to check if Perl is installed successfully. Figure 8 displays how the current version will be shown if Perl is installed.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

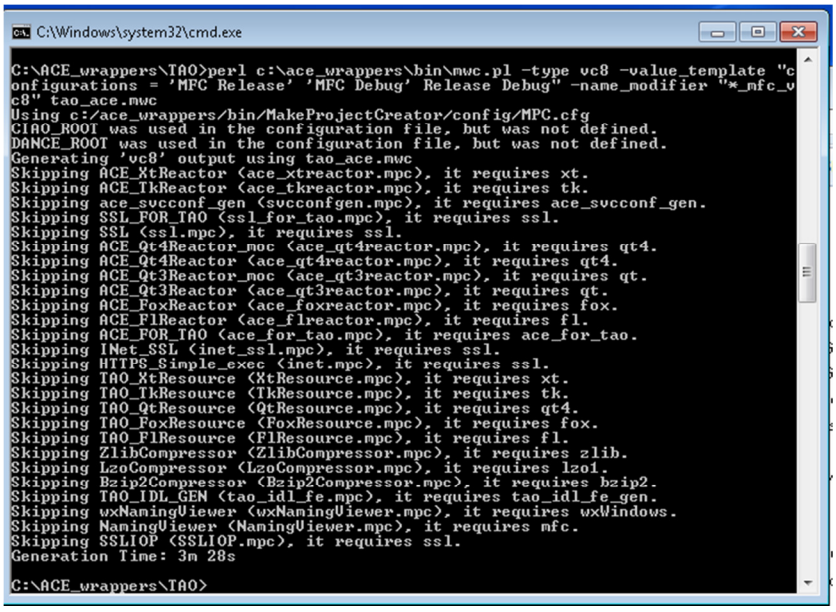
Figure 8. ActivePerl version.

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).

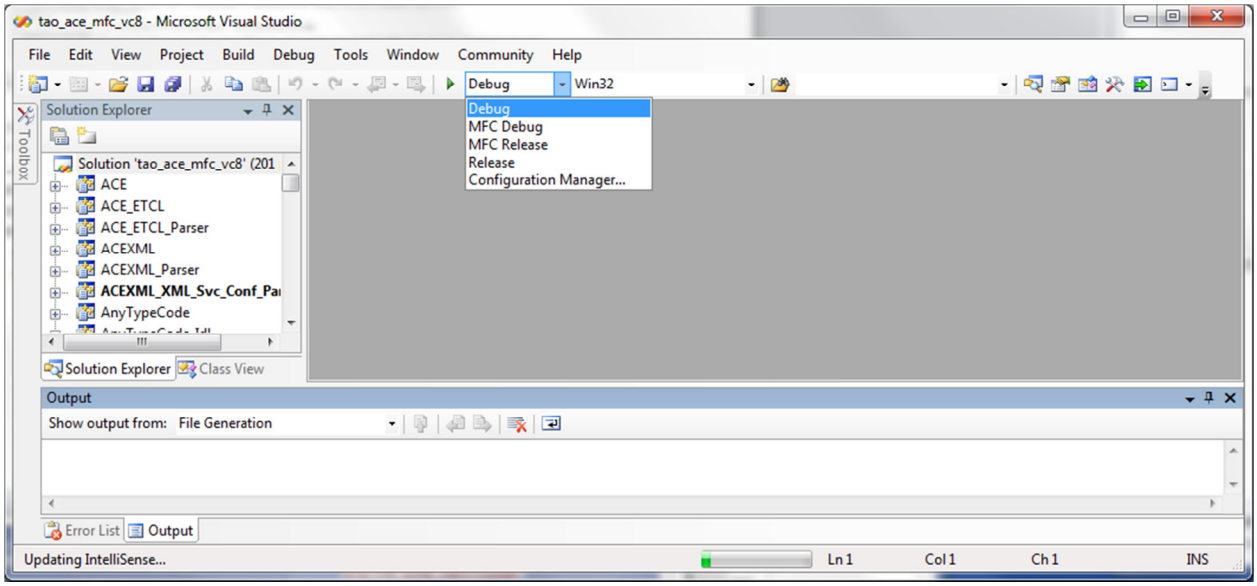
e. Enter the following command (Figure 9): **perl c:\ace_wrappers\bin\mwc.pl –type vc8 –value_template "configurations = "MFC Release" "MFC Debug" Release Debug" –name_modifier "*_mfc_vc8" tao_ace.mwc**



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 9. MFC generation process.

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



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

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:

      Tools → Options → Projects and Solutions → VC++ Directories.
   c. These changes are made on the "x64" platform.

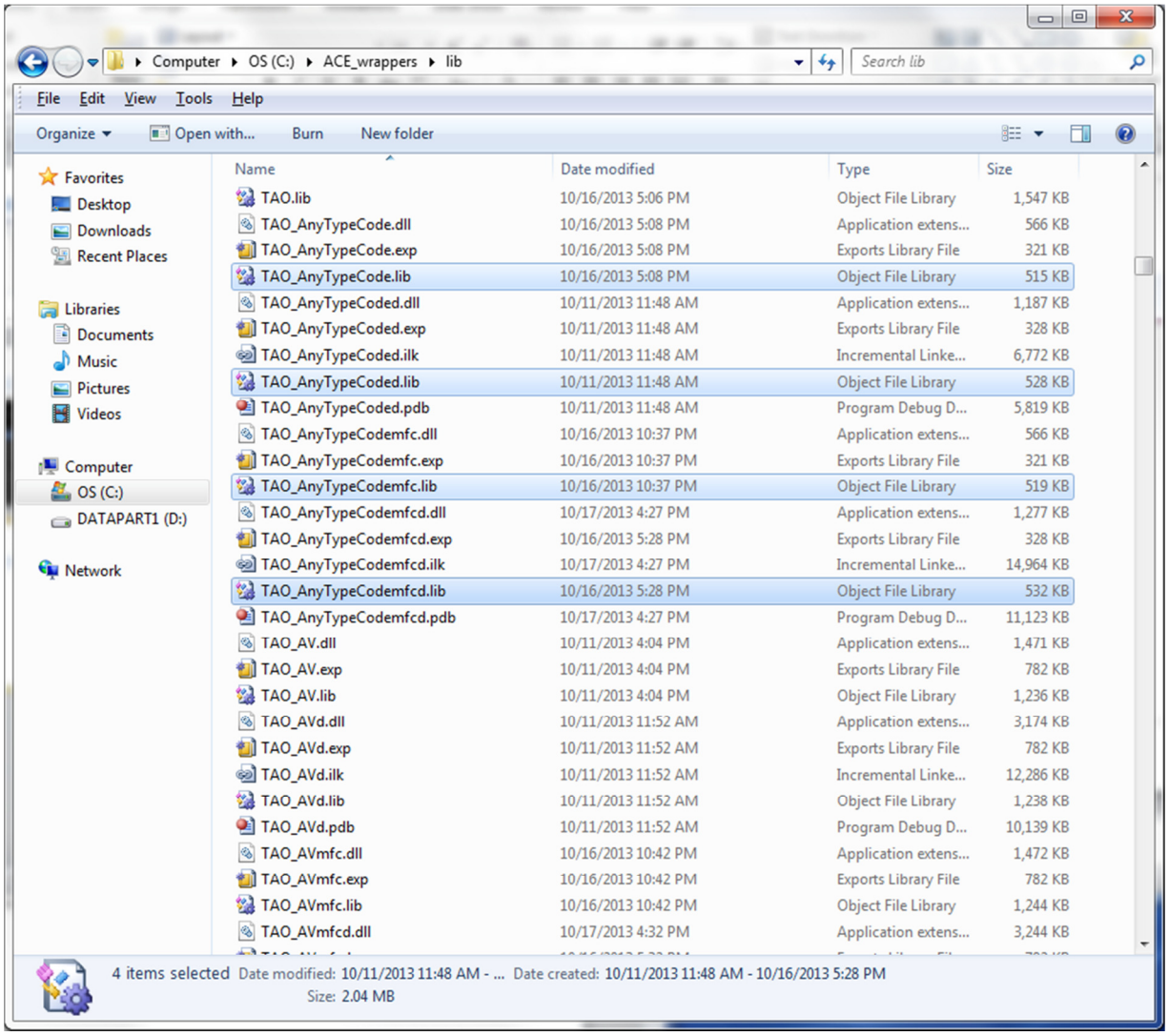
Table 53. ACE_TAO visual studio setting.

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

Source: FHWA, 2019.
Note: 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.

   d. Open "**TAO_ACE_vc8.sln**," rebuild "TAO_IDL_EXE" and "Naming Service" projects under both "Release" and "Debug" configurations, on the "x64" platform.
   e. 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 (Figure 11). 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.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

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
  – Internet protocol 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 /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 Integrated Modeling for Road Condition Prediction project and are provided separately:

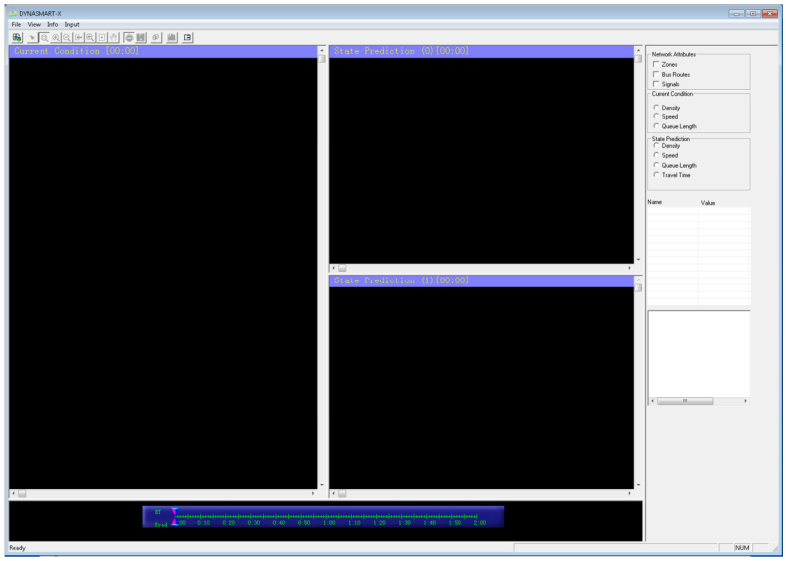
1. Northwestern University Transportation Center. 2016. *Integrated Modeling for Road Condition Prediction – DYNASMART TrEPS Input Files Description*. Technical Memorandum. June 28.
2. Northwestern University Transportation Center. 2016. *Integrated Modeling for Road Condition Prediction – DYNASMART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

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.

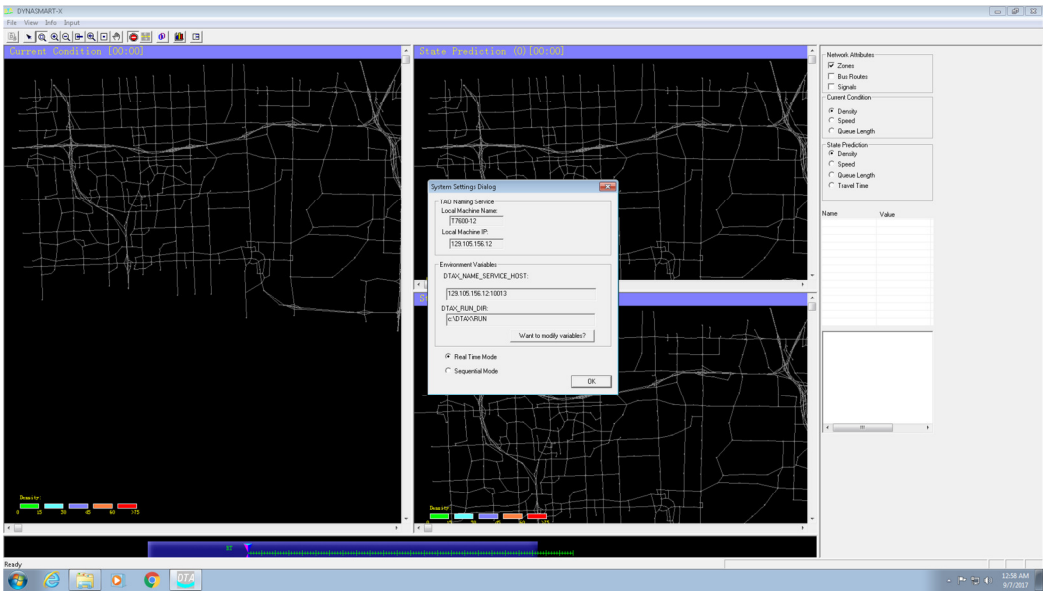
The window will open (Figure 12). It is comprised of a pull-down menu, toolbar, simulation view, clock view, and information view.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 12. Initial graphical user interface of DYNASMART-X.

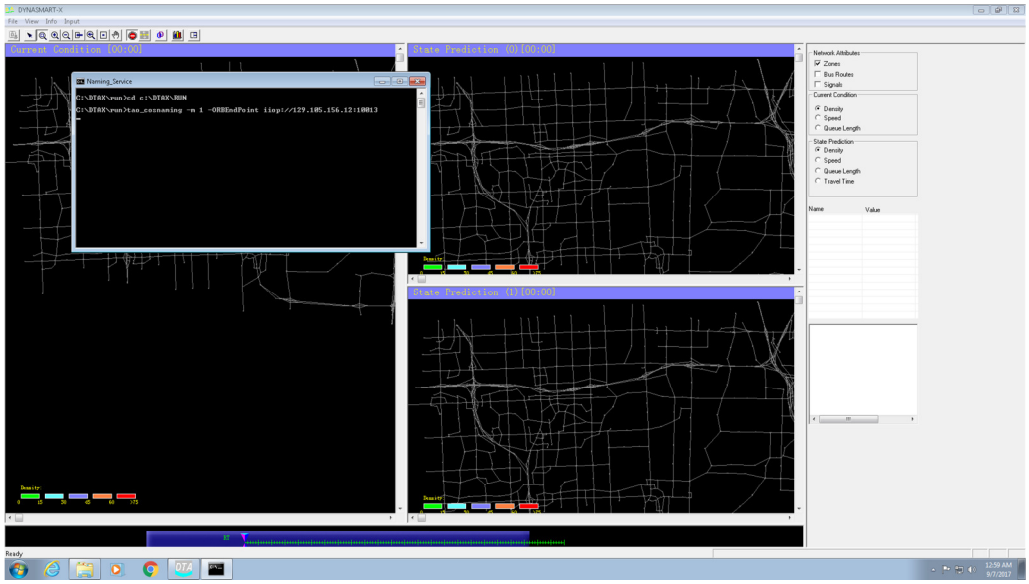
To initiate DTAX, go to “File” on the DYNASMART-X window menu, then to “Load Data,” and click “1 Check System Settings.” When it displays the window (Figure 13), choose “Real Time Mode” instead of “Sequential Mode.”



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 13. DYNASMART mode selection window.

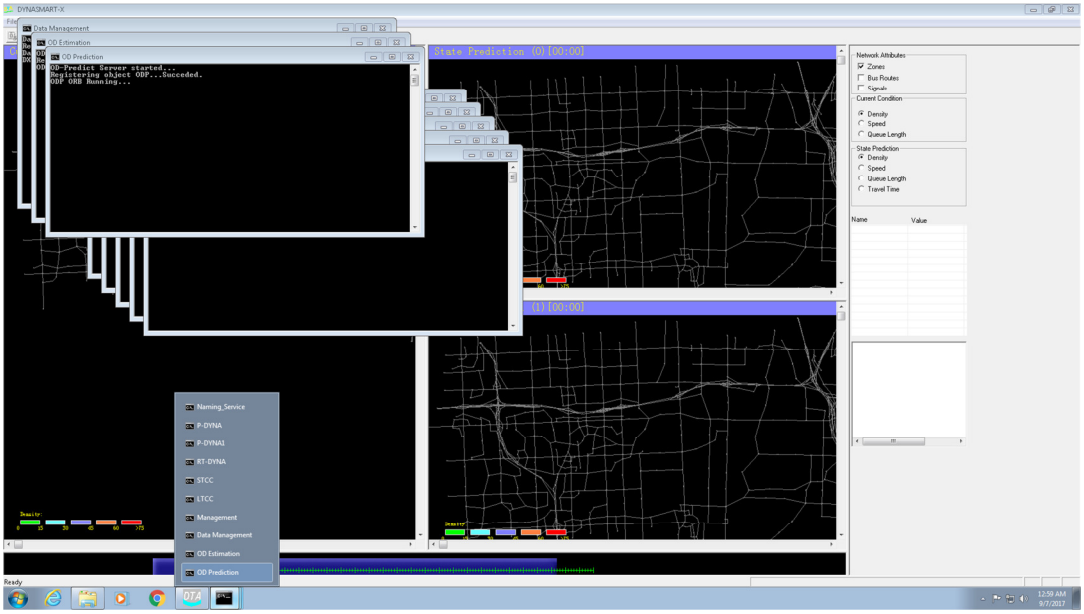
Next, click “2 Naming Service”; a window will open (Figure 14).



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 14. DYNASMART naming service window.

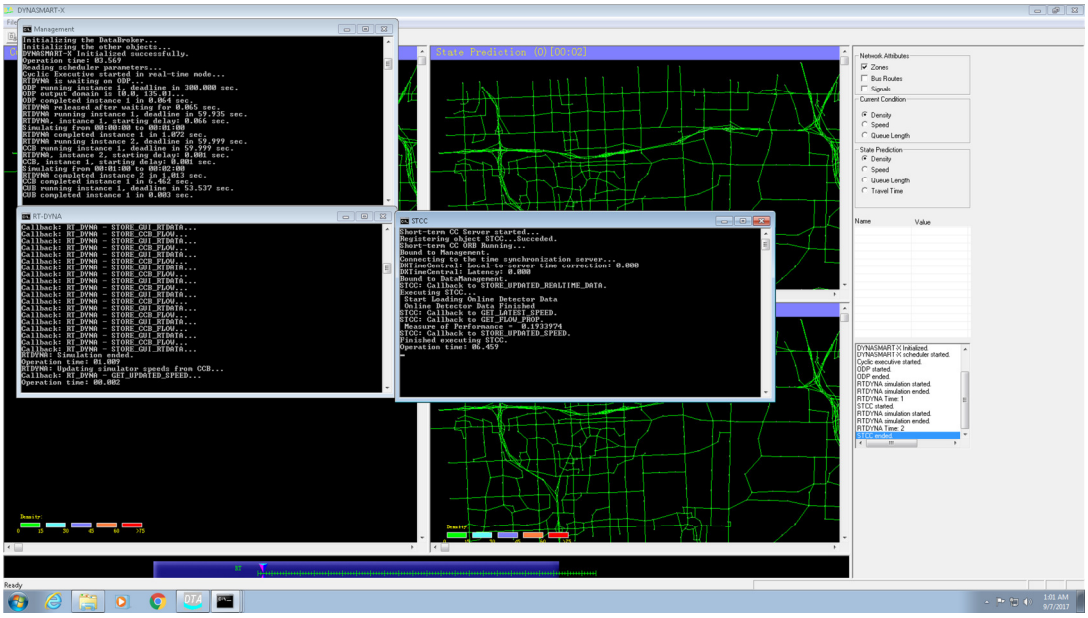
Skip the third item in the menu list and click “4 Active Models on Server Machine”; this will initiate several modules (Figure 15).



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

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, a window will appear (Figure 16). The right bottom box indicates which module is currently running in the model.



Source: Produced for Integrated Modeling for Road Condition Prediction Project by NUTC, 2019.

Figure 16. Completed DYNASMART initialization.

Appendix D. Glossary

AHPS	Advanced Hydrologic Prediction Service
CAP	Common Alerting Protocol
ConOps	Concept of Operations
CSV	Comma-Separated Value
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
GB	Gigabyte
IMRCP	Integrated Modeling for Road Condition Prediction
IP	Internet Provider
LTCC	Long-Term Consistency Checking
MARC	Mid-America Regional Council
METRo	Model of the Environment and Temperature of Roads
MFC	Microsoft Foundation Class
MLP	Machine Learning-based Prediction
MRMS	Multi-Radar/Multi-Sensor System
NDFD	National Digital Forecast Database
NED	National Elevation Database
NOAA	National Oceanic and Atmospheric Administration
NUTC	Northwestern University Transportation Center
NWS	National Weather Service
O-D	Origin-Destination
OSM	OpenStreetMap
RAP	Rapid Refresh
RTMA	Real-Time Model Assessment
RDBMS	Relational Database Management System
SHP	Shapefile
SSD	Solid State Disc
TB	Terabyte
TMC	Traffic Management Center
TrEPS	Traffic Estimation and Prediction System
TSMO	Transportation System Management and Operations
USDOT	U.S. Department of Transportation

U.S. Department of Transportation
ITS Joint Program Office – HOIT
1200 New Jersey Avenue, SE
Washington, DC 20590

Toll-Free “Help Line” 866-367-7487

www.its.dot.gov

U.S. Department of Transportation