Integrated Modeling for Road Conditions Prediction, Phase 5: Installation and Administration Guide

September 10, 2025

Final

U.S. Department of Transportation
Federal Highway Administration

SI* (MODERN METRIC) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		AREA		
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
		VOLUME		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
		NOTE: volumes greater than 1,000 L shall be shown in m^3		
		MASS		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2,000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
		TEMPERATURE (exact degrees)		
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
		ILLUMINATION		
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m^2	cd/m^2
		FORCE and PRESSURE or STRESS		
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS FROM SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
		LENGTH		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		AREA		
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
		VOLUME		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3
		MASS		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2,000 lb)	T
		TEMPERATURE (exact degrees)		
°C	Celsius	1.8C+32	Fahrenheit	°F
		ILLUMINATION		
lx	lux	0.0929	foot-candles	fc
cd/m^2	candela/m2	0.2919	foot-Lamberts	fl
		FORCE and PRESSURE or STRESS		
N	newtons	2.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380. (Revised March 2003)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATMS	Advanced Transportation Management System
CAP	Common Alerting Protocol
CMS	Changeable Message Signs
ConOps	Concept of Operations
DEM	Digital Elevation Map
ESS	Environmental Sensor Station
FHWA	Federal Highway Administration
GFS	Global Forecast System
GRIB2	NCEP and WMO's Gridded Binary File Format Version 2
IMRCP	Integrated Modeling for Road Condition Prediction
ITS	Intelligent Transportation Systems
LaDOTD	Louisiana Department of Transportation and Development
MDSS	Maintenance Decision Support System
METRo	Model of the Environment and Temperature of Roads
MLP	Machine Learning-based Prediction
MRMS	Multi Radar Multi Sensor
NDFD	National Digital Forecast Database
NHC	National Hurricane Center
NOAA	National Oceanic and Atmospheric Administration
NWM	National Water Model
NWS	National Weather Service
RAP	Rapid Refresh weather model
RTMA	Real-Time Mesoscale Analysis
SAD	System Architecture Description
TMC	Transportation Management Center
URL	Uniform Resource Locator
USDOT	United States Department of Transportation
WxDE	Weather Data Environment

CHAPTER 1. INTRODUCTION

BACKGROUND

Transportation agencies face many challenges when working toward effective transportation systems management and operations (TSMO). Weather, incidents, and events outside the knowledge and control of transportation agencies and travelers can reduce a free flow of traffic to a snarled mess in minutes or seconds. Whether the focus is on traffic incident management, work zones, active traffic management, or emergency evacuation routing, the intent of TSMO is to minimize and mitigate the impact of disruptions and to enable travelers to make better choices for safe and reliable travel.

The breadth of information available to support TSMO strategies fortunately has been expanding to meet the needs. Views of traffic conditions, road weather, incidents and work zones, and flooding and extreme events have been expanded by augmenting an agency's own data with other private and crowdsourced data. The views of conditions that might affect safety and mobility across the transportation network has grown both broader and more detailed. These views have, however, still been somewhat limited by their lack of integration and by being limited to current and past conditions. The next level of decision support for TSMO needs to be integrated across the technical disciplines and provide a view into likely future conditions.

The Federal Highway Administration (FHWA) Road Weather Management Program[1] began developing Integrated Modeling for Road Condition Prediction (IMRCP) in 2015 to provide data and decision support for these kinds of events. IMRCP is a tool that incorporates real-time and archived weather and traffic conditions data and results from forecast and probabilistic models to predict current and future road and travel conditions. It provides tools for what-if analyses; views of the data from archives; and views of the data from near-past, present, and future operational time horizons. IMRCP supports transportation system and emergency operations in decisionmaking and after-action reviews.

Previous phases of IMRCP developed the foundational traffic and weather data system components, including a machine learning-based traffic model. Previous phases also deployed the system at the scale of a metropolitan area, operated the system for five winter seasons, evaluated the operational results, and updated the system documentation. IMRCP phase 5 improved and demonstrated IMRCP capabilities as an agency-deployable system with multiple contiguous States managing evacuations and responses to adverse weather conditions. IMRCP's improvements have included traffic forecasting for tropical storm conditions, a dashboard for user-configurable event alerts, data management and processing speed to accommodate multiple large road networks, and data sharing across deployments. IMRCP capabilities have been demonstrated in a regional deployment over Louisiana, Mississippi, and Alabama to support analysis and management of evacuation, response, and recovery from adverse weather

[1] "Road Weather Management Program," FHWA, USDOT, last accessed April 3, 2025, https://ops.fhwa.dot.gov/weather/.

conditions. The improved IMRCP capabilities, updated technical documentation, and training materials are packaged for future deployments in an open-source repository.[2]

PURPOSE

The objective of Phase 5 of IMRCP is to improve and demonstrate IMRCP capabilities as an agency-deployable system with multiple contiguous States managing evacuations and responses to adverse weather conditions. IMRCP improvements are based on the platform and gaps identified in Phase 4. Additional improvement priorities have been developed from stakeholder engagement and requirements to support an agency-deployable system. IMRCP capabilities are demonstrated in a regional deployment over multiple contiguous States—specifically Louisiana, Mississippi, and Alabama—to support analysis and management of evacuation, response, and recovery from adverse weather conditions as identified in the Congressionally-authorized Emergency Planning Transportation Data Initiative. The improved IMRCP capabilities, updated administrative guidance, technical documentation, and training materials are to be packaged for future agency deployments and placed in an open-source repository.

The purpose of this Installation and Administration Guide is to describe the installation, configuration, operations, and administration of the IMRCP system. The system, as packaged, includes a default configuration for demonstration deployment. While the system will install and run in this configuration, localization of some data sources, traffic data collectors, and the road network model will be needed for application in other geographic contexts. The localization may require development of new software components specific to the applicable traffic data sources. Configuration settings and modules for which this may be the case are noted as such in this document.

DOCUMENT OVERVIEW

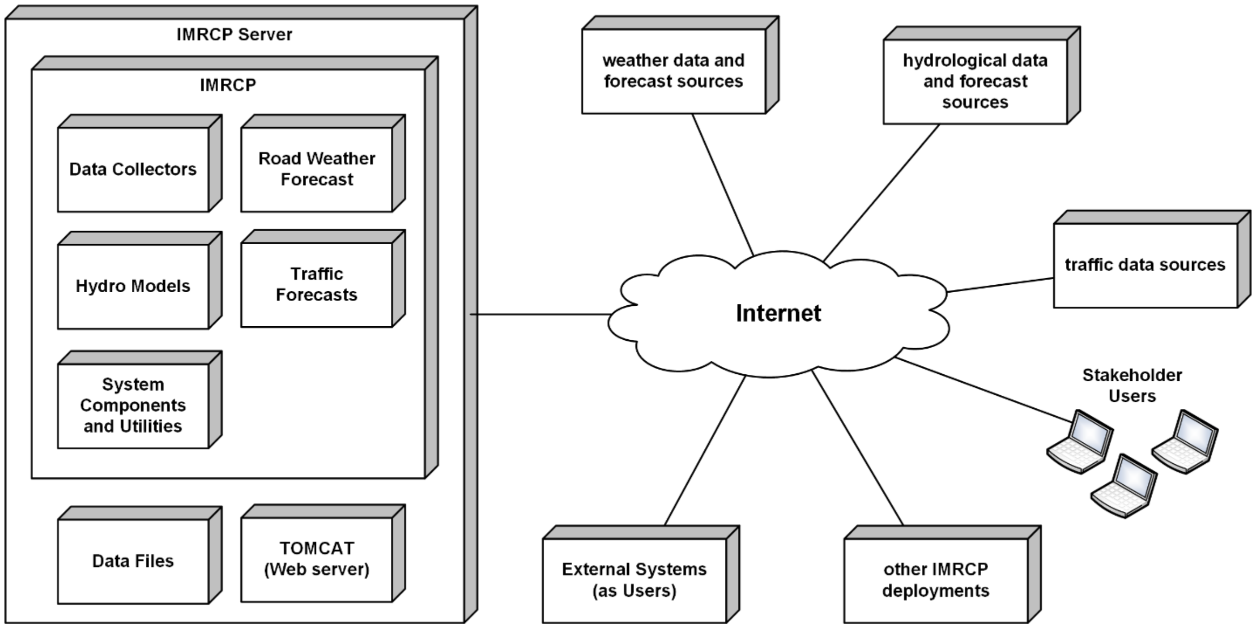
This document provides guidance on the administration and installation of the IMRCP system:

- Chapter 2 describes the installation of IMRCP components.
- Chapter 3 describes the IMRCP system configuration .
- Chapter 4 outlines the continuous operation requirements for the IMRCP system.
- Chapter 5 details the maintenance required for the IMRCP system.
- Chapter 6 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.

[2] "OSADP/IMRCP," IMRCP GitHub repository, last accessed April 3, 2025, https://github.com/OSADP/IMRCP.

CHAPTER 2. SERVER ENVIRONMENT AND INSTALLATION

The Integrated Modeling for Road Condition Prediction (IMRCP) system package contains all of the IMRCP system components. Figure 1, developed as part of the IMRCP System Architecture Description,[3] shows the deployment of the IMRCP system.



Source: FHWA, 2025.

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

INTEGRATED MODELING FOR ROAD CONDITION PREDICTION SERVER

As described in the *IMRCP System Design Description*,[4] the IMRCP server is used to provision all of the core system components.

The IMRCP System software uses many different data sets that include large weather files both in terms of number and size of each file, and performs complex calculations on those large data sets. This is reflected in the basic hardware configuration in number of processor cores, amount of main memory, and storage capacity.

CPU: 16 core/32 threads 2.0 GHz (minimum)

RAM: 128 GB (minimum) 256 GB (recommended)

DISK: 10 TB SSD RAID6 (minimum) 10 TB SSD RAIDZ2 (recommended)

[3] Leidos, *Integrated Modeling for Road Condition Prediction System Architecture Description* (unpublished working paper developed under Contract 693JJ322A00005, Integrated Modeling for Road Condition Prediction (IMRCP)–Phase 5, March 21, 2025.

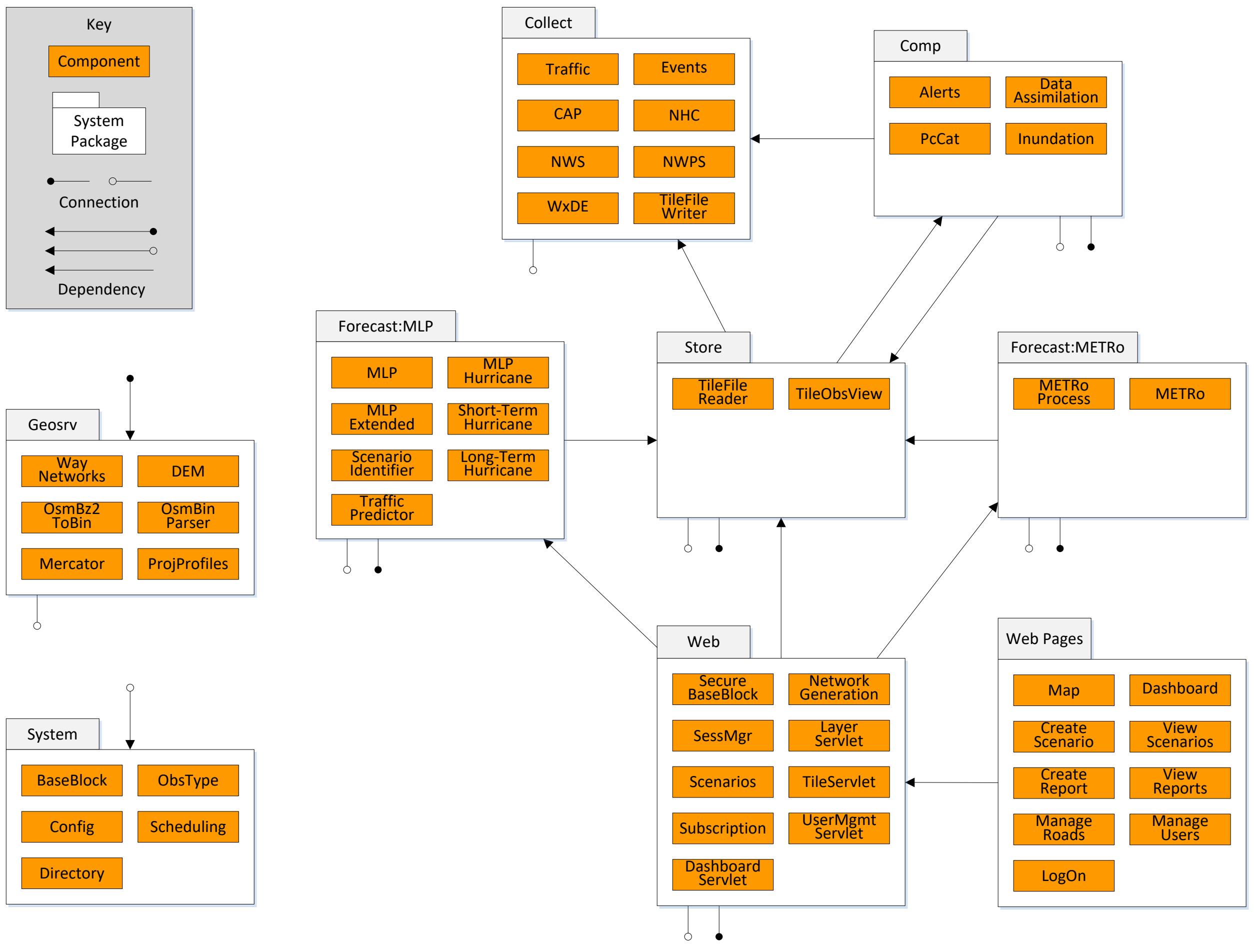
[4] Leidos, *Integrated Modeling for Road Condition Prediction System Design Description* (unpublished working paper developed under Contract 693JJ322A00005, Integrated Modeling for Road Condition Prediction (IMRCP)–Phase 5, January 17, 2025.

BACKUP: similar to or larger than primary storage

The server hardware should be deployed in the hosting organization's environment following the organization's established processes and procedures. Selection of a particular Linux operating system is not a critical decision. The IMRCP System has operated reliably using Debian. Debian-related distributions such as Fedora and Ubuntu are fine choices. Red Hat is used across many organizations and FreeBSD is also a good option if default support for the Zettabyte File System (ZFS) is desired.

INTEGRATED MODELING FOR ROAD CONDITION PREDICTION COMPONENTS

As shown in Figure 1, the IMRCP server itself contains three server packages to support the functional components. These server packages are 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.



Source: FHWA, 2025.

Figure 2. Integrated Modeling for Road Condition Prediction components.

INSTALLATION GUIDE

The IMRCP System software is intended to be deployed on a Linux operating system. Personnel involved in the deployment should possess the following skills and understand the listed concepts.

Table 1. Installation Skills and Concepts

Skills	Concepts
Install Linux operating system (Debian/RHEL)	Remote terminal access (SSH/SCP)
Manage Linux users (useradd/adduser/sudo)	Domain name (DNS/FQDN)
Linux file systems (mkdir/chmod/chown)	Security certificate (TLS)
Editing configuration files (nano/vim)	Network address (IPv4/IPv6)
Linux package manager (apt/rpm)	Firewall (rules/port forwarding)
File download utilities (wget/curl)	App frameworks (NGINX/Tomcat/Python)
File distribution utilities (tar/gzip)	HTML/CSS (optional)

Installing the IMRCP system on a server involves multiple dependent steps. Personnel may want to completely read through the installation steps to plan for some aspects of the deployment and Linux operating system installation such as domain name, username format, initial username, storage volume layout, network filing systems, and SMTP server.

1. Install a Linux operating system on the server. Update the package manager, and upgrade the operating system. For Debian, the commands are apt update, and apt upgrade. Ensure the desired data storage directories are mounted and accessible, updating /etc/fstab as needed.

2. Verify utility applications are installed. Use the Linux package manager (apt/rpm) to install utilities that may be missing. Check that nano, SSH/SSHD, sudo, wget, and xz-utils are available.

3. Set system time zone to UTC. Configure server network address, and update firewall rules to forward the public network address HTTPS port 443 to the server internal network address.

4. Install NGINX web server using Linux package manager (apt/rpm). For Debian, the command is apt install nginx. Verify the /etc/nginx/sites-available and /etc/nginx/sites-enabled directories exist, and create them as needed. Examine the /etc/nginx/nginx.conf file to verify that the "gzip on;" entry is in the Gzip Settings section, and that the "include /etc/nginx/sites-enabled/*;" entry is in the Virtual Hosts section.

5. Verify Python version, using the python3 --version command. The minimum recommended Python version is 3.11.x. If Python 3 is not installed, first use the operating system's package manager to install it, such as apt install python3. If this version is not 3.11.x then follow the manual installation instructions from https:// python.org.

6. Configure and set up the Python installation. Create a virtual environment for Python and verify Python library dependencies exist by attempting to install them. If 3.11.x was manually installed, substitute python3.11 for the python3 command. Existing libraries will be skipped automatically. Here are the typical commands to do so:

    - apt-get install python3-venv
    - python3 –m venv /opt/imrcp-python
    - source /opt/imrcp-python/bin/activate
    - python3 -m pip install pandas
    - python3 -m pip install scikit-learn==1.2.1
    - python3 -m pip install torch
    - python3 -m pip install statsmodels
    - python3 -m pip install pmdarima
    - deactivate

7. Download the IMRCP deployment tarball "IMRCPv05.12.tar.gz" from GitHub (https://github.com/OSADP/IMRCP/releases/download/IMRCPv05.12/IMRCPv05.12.tar.gz) and unpack it into the /opt directory.

8. Determine the server's public network address, register the domain name, and follow your organization's process to obtain a security certificate. Save a copy of the key, and certificate files in the /opt/imrcp05.12 directory as imrcp.key, and imrcp.crt.

9. Sign-up for a Mapbox developer account (https://account.mapbox.com/auth/signup). When the account is created, a default unrestricted default public access token is also made. Do not use the default public access token. Instead, create a new token with the default selections. Name it "imrcp", and also restrict it to the https:// <fully-qualified-domain-name> URL.

10. Edit the /opt/imrcp05.12/webapps/IMRCP/ROOT/script/map-common.js file, and replace <mapbox-access-token> with the deployment domain's mapbox access token.

11. Edit the /opt/imrcp05.12/webapps/IMRCP/ROOT/mapbox/sourcelayers.json file. Find and replace all instances of <fully-qualified-domain-name> with the domain name.

12. Edit the JSON configuration file /opt/imrcp05.12/webapps/IMRCP/config/imrcp_system_BaseBlock.json and set the path for the data, archive, and temporary storage locations. These locations should have read/write privilege by www-data, and can be the same directory. It is best NOT to place them within /opt/imrcp05.12 for smoother upgrades.

13. Ensure the directory "networks" exists in the configured data storage location. Move nine files from the /opt/imrcp05.12/webapps/IMRCP/pymrcp directory to the networks directory: decision_tree.pickle, mlp_python_data.pkl, and *.pth (oneshot_model.pth and online_model_1hour.pth through online_model_6hour.pth).

14. Edit the JSON configuration file /opt/imrcp05.12/webapps/IMRCP/config/imrcp_system_Emails.json and set the email, pw, smtphost, and domain parameters; as well as the support email address where bug reports are sent. This configuration is for an existing SMTP server needed to enable the reset password function. The default SMTP submission network port parameter “smtpport” is 587, and the “from” parameter defaults to “donotreply@<FQDN>”, but can be uncommented and set to different values if the SMTP server port is different or only allows the same email address as the email login account name.

15. Edit the /opt/imrcp05.12/nginx-imrcp file and set the server name to the domain name. Create a symbolic link for the nginx-imrcp file in the /etc/nginx/sites-enabled directory. Restart NGINX using system control commands, typically systemctl restart nginx.

16. Run the /opt/imrcp05.12/start_imrcp.sh script, Tomcat running under the www-data account. Inspect the /opt/imrcp05.12/logs/catalina.out file to confirm that the Tomcat application server is running.

CHAPTER 3. MODEL LOCALIZATION AND CONFIGURATION

INTRODUCTION TO CONFIGURATION

IMRCP was designed to be flexible in implementation and contains many configuration options for deployers. However, in most cases the default configuration is sufficient and should not be changed to ensure stable operation of the system. The following sections will describe the configuration files and models used in IMRCP and highlight configuration items that could be changed for a typical deployment.

CONFIGURATION FILES

Component specific files

Configuration for IMRCP components are described by JSON files within the configuration directory which is defined in the web.xml file. These JSON files can be named using one of three conventions and regardless of the chosen convention must end with the file extension ".json".

The first convention is a fully qualified Java name, with periods replaced by underscores, representing a class name that extends the imrcp.system.BaseBlock class. For example, the configuration file name for the imrcp.collector.Traffic class would be imrcp_collector_Traffic.json.

The second convention is a named instance of an imrcp.system.BaseBlock. Named instances are defined in the imrcp_system_Directory.json configuration file.

The third convention is in the format "rangerules_<obstype id>.json" where <obstype id> refers to a system observation type id which is a base-36 case-insensitive 6-character string, for example the obstype id for air temperature is "TAIR".

Configuration for named instances follow their inheritance chain, starting with imrcp.system.BaseBlock continuing to each derived class in order, for example the component named "Radar" will search for configuration items contained in the following files: imrcp_system_BaseBlock.json, imrcp_collect_TileFileWriter.json, imrcp_collect_Collector.json, imrcp_collect_NWS.json, and Radar.json.

If a configuration item is found in multiple files, the value found in the file furthest along the inheritance chain will be used. Continuing the example for the component named "Radar", if the configuration item "datadir" was contained in imrcp_system_BaseBlock.json, imrcp_collect_Collector.json, and Radar.json, the value in Radar.json would be used by the system. Descriptions of component configuration can be found in Appendix B.

Web.xml

This file contains configuration parameters for the IMRCP web application. Its path is application root/WEB-INF/web.xml. It contains the components that Tomcat is responsible for starting which includes the Directory, Session Manager, and servlets that respond to client requests.

log4j2.properties

This configuration file contains the configuration for the logger log4j2. Documentation for the format of this file can be found (as of August 14, 2025) at: <https://logging.apache.org/log4j/2.x/manual/configuration.html>.

TRAFFIC DATA COLLECTION CONFIGURATION

Traffic data will be localized for each deployment and therefore must be setup and configured by the system administrator. A typical configuration includes an external data adapter and a system Traffic collector. The data adapter ingests data from a traffic data provider like an advanced transportation management system, INRIX, or HERE. It must process the data into the IMRCP Traffic Speed Interface Specification and create files with the correctly formatted time-based file names (refer to Appendix E of the SDD). To configure the system data collector, an entry must be added to the “classes” and “resources” arrays in the “imrcp_system_Directory.json” file to instruct the system to instantiate the Traffic collector and a Resource Record, and a new JSON configuration file must be created for the Traffic collector. The following examples and explanations will assist in this configuration.

imrcp_system_Directory.json

To instantiate and register a Traffic Data Collection component, add two String values in the “classes” array. The first String will be “imrcp.collect.Traffic” which is the class name of Traffic Data Collector class. The second String is the instance name. It is best to choose a simple descriptive name. To avoid errors in the “classes” array, append these two Strings at the end of the array. To add a Resource Record to the system for the Collector, an object must be added to the “resources” array. To ensure there are no errors in key names, it is suggested to copy and paste from the example below and edit the property values. Not all values need to be changed and detailed information on each property can be found in the IMRCP SDD. First choose a contributor identifier which is a case-insensitive 6-character string that gets converted into an integer using base-36 for the “contrib” property. Use that contributor identifier to replace all of the instances of “kdot” in the “archiveff” and “tileff” properties. The “range”, “freq”, “archivesearchoffset”, “archivefreq”, and “delay” properties all relate to the timing for collection and processing of the data and are in milliseconds. IMRCP stores traffic data as observations valid for 5 minutes. If data sources have higher temporal resolution than this, the mean value of all observations in a 5-minute window is saved. In the example, data for this collector are saved in files that are valid for 5 minutes (“range”) and produced every 5 minutes (“freq”). Data is available 7 minutes after collection (“delay”). Source data is produced every 1 minute (“archivefreq”) so 5 minutes’ worth of source data makes up 1 system file (“archivesearchoffset”). If speed source data is in miles per hour, change the “srcunits” property to “mph”. Change the “writer” property to the instance name of the Collector that was used in the “classes” array.

```
{
 “classes”: [“imrcp.collect.Traffic”, “KansasTraffic”, … ],
“resources”:
[
 {
  "contrib": "KDOT",
  "archiveff":
"kdot/%ryM/%ryMd/kdot_speed_%syMdHm_%eyMdHm_%ryMdHm.csv.gz",
  "tileff": "%S/kdot/%ry/%ryMd/kdot_%S_%syMdHm_%eyMdHm_%ryMdHm.bin",
  "range": 300000,
  "freq": 300000,
  "delay": -420000,,
  “archivesearchoffset”: 300000
  “archivefreq”: 60000
  "zoom": 9,
  "obsid": ["SPDLNK"],
  "srcunits": ["kph"],
  "round": [1.0],
  "classes": [""],
  "valuetype": [1],
  "preference": [0],
  "writer": "KansasTraffic"
 }, …
 ]
}
```

KansasTraffic.json

A new file needs to be created in the configuration directory, *imrcpNN.MM*/webapps/IMRCP/config/. To avoid errors in property names, it is suggested to copy and paste from the example below and edit the property values. Properties “period”, the period of execution, and “offset”, the midnight offset” tell the system when to process data for this Collector and are in seconds. In this example the Collector executes every 5 minutes with a 2 and a half minute offset from midnight, meaning the Collector would execute starting a 12:02:30am and continue doing so every 5 minutes. The “contrib” property must match the contributor identifier in it corresponding Resource Record detailed in the section above. “threads” tells the system to allocate 2 threads to process data files. “tolmul” is a tolerance multiplier used in the algorithm that snaps traffic observations to system roadway segments. The tolerance is based off of the resolution a Spherical Mercator map at the configured zoom level. A larger tolerance multiplier increases the search space for nearby roadway segments.

```
{
“period”: 300,
“offset”: 150,
“contrib”: “KDOT”,
“threads”: 2,
“tolmul”: 20
}
```

EVENT DATA COLLECTION CONFIGURATION

Like traffic data, event (incident, roadwork, etc.) data will be localized for each deployment and therefore must be setup and configured by the system administrator. A typical configuration includes an external data adapter and a system Event collector. The data adapter injests event data from an external source and must process the data into the IMRCP Event Interface Specification and create files with the correctly formatted time-based file names (refer to Appendix E of the SDD). To configure the system data collector, an entry must be added to the "classes" and "resources" arrays in the "imrcp_system_Directory.json" file to instruct the system to instantiate the Event collector and a Resource Record, and a new JSON configuration file must be created for the Event collector. The following examples and explanations will assist in this configuration.

imrcp_system_Directory.json

To instantiate and register an Event Data Collection component, add two String values in the "classes" array. The first String will be "imrcp.collect.Events" which is the class name of Event Data Collector class. The second String is the instance name. It is best to choose a simple descriptive name. To avoid errors in the "classes" array, append these two Strings at the end of the array. To add a Resource Record to the system for the Collector, an object must be added to the "resources" array. To ensure there are no errors in key names, it is suggested to copy and paste from the example below and edit the property values. Not all values need to be changed and detailed information on each property can be found in the IMRCP SDD. First choose a contributor identifier which is a case-insensitive 6-character string that gets converted into an integer using base-36 for the "contrib" property. Use that contributor identifier to replace all of the instances of "kdot" in the "archiveff" and "tileff" properties. The "range", "freq", and "delay" properties all relate to the timing for collection and processing of the data and are in milliseconds. In the example, data for this collector are saved in files that are valid for 1 week ("range") and produced every 5 minutes ("freq"). Data is available upon collection ("delay"). Change the "writer" property to the instance name of the Collector that was used in the "classes" array.

```
{
 "classes": ["imrcp.collect.Events", "KansasEvents", … ],
 "resources":
 [
  {
   "contrib": "KDOT",
   "archiveff":
"kdot_events/%ryM/%ryMd/kdot_event_%syMdHm_%eyMdHm_%ryMdHm.csv.gz",
   "tileff": "%S/kdot/%ry/%ryMd/kdot_%S_%syMdHm_%eyMdHm_%ryMdHm.bin",
   "range": 604800000,
   "freq": 300000,
   "zoom": 9,
   "obsid": ["EVT", "VARIES"],
   "srcunits": ["", ""],
   "round": [1.0, 1.0],
   "classes": ["", ""],
   "valuetype": [5, 5],
   "preference": [0, 0],
   "delay": 0,
   "writer": "KansasEvents"
  }, …
 ]
}
```

KansasEvents.json

A new file needs to be created in the configuration directory, *imrcpNN.MM*/webapps/IMRCP/config/. To avoid errors in property names, it is suggested to copy and paste from the example below and edit the property values. Properties “period”, the period of execution, and “offset”, the midnight offset” tell the system when to process data for this Collector and are in seconds. In this example the Collector executes every 5 minutes with a 15 second offset from midnight, meaning the Collector would execute starting a 12:00:15am and continue doing so every 5 minutes. The “contrib” property must match the contributor identifier in it corresponding Resource Record detailed in the section above. “threads” tells the system to allocate 2 threads to process data files. “strings” tells the system that for each observation 5 string values will be saved, this is constant for all Event collectors and should not be changed.

```
{
 "offset": 15,
 "period": 300,
 "contrib": "kdot",
 "threads": 2,
 "strings": 5
}
```

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.

Configuration items for NWS collectors describe how and from where the system should download files. Changing these configuration items could cause data to not be collected correctly. However, NWS data endpoints are subject to change on an infrequent basis; refer to Appendix B for details on which parameters would need to be modified in this case.

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. Refer to Appendix B for configuration items that are deployment specific.

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.[5] METRo computes the road temperature, subsurface temperature, pavement state, liquid depth, and snow/ice depth. Changing the default configuration for METRo may cause abnormal behavior and therefore should not be modified.

METRo uses pavement and subsurface temperature values derived from kriging algorithms based off of data collected from the Weather Data Environment (WxDE). The data endpoint and subscription Id are subject to change on an infrequent basis. Refer to Appendix B for details on which parameters would need to be modified in this case.

[5] 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>.

CHAPTER 4. 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. The distribution package includes convenience start and stop scripts.

- *imrcpNN.MM*/start_imrcp.sh
- *imrcpNN.MM*/stop_imrcp.sh

CHAPTER 5. 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 using the Manage Users webpage. Administrators can use the default account to setup users upon installation of the software. Open an internet browser and navigate to the domain name URL. Login in with the default credentials (user = imrcpadmin, password = password). Left-click on “Manage Users” menu option. Add a new user and enter the email address of someone who will administer IMRCP. Set the User Group to imrcp-admin, set a new password, and click “Save”. Log off of the IMRCP system. Again open an internet browser and navigate to the domain name URL. Login as the newly created system administrator using the email address and password, then left-click on “Manage Users”. Double-click on the entry for imrcpadmin, change the User Group to imrcp-user, select “Disabled”, and click “Save”. Other users and administrators can be added at any time.

CHAPTER 6. REFERENCES

ISO/IEC/IEEE. *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*. ISO/IEC/IEEE 29148:2011.

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

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

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

Leidos, *Integrated Modeling for Road Condition Prediction Phase 5: System Architecture Description* (unpublished working paper developed under Contract 693JJ322A00005, Integrated Modeling for Road Condition Prediction (IMRCP)–Phase 5, January 17, 2025

Leidos, *Integrated Modeling for Road Condition Prediction System Design Description* (unpublished working paper developed under Contract 693JJ322A00005, Integrated Modeling for Road Condition Prediction (IMRCP)–Phase 5, January 17, 2025

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. CONFIGURATION DESCRIPTIONS

IMRCP_SYSTEM_DIRECTORY.JSON

The Directory component contains configuration items that greatly impact how the application operates.

- classes
    - String array that instructs the system what components to instantiate and register.
    - Written in pairs, where the first string is the fully qualified java name of the class to instantiate and the second string is the name of the instance, which must be unique.
    - Placing a "#" in front of a class name cause the system to ignore that entry.
    - A "@" needs to be placed in front of any component configured to be started by the application webserver.
    - Example: "classes": ["imrcp.geosrv.WayNetworks", "WayNetworks", "imrcp.collect.NWS", "RAP", "imrcp.collect.NWS", "RTMA", "#imrcp.collect.NWS", "OldRTMA", "@imrcp.web.tiles.TileServlet", "TileServlet"] would instantiate and register three components with names "WayNetworks", "RAP", and "RTMA", ignore the component named "OldRTMA" and register the component "TileServlet" which was instantiated by the application webserver.
- threads
    - Integer that specifies how many threads can be used by the application's thread pool.
- resources
    - Array of objects that define the Resource Records in the system. A detailed explanation of Resource Records can be found in the SDD in the Directory Component section.
    - Resource Records will need to be created for any additional data source used in a deployment. In most cases this would include Resource Records for Traffic and Event collectors specific to the deployment.
    - Resource Records for default data sources should not be modified.

IMRCP_SYSTEM_BASEBLOCK.JSON

The BaseBlock configuration contains three absolute paths that define default locations for files: “archpath”, “datapath”, and “temppath”. “archpath” is where original data files are archived. “datapath” is where processed data files will be saved. “temppath” is where temporary files created by the system will be saved.

TILEFILEWRITERS

Configuration for TileFileWriters include parameters to specify how often to process data, where to download data if necessary, and the number of threads allocated to the component to process data. The following files contain the configuration for the default TileFileWriters and should not be changed to ensure normal operations of the system.

- AdcircEast.json
- AdcircWest.json
- AHPSForecast.json
- AHPSObservation.json
- GFS.json
- imrcp_collect_AHPS.json
- imrcp_collect_CAP.json
- imrcp_collect_NWM.json
- imrcp_collect_WxDESub.json
- NDFDQpf.json
- NDFDSky.json
- NDFDTd.json
- NDFDTemp.json
- NDFDWspd.json
- NHC.json
- Radar.json
- RadarPrecip.json
- RAP.json
- RTMA.json
- PcCat.json
- TileFileWriterTYPPC.json
- TPVTDA.json
- TSSRFDA.json
- Inundation.json
- Metro.json
- imrcp_comp_DataAssimilation.json
- imrcp_forecast_mdss_Metro.json

An exception to this would be changing the parameters for NWS collectors if the data endpoint changes. Depending on the change the “src”, “url”, “urlext”, “pattern”, “start”, and “end” parameters might need to change. “src” is a string used to format time dependent file names.

"url" is the URL of the base directory for the NWS product that contains data directories/files. "urlext" is a string used to format time dependent directories that is appended to the "url" string to access specific day's data files. "pattern" is a regular expression string used to match valid file name patterns in the directory. "start" and "end" are strings used to search for the start and end of a file name.

Another exception would be changing the parameters for the WxDESub collector if the data endpoint or subscription changes. "url" is the URL of the WxDE for downloading subscription data. "uuid" is the identifier provided by WxDE for the subscription.

MACHINE LEARNING PREDICTION COMPONENTS

Configuration parameters for MLP Components include "script", "python", and "processes". "script" defines the absolute path of the python script used to execute MLP functions. If the installation guide was followed, the default value should not be modified. "python" defines the command used to call python code which allows users to decide which version and virtual environment of python to use. "processes" defines the number of processes that can be spawned simultaneously to run the traffic model. The following list contains the configuration files for different MLP components.

- imrcp_forecast_mlp_MLP.json
- imrcp_forecast_mlp_MLPHurricane.json
- MLPExtended.json

IMRCP_WEB_SECUREBASEBLOCK.JSON

Configuration for SecureBaseBlocks include parameters for specifying what users groups can use different APIs, file name formats, and how the system parses user requests for the different components. The default configurations are sufficient and should not be modified. The following list contains the configuration files of the different SecureBaseBlocks.

- DashboardServlet.json
- imrcp_web_layers_AreaLayerServlet.json
- imrcp_web_layers_RoadLayerServlet.json
- imrcp_web_tiles_TileServlet.json
- imrcp_web_Subscriptions.json
- NetworkGeneration.json
- Scenarios.json
- UserManagementServlet.json
- UserSettingsServlet.json

RANGERULES

Rangerules define ranges of values for different observation types that instruct the system how to interpret values when parsing data files and when displaying observations on the map. The following files contain the default rangerules and should not be modified.

- rangerules_DPHLIQ.json
- rangerules_DPHLNK.json
- rangerules_DPHSN.json
- rangerules_EVT.json
- rangerules_GSTWND.json
- rangerules_PCCAT.json
- rangerules_RDR0.json
- rangerules_RTEPC.json
- rangerules_SPDLNK.json
- rangerules_SPDWND.json
- rangerules_SSCST.json
- rangerules_STG.json
- rangerules_STPVT.json
- rangerules_TAIR.json
- rangerules_TPVT.json
- rangerules_TRFLNK.json
- rangerules_TSSRF.json
- rangerules_VIS.json

IMRCP_SYSTEM_EMAILS.JSON

The Emails component configuration determines how the application sends emails and must be modified per deployment for emails to be successfully sent from the system for password reset and bug notifications. The email, pw, smtphost, domain, and support parameters need to be set. “email” and “pw” are used as login credentials for the system to send emails. “smtphost” is the domain name for the SMTP host server being used. “domain” is the fully qualified domain name used for the deployment of IMRCP. “support” is the email address that will receive any bug reports submitted by users. Optional parameters include “from” (the email address password reset details are sent from) and “smtpport” which default to “donotreply@<FQDN>” and 587, respectively.

IMRCP_SYSTEM_UNITS.JSON

The Units component configuration contains the absolute path to the unit conversion file. If the installation guide is followed this should not need to be modified.

WAYNETWORKS.JSON

The WayNetworks component configuration contains parameters for file name formats used by the component. The default values are sufficient and do not need to be modified.

OBSTYPE.JSON

The ObsType component configuration contains parameters for precipitation thresholds to determine precipitation categories. The default values are sufficient and do not need to be modified.

U.S. Department of Transportation
Federal Highway Administration
Office of Operations
1200 New Jersey Avenue, SE
Washington, DC 20590

Office of Operations website
https://ops.fhwa.dot.gov