Draft Motor Vehicle Emission Simulator (MOVES) 2009

Software Design and Reference Manual

United States
Environmental Protection
Agency

Draft Motor Vehicle Emission Simulator (MOVES) 2009

Software Design and Reference Manual

Assessment and Standards Division
Office of Transportation and Air Quality
U.S. Environmental Protection Agency

EPA United States Environmental Protection Agency

EPA-420-B-09-007
March 2009

A Note about the Capitalization and Naming Conventions Used in this Document

Because object orientation is central to the design of the MOVES software, this document often refers to classes and objects. The class names used in MOVES are often formed from several English words which are run together without spaces, e.g. “EmissionCalculator”, “RunSpecEditor”, etc. MOVES follows the widely-used Java naming convention of capitalizing each word in these run-together names, and this convention is also used in this document.

Because databases are also central to MOVES, this document includes many references to databases, tables, and fields. It is our intention that database and table names follow the same naming convention as classes in MOVES. The names of fields within these tables also use this scheme, except that they begin with a lower case letter (unless they begin with an acronym). E.g. there is a “MOVESDefault” database which contains an “EmissionRate” table and this table contains a field named “meanBaseRate”. We’ve applied the same naming convention to file and directory names as well because they are somewhat analogous to tables and databases.

Acronyms, such as “VMT” (from “vehicle miles traveled”) or “AC” (from “air conditioning”) are capitalized in all contexts, even when they begin a field name, even though this is not standard programming practice. Thus there is a field named “ACPenetrationFraction”. A table of acronyms used in this document is contained in Appendix A.

There are undoubtedly instances where this document fails to fully comply with these conventions. In some contexts it seems natural to use ordinary English, e.g. “emission rate”, interchangeably with a class name, e.g. “EmissionRate”, and we have not attempted to be highly rigorous in this regard. The Windows operating system is not case sensitive and MySQL is also rather forgiving in this respect. We hope our readers will be as well.

Table of Contents

1. Introduction

The Draft 2009 Motor Vehicle Emission Simulator (DRAFT MOVES2009) is released to the general public mainly for users' review and comments. A follow-up final version of MOVES, scheduled to be released in late 2009, will include modifications based on users' feedback and EPA's planned enhancement. DRAFT MOVES2009, which is the third of the current series of the MOVES "implementations", includes several new features, e.g., user data importers and air toxic. Although this draft model contains more realistic estimates of pollutant emissions than previous versions, it still holds some placeholder data, for example the emission rates for motorcycles were set to 0 (zero), which serve as placeholders to make users aware that the model includes motorcycles, while numeric estimates are not currently available. In addition, the GREET model interface that was incorporated in earlier versions has been disabled in this version because it is no longer operational. EPA hopes to restore this GREET functionality in a future version of MOVES.

A brief description of the two previous versions of MOVES before Draft MOVES2009 is as follows.

- MOVES2004 (released in 2004): the first version of MOVES that can be used to estimate and project national inventories at the county level for energy consumption, N_2O, and CH_4 from highway vehicles.
- MOVES-HVI (released in 2007): a demonstration version of MOVES that is the Highway Vehicle Implementation of EPA's Motor Vehicle Emission Simulator. The MOVES-HVI retains much of the functionality of MOVES2004 with a few significant updates as it is only intended to demonstrate the significant features added to MOVES to estimate criteria pollutant emissions (gaseous hydrocarbons, carbon monoxide, oxides of nitrogen and particulate matter) from highway vehicles. It is suitable only for demonstration purposes; none of its numerical value results should be considered to be realistic.

Future implementations of MOVES are planned to operate at smaller scales, estimate non-highway mobile source emissions, and estimate pollutants from additional mobile sources such as aircraft, locomotives, and commercial marine activity.

Two documents can be considered precursors to this one: The first is a report published in October 2002 entitled *Draft Design and Implementation Plan for MOVES*. This plan includes extensive background on the impetus for MOVES, an analysis of the "use cases" MOVES is intended to address, and the conceptual design for the model. The second was the *MOVES2004 Software Design Reference Manual* (SDRM) dated November, 2004. Both documents are available on the MOVES website (www.epa.gov/otaq/ngm.htm) and the reader is encouraged to consult them if additional background is desired.   The draft plan underwent formal peer review and public stakeholder review; the comments resulting from this process were summarized in Appendix I of the MOVES2004 SDRM. A preliminary version of the MOVES2004 SDRM was peer-reviewed, and the comments from this process were summarized in Appendix II of the released draft version. Because this document has the same purpose, structure, scope and formatting conventions relative to Draft MOVES2009 as the previous MOVES 2004 SDRM had relative to MOVES2004, EPA does not plan for it to undergo formal peer review. Comments from the public are welcome.

The overall purpose of this *Software Design and Reference Manual*, is, together with the Draft *MOVES2009 User Guide*, to answer questions pertaining to the MOVES software.   The User Guide is tailored to the beginning user and to getting started quickly using the model. It focuses on operation of the MOVES Graphical User Interface (GUI). This *Software Design and Reference Manual* intended to answer more substantive questions about the model software and document the calculations the model performs. It also provides more detail on configuring, installing, and running the MOVES program than the User Guide.

Chapter 2 identifies the major software and database components which make up DRAFT MOVES2009. At this general level the MOVE Software is considered to consist of 8 components.

Chapter 3 covers the hardware and system software required to run each major component.

Chapter 4 covers configuration of the MOVES software, which can be run on a single computer or a network of computers.

Chapter 5 covers MOVES software licensing.

Chapter 6 discusses the MOVES installation process in somewhat greater detail than the MOVES User Guide or the Readme file included in the installation package itself.

Chapter 7 provides a "processing overview" of MOVES.

Chapter 8 diagrams and discusses the flow of data and control between more detailed components of MOVES. At this level the DRAFT MOVES2009 software is divided into about 25 components.

Chapter 9 discusses the functional design concepts of MOVES.

Chapter 10 documents the functionality of most of the individual components of MOVES, their inputs, the calculations they perform, and the outputs they produce. This single chapter composes nearly half of the entire document.

Chapter 11 provides some top level documentation of the tables in the MOVES Input Database and contains more detailed documentation of the flow of data in MOVES by indicating which MOVES components read and write these tables in the MOVES Execution Database.

Chapter 12 documents the structure of the MOVES Output Database.

Other documents supplement the *DRAFT MOVES2009 User Guide* and the Draft *MOVES2009 Software Design and Reference Manual.* Those most closely related to the software are:

The MOVES Program Suite Distribution which includes a README file more briefly explaining the process of installing all MOVES-related components, and where to get assistance or report problems with the MOVES installation process.

Detailed documentation on the MOVES input database is included in a README directory within the database itself.

Technical documentation explaining the data sources and methods used to estimate the default fleet, activity, and emission data underlying DRAFT MOVES2009 is contained in technical reports separately downloadable from the EPA web site.

Documentation produced by Argonne Laboratories covering the GREET model is contained in the GREET directory within the model itself. Although the GREET interface has been disabled in MOVES, the documentation for GREET is provided in this version because EPA hopes to restore this functionality in a future version of MOVES.

2. MOVES Software Components

MOVES is written in Java™ and the MySQL relational database management system, a product of MySQL AB. Its principal user inputs and outputs, and several of its internal working storage locations, are MySQL databases. A "default" input database, covering 3222 counties of the United States and which supports model runs for calendar years 1990 and 1999 - 2050 is included with the model.

MOVES has a "master – worker" program architecture which enables multiple computers to work together on a single model run. A single computer can still be used to execute MOVES runs by installing both the master and worker components on the same computer.

Looking at this architecture in greater detail, the MOVES software application consists of eight components, each described briefly here.

MOVES Graphical User Interface (GUI) and Master Program: This is a Java program which manages the overall execution of a model run. Its MOVES GUI (also sometimes referred to as the run specification editor) may be used to create, save, load, and modify a run specification or RunSpec, and to initiate and monitor the status of a model run. A basic command line interface may be employed (in lieu of the GUI) by users (or by other computer programs) to execute the model without interacting with the MOVES GUI. If MOVES is installed on a computer network several model runs may be executed concurrently by different copies of the MOVES Master Program.

MOVES Worker Program: This is also a Java program. At least one executing copy of this program is needed to complete a MOVES run. It may execute on the same computer as the MOVES Master Program, or on other computer(s) having access to the SharedWork file directory.

Default input database, normally named "MOVESDefault": this MySQL database must reside on the same computer as the MOVES Master Program. A version of this database is included in the MOVES Installation Package. This **"MOVESDefault"** may be replaced by a database specifically named as **"MOVESDByyyymmdd"** in the

MOVES installation package, where yyyymmdd stands for a string of year, month and day.

SharedWork: This is a file directory or “folder” which is accessible to all executing copies of the MOVES Master program and the MOVES Worker program. It is not a MySQL database but simply a file directory or “folder” provided by the file services of a software operating system.

Optional user input databases: These MySQL databases are normally located on the same computer as the MOVES Master Program. They may contain any of the same tables that are in the default input database and are used to add or replace records as desired by the user.

The MOVESExecution database: This MySQL database is created by the MOVES Master Program. It is used for temporary working storage and does not interact directly with the user. It must be on the same computer as the master program.

MOVES output databases: MySQL databases are named by the user and produced by MOVES model program runs. While normally located on the same computer as the MOVES Master program, they could be located on any MySQL server accessible to it.

MOVESWorker database: This MySQL database is used as working storage by the MOVES Worker Program. It does not interact directly with the user. It is on the same computer as the MOVES Worker program.

3. Computer Hardware and System Software Requirements

The MOVES application software components require a computer hardware and software “platform” upon which to operate. The hardware platform can consist of a single computer system or a network of computers.

Computer(s) used to run either of the MOVES application programs should have at least 512MB of RAM. (Having additional memory is highly recommended, and is now relatively inexpensive). Execution run time performance is a constraint with MOVES so high speed dual-core processor(s), at least 1-2 GHz and preferably faster, are highly recommended. The MOVESDefault database distributed with MOVES requires approximately 1.3 GB of disk storage. MOVES Worker and Output databases are also often voluminous, so several gigabytes of disk space should be available on all machines used to run either MOVES program. Extensive users of MOVES will want to use the highest performance microcomputer systems that they can afford.

3.1. Details on JAVA Platform Requirements

The MOVES GUI/Master and the MOVES Worker are Java programs and for operation require a Java RunTime Environment (also sometimes referred to as the “Java Virtual Machine”). The Java Software Development Kit (SDK) includes the Run Time Environment. This version of MOVE uses version 1.4.2 of the Java SDK, produced by SUN Microsystems Inc. The MOVES Program Suite Distribution includes an installation package for this Java version, suitable for installation on WINDOWS NT, WINDOWS 2000, WINDOWS XP, and 32-bit Vista systems. MOVES does not operate successfully on 64-bit Vista or on versions of WINDOWS that predate WINDOWS 2000. Users should not attempt to operate either MOVES program with other versions. While Java is available for other software operating systems, such as LINUX, UNIX, etc. and porting MOVES to such software operating systems should not be difficult, EPA has not tested such configurations and is not prepared to support them. Sun’s main web site for information related to Java is http://java.sun.com/ .

Several extensions to Java are also required by MOVES and are included in the MOVES Installation Package. These include JUnit and JFCUnit, which facilitate software testing, and JavaHelp used to construct the on-line help facility in MOVES. The ANT software build utility is also included.

3.2. Details on MySQL Platform Requirements

The MySQL database management software has a client-server architecture. The MOVES GUI/Master, the MOVES Command Line interface/Master, and MOVES Worker programs function as MySQL clients and require access to MySQL server(s). Since both programs require a MySQL database to be located on the same computer (MOVESExecution and MOVESWorker), all computers that run any of these MOVES components must also operate a MySQL server. Additional computers operating MySQL servers can also be utilized; e.g., for MOVES Output databases.

This version of MOVES uses MySQL version 5.0.27. The DRAFT MOVES2009 Program Suite Distribution includes an installation package for this MySQL version, suitable for installation on WINDOWS NT, WINDOWS 2000, WINDOWS XP and 32-bit Vista systems. The version of DRAFT MOVES2009 will not operate with MySQL version 4.0.21 or earlier, neither will the future MOVES versions. Users are not recommended to attempt to operate MOVES program with MySQL versions of 5.1 or later since EPA has not tested MOVES on them. While MySQL is available for other software operating systems, such as LINUX, UNIX, etc. and porting MOVES to such software operating systems should not be difficult, EPA has not tested such configurations and is not prepared to support them. DRAFT MOVES2009 does not operate successfully with versions of WINDOWS which predate WINDOWS 2000.

The MySQL installation includes a command line MySQL client program. The MOVES Program Suite Distribution also includes an installation package for the MySQL Query Browser which is a GUI MySQL client program. Either of these MySQL client programs can be used to help construct MOVES input databases or to analyze the contents of MOVES output databases. Other data base management software, such as Microsoft ACCESS can also be used via an ODBC driver. Appendix B of the DRAFT MOVES2009 User Guide explains how this can be accomplished.

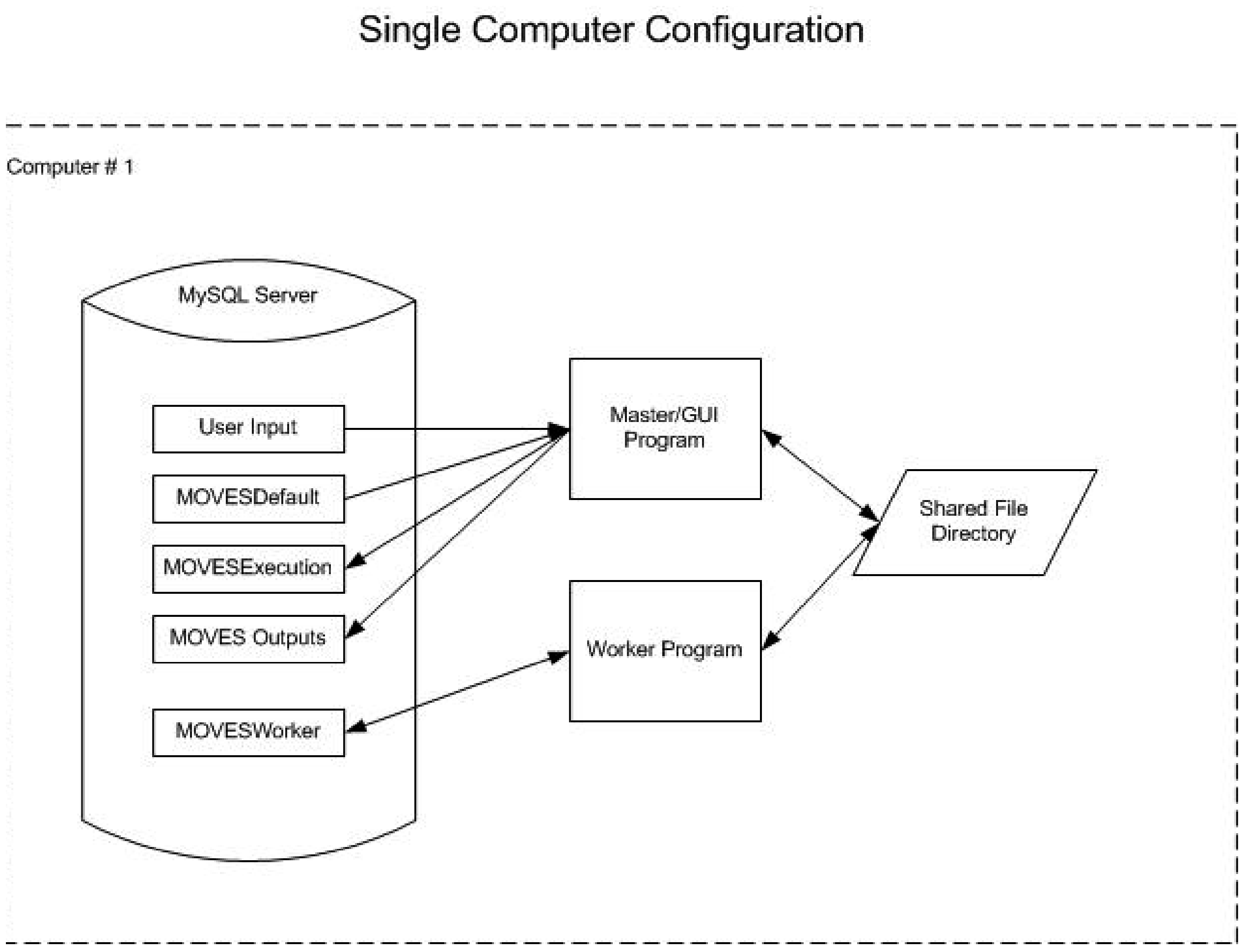
Additional information about MySQL is available at the MySQL web site (http://www.mysql.com/) operated by MySQL AB/Sun Microsystems.

3.3. Details on Shared File Directory Platform Requirements

The SharedWork file directory may be located on a computer that runs the MOVES Master Program or the MOVES Worker Program or on a separate “file server” computer. All that is necessary is that the MOVES GUI and Master program and at least one MOVES Worker program have access to this shared file directory with permission to create, modify and delete files. In the simplest case, diagrammed early in the next section, all MOVES Application Software components may reside on a single computer. In this case the SharedWork directory is simply on this computer’s local hard drive. All files created by MOVES in this directory are temporary, but because the files can be numerous and large, at least 3 GB of disk space should be available.

4. MOVES Computer Platform Configuration

All MOVES application components may be installed on a single computer system as shown in the following Figure 4-1.



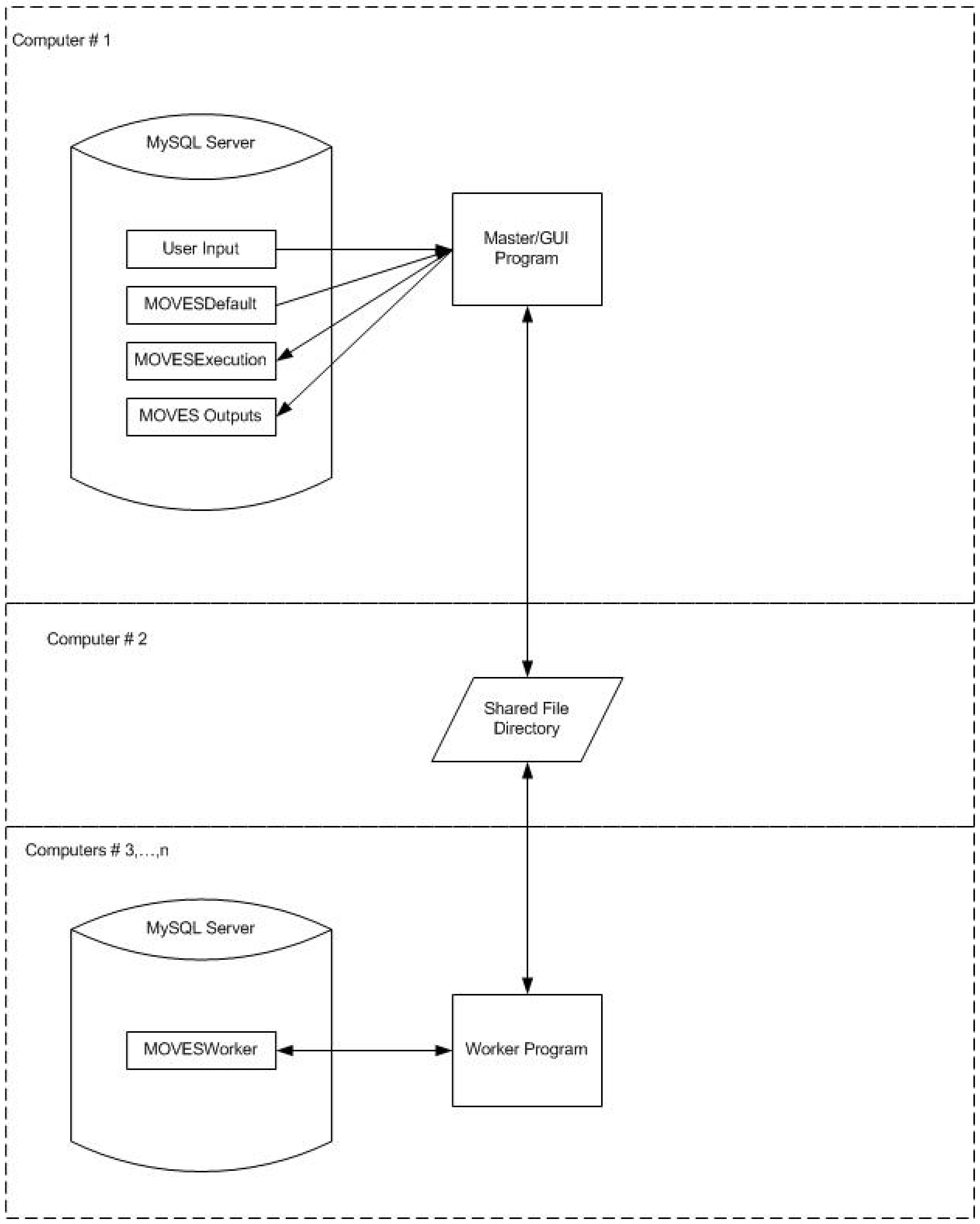
MOVES may also be configured so that several computers work together to execute model simulation runs. This can significantly improve execution time performance of large simulations. (Improvements diminish, however, as more worker computers are added. The number of worker computers needed to approach the minimal execution time for a model run depends on the specific nature of the run.)

For several computers to work together on a MOVES run, all that is necessary is for the computers to have access to the SharedWork directory. For one computer this file directory can be on its local hard drive; other computers must access the SharedWork directory via a computer network. Of course, the platform requirements of each MOVES component must still be satisfied by the computer(s) on which it is installed. A variety of network configurations are possible. The principal consideration is that each Master/GUI Program must have the MOVESDefault database on its computer and that each Worker Program must be able to create a MOVESWorker database on its computer.

Two text files, MOVESConfiguration.txt, and WorkerConfiguration.txt, versions of which are built by the MOVES installation program, are used by the MOVES Master/GUI program and the MOVES Worker program to locate their databases and the SharedWork directory. The default configuration is:

Figure 4-2 shows a typical Multiple Computer configuration.

Typical Multiple Computer Configuration



5. MOVES Software Licensing

EPA distributes a complete installation package for MOVES as open source software. EPA asserts a copyright to the MOVES application but allows MOVES to be used pursuant to the GNU General Public License (GPL), which is widely used for the distribution of open source software.

Restrictions apply to use of MOVES pursuant to the GPL. For example, the program may not be sold, even in modified form, for commercial profit without obtaining a commercial license to MySQL from MySQL AB and, if redistributed in modified form, must be identified accordingly and source code included. Distribution of modified versions of the program also requires compliance with the GPL unless commercial licenses are obtained. The terms of the GPL are explained in detail at http://www.gnu.org/licenses/.

6. Installation Overview

The DRAFT MOVES2009 Program Distribution Suite contains all software components necessary to install and use DRAFT MOVES2009 on microcomputer systems based on the WINDOWS NT, WINDOWS 2000, WINDOWS XP and 32-bit Vista software operating systems. Installing MOVES on a 64-bit Vista machine is not recommended at this time since MySQL on 64-bit Windows Vista machines is not yet supported by MySQL AB. Users should be able to install and run MOVES and its components/tools on a 32-bit Vista machine. Please be advised that currently EPA won't be unable to provide further support for Vista until Vista has become part of EPA IT OS standards. The MOVES Program Suite Distribution is available for download from the EPA web site. Because of its size (several hundred MBs) it is highly desirable to have a high speed connection to the Internet to obtain this download.

The MOVES Program Suite Distribution includes a README file that summarizes the information provided in the remainder of this section.

In order to install and run several of the MOVES-related software components you must have administrative rights to the computer system(s) involved. Organizations are increasingly restricting these rights to a limited number of individuals. If you do not have these rights, you need to obtain them, or enlist the help of someone who does.

Assuming you have these rights and have obtained the MOVES Program Suite Distribution the first actual installation step is to install Java version 1.4.2 that MOVES requires or verify that it has already been installed. The MOVES Program Distribution Suite includes a separate installation package for this version of Java. **EPA intends this Java installation package to be used only for computers that do not already have Java installed**. If you are running older versions of Java you will need to upgrade to this version. This can be a complicated situation because this may affect preexisting Java applications on your computer, and EPA cannot provide support for this. Conversely, if you are already running a later version of Java, MOVES may not operate correctly with it. In this kind of situation you may wish to install MOVES on a different computer. If it is appropriate to run the Java version 1.4.2 installation package provided by EPA, just double-click on the installer installation program (j2sdk-1_4_2_03-windows-i586-p.exe provided in the java 1.4.2 directory) and follow the installer's instructions.

The second actual installation step is to install the MySQL database management system software that MOVES requires or verify that it has already been installed. The demonstration version of DRAFT MOVES2009 operates with MySQL version 5.0.27 and EPA's Distribution includes a separate installation package for this. **EPA intends this MySQL installation package to be used only for computers that do not already have MySQL installed**. If you are running older versions of MySQL you will need to upgrade to this version. This can be a complicated situation because this may affect preexisting MySQL applications on your computer, and EPA cannot provide support for this. Conversely, if you are already running a later version of MySQL, MOVES may not operate correctly with it. In this kind of situation you may wish to install MOVES on a different computer. If it is appropriate to run the MySQL installation provided by EPA you can read and follow the instructions in InstallMySQL5.doc:

All other required DRAFT MOVES2009 components can be installed by running the MOVESInstallationPackage.jar file included in the DRAFT MOVES2009 Program Installation Suite. This graphical "wizard-style" program guides the user through the process of installing the MOVES application. It was prepared with the IZPACK open source installation packaging tool. You will need to know the location where MySQL has been installed. Assuming you accepted the default location this is "c:\mysql". This installation creates three desktop icons. One executes the MOVES GUI and master program, a second executes the MOVES Worker Program, and a third icon can be used to "Uninstall" MOVES. The MOVES installation keeps track of components it installs and this "Uninstall" feature can be used to remove all those components, including the Java extensions needed for MOVES. It does not remove components which have been modified or which have been installed in some other fashion.

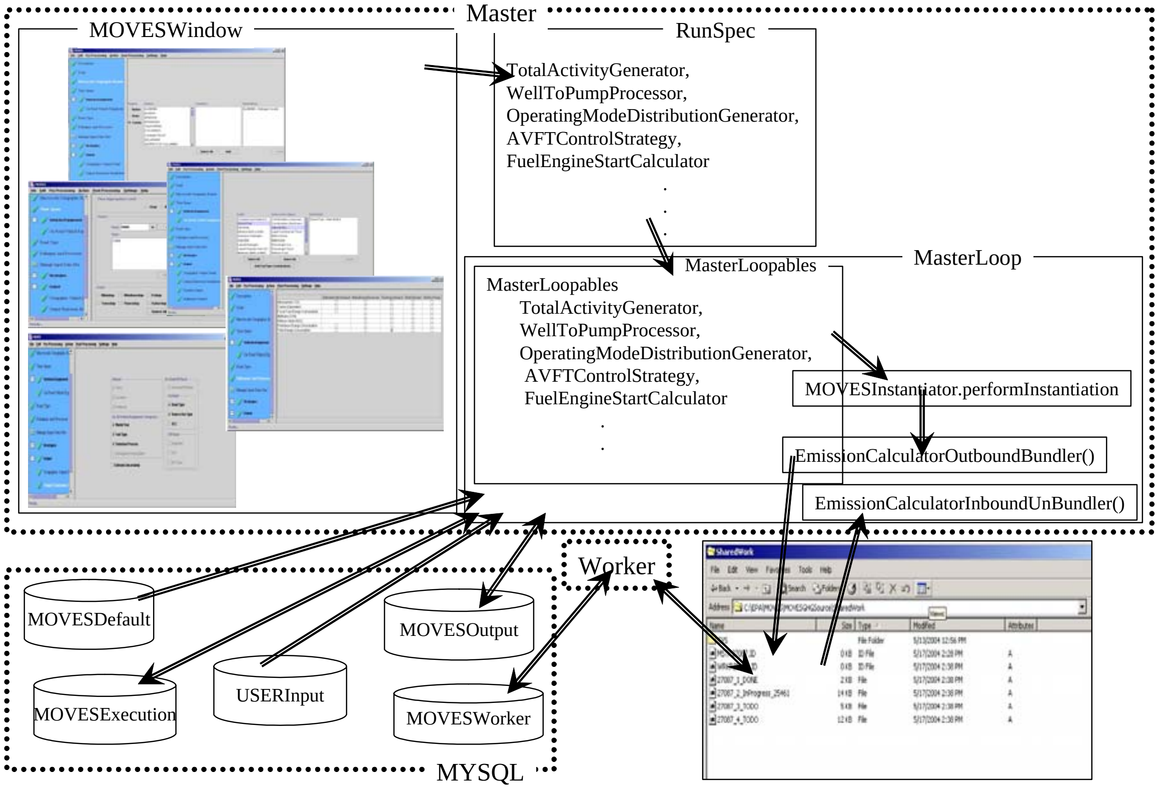
Users who desire a graphical client program to use with MySQL should execute the installation package for the MySQL Query Browser by following the steps in the README file. The MySQL Query Browser is a product of MySQL AB and replaces the earlier MySQL Control Center which EPA distributed with DRAFT MOVES2009.

Users who desire to use Microsoft Access or another DBMS to prepare or query MySQL tables via ODBC connections should execute the installation package for MySQL Connection ODBC by following the steps in the README file.

7. Processing Overview

The following diagram illustrates the overall flow of processing in MOVES in a way which illustrates the division of work between the MOVES Master and Worker programs.

MOVES Processing Overview Diagram



Typical Processing Steps:

Before beginning a model run any desired User Input datatabases (shown near the bottom left of the diagram) must be prepared. (The diagram does not attempt to show this intial step, which is only required if the user wishes to deviate from the default database inputs.) The components accessed via the “Pre-Processing Menu” in the MOVES GUI can be used to produce User Input Databases for particular purposes. A button in the MOVES graphical user interface, represented in the diagram by its MOVESWindow, can also be used to create an empty User Input database to which users can add the table records they need.

A run specification, or RunSpec, is loaded or produced by using the MOVES graphical user interface (MOVESWindow).

The user initiates execution of the actual model run via the “Action” menu item of MOVESWindow.

The MasterLoop, within the MOVES Master program, then merges any User Input databases identified in the RunSpec with the MOVESDefault database to produce the MOVESExecution database. Most data not needed to fulfill the RunSpec is discarded or “filtered” in this process.

The MasterLoop then uses the MOVESExecution database to produce files containing work “bundles” in the SharedWork directory.

MOVES Worker program(s) perform these bundles of work, using their MOVESWorker databases for temporary storage. They place files containing the completed work back into the SharedWork directory.

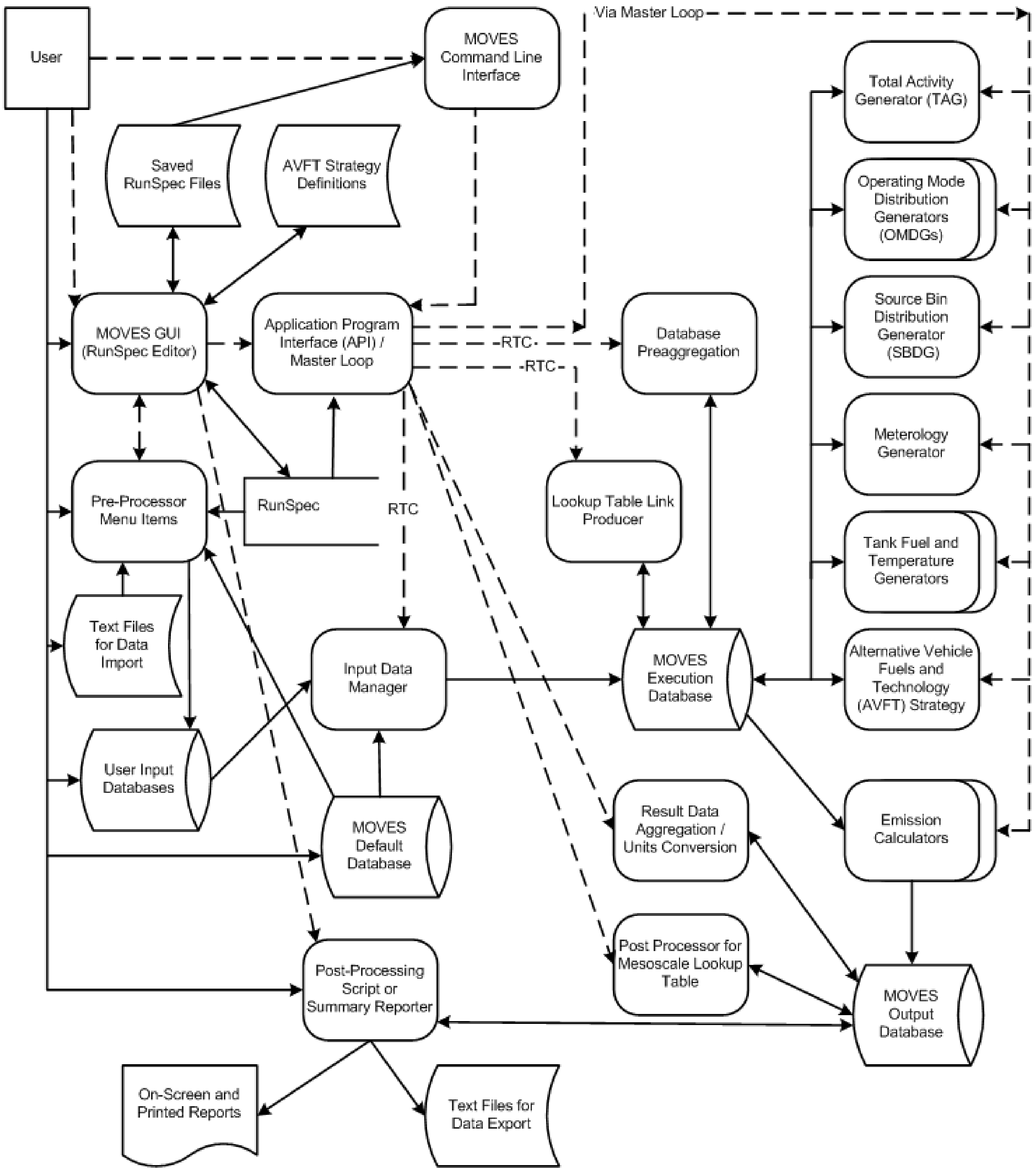
The MasterLoop retrieves these completed work files and processes them into a MOVES output database. The name of the output database is specified in the Run Spec.

The “Post Processing” menu in the MOVESWindow allows for additional, optional processing steps to be performed on the output database.

8. Data and Control Flow

Figure 8-1 illustrates the logical flow of data and control within the MOVES software. While generally more detailed than the diagram in the previous section, it does not attempt to illustrate the division of work between the MOVES Master and Worker.

Figure 8-1 MOVES Logical Level Data Flow Diagram


User
MOVES Command Line Interface
Via Master Loop
Saved RunSpec Files
AVFT Strategy Definitions
Total Activity Generator (TAG)
Operating Mode Distribution Generators (OMDGs)
MOVES GUI (RunSpec Editor)
Application Program Interface (API) / Master Loop
RTC
RTC
Database Preaggregation
Source Bin Distribution Generator (SBDG)
Meterology Generator
Pre-Processor Menu Items
RunSpec
RTC
Lookup Table Link Producer
Tank Fuel and Temperature Generators
Text Files for Data Import
Input Data Manager
MOVES Execution Database
Alternative Vehicle Fuels and Technology (AVFT) Strategy
User Input Databases
MOVES Default Database
Result Data Aggregation / Units Conversion
Emission Calculators
Post-Processing Script or Summary Reporter
Post Processor for Mesoscale Lookup Table
MOVES Output Database
On-Screen and Printed Reports
Text Files for Data Export

The graphical conventions used in this diagram are*:*

Cylinders represent databases.

Boxes with curved ends (simplified cylinders) represent data files.

Rectangles open on the right hand side represent temporary data storage elements.

Round-cornered rectangles represent processes that operate on and produce data.

Square boxes represent interfaces external to the system.

Solid line arrows represent data flow.

Dashed line arrows represent control flow.

The "RTC" notation on an arrow indicates that the control flow "runs to completion" before any of the data is processed by subsequent steps.

In order to illustrate the specifics of MOVES, several databases envisioned by the general MOVES design have been combined into the MOVESDefault and MOVESExecution databases illustrated here. Chapter 11 contains a more detailed discussion of these.

The following text attempts to lead the reader through the diagram. The first occurance of each diagram block is bolded.

The **User** normally executes the **MOVES GUI** program, which may access and update files containing **Saved Run Specifications** and **AVFT Strategy Definitions**. A simple **MOVES Command Line Interface** is available as an alternative to execute an existing run specification or **RunSpec**.

The Command Line Interface passes a selected Run Spec directly to the **MOVES Application Program Interface (API)**, whereas the MOVES GUI Program allows the user to select, edit, and save run specifications, and/or operate any **Pre-Processor Menu Items** before passing control to the MOVES API. These pre-processors typically use the MOVESDefault database and **Text Files for Data Import** to create **User Input Databases** which can be included in the run specification.

Once control is passed to the MOVES API, it manages the actual model run. Control is first passed to the **Input Data Manager,** which merges the **MOVESDefault** database with any **User Input Databases** specified by the RunSpec, producing the **MOVESExecution** database.

A **Database Pre-aggregation** function was added to MOVES to reduce execution time performance at the expense of some added approximation in the calculations. This function is executed next if called for by the run specification.

If the run specification specifies that the run will be performed at the "Mesoscale Table Lookup" scale the database is adapted for this at this stage by the **Lookup Table Link Producer**.

The **Master Loop** then manages execution of the generators, internal control strategies, and calculators. DRAFT MOVES2009 includes a **Total Activity Generator (TAG), several Operating Mode Distribution Generators (OMDGs**), a **SourceBinDistributionGenerator (SBDG),** a **Meteorology Generator, a Tank Fuel Generator, and a Tank Temperature Generator**. It includes one internal control strategy: the **Alternative Vehicle Fuels and Technologies (AVFT) Strategy.** All generators and control strategies receive input from tables in the MOVESExecution database and place their output there as well. Tables written by Generators are termed "Core Model Input Tables" or "CMITs" and have an important role in the design of MOVES. Additional detail as to which tables are used by each generator and control strategy is contained in Chapters 10, 11, and 12.

Emission Calculators are the most central or "core" portion of MOVES model. They consume much of its execution time and so the MOVES software design provides for portions of them to be run by the MOVES Worker Program. They receive input from the MOVESExecution database (some of which has been produced or altered by the generators and the control strategies). The CMIT tables within the MOVESExecution database provide their principal input data, but other tables may also be used by emission calculators for specialized calculations. The emission calculators output their results to databases on worker machines that are further processed to become the **MOVESOutput Database**. DRAFT MOVES2009 includes approximately 20 EmissionCalculators which calculate distance traveled and the emission results for various pollutants resulting from various emission processes. These are documented further in Chapter 10.

After all Generators, Internal Control Strategies and Emission Calculators needed to produce the results required by the run specification have executed, several more processing steps are required**.** **Result Aggregation and Engineering Units Conversion** functions are performed on the MOVESOutput database. For mesoscale table lookup an integrated **Post-Processor for Mesoscale Lookup** is then executed to produce an additional emission rate table in the output database.

Following the model run, the MOVES GUI can be used to invoke additional "post-processing" functions to operate on MOVESOutput databases. DRAFT MOVES2009 includes two such post-processing functions. **Post-Processing Script Execution**, runs a selected MySQL script against the MOVESOutput database specified by the run specification. Several post processing scripts are provided with the model and users may add to these if further customization of MOVES output is desired. A **Summary Reporter** has been added to this version of MOVES which allows the user to further aggregate the output results, produce summary on-screen or printed reports, and/or tab-separated variable ASCII files of selected results. These are suitable for importing into other software, such as spreadsheets, for further display and analysis.

9. Functional Design Concepts

The functional scope of this MOVES version can be characterized as follows:

Geography: The entire U.S. (plus Puerto Rico and the U.S Virgin Islands) at the county level. There are options to run at a more aggregate state or national level. By modifying the database, counties can be divided into zones.

Time Spans: Energy/emission output by hour of the day, and month for calendar years 1990 and 1999 through 2050, with options to run at more aggregate month or year levels.

Sources: All highway vehicle sources, divided into 13 "use types"

Outputs and Pollutant Emissions: Energy consumption (characterized as total energy, petroleum-based energy and fossil fuel-based energy), N2O, CH4, Atmospheric CO2, CO2 equivalent, total gaseous hydrocarbons, CO, NOx, and several forms of particulate matter (PM).

Emission Processes: running, start, extended idle (e.g. heavy-duty truck "hoteling"), well-to-pump, brakewear, tirewear, evaporative permeation, evaporative fuel vapor venting, and evaporative fuel leaks.

Understanding how MOVES operates and why the input and output databases are set up as they are requires an understanding of some basic functional design concepts. Fundamental aspects of the MOVES design are geographic locations, time periods, emission sources, emission pollutants, emission processes, vehicle fuels, and emission source activity.

9.1. Geographic Locations

The default geographic modeling domain in DRAFT MOVES2009 is the entire United States of America. This domain is divided first into "*states*." In this context the District of Columbia, Puerto Rico, and the U.S.Virgin Islands are considered to be "states", so the nation has 53 states in the default MOVES database.

"States" are divided into "*counties*." In the default MOVES database these correspond to the 3222 political subdivisions of the states in 1999. Counties must belong to a single state. While the county-level subdivisions of some states have changed slightly over time, the MOVES database is set up to store only a single set of counties. The database could be adapted relatively easily to a different set of counties, but it would be a significant structural change to MOVES to allow the set of counties to depend upon the calendar year.

In the general MOVES design framework, “counties” may be further divided into “*zones*,” but in the default database each county is consists of a single zone. Zones are intended to play a distinct role in future versions of MOVES that operate at smaller scales.

“Zones,” and thus “counties” in the default database, are further divided into “*links*.” In this version of the MOVES default database each county is divided into five links; four of these represent actual roadways and one represents locations not on the county’s roadway network. This set of roadtypes has been reduced, relative to those in the default database distribution with previous versions of MOVES which had a total of 13 roadtypes. The five roadtypes in this version of MOVES are:

1	Locations which are off of the highway network
2	Rural restricted access roadways (i.e. freeways and interstates)
3	Rural roads to which vehicle access is unrestricted
4	Urban restricted access roadways (i.e. freeways and interstates)
5	Urban roads to which vehicle access is unrestricted

Each of these roadtypes is a combination of one or more of the 13 roadtypes used in the database distributed with previous versions of MOVES. The set of roadtypes used in MOVES is driven by the database and changing this set does not require changes to the MOVES program itself.

Running, tirewear, brakewear, and some evaporative process emissions are considered to occur on the four “real roadway” roadtype locations, while its other emission processes (i.e. start, extended idling, well-to-pump, and most evaporative emissions) are associated with the “off network” link locations. (Emission processes are discussed in a subsequent section.)

In this version of MOVES at macroscale, a link is a combination of a road type and a county or zone. Road types, while not in themselves geographic locations, help to define links at the macroscale. This version of moves can also be run in a “mesoscale table lookup” mode in which a link represents all highway segments in the county or zone which have the same roadtype and average speed. In future versions of MOVES, when modeling at smaller scales, links may represent segments of actual roadways and road types will serve only to classify links.

Finally, the general MOVES geographic framework envisions that zone locations may be overlaid by a set of "*grids*," or grid cells. There may be multiple grid cells in a zone and multiple zones in a grid cell, but grids and grid cells play no role in this version.

9.2. Time Periods

MOVES describes time in terms of calendar years, 12 months of the year, portions of the seven-day week which are termed "Days" (but which may include more than one 24 hour period), and 24 hours of the day. This arrangement appears simple but there are several subtleties to keep in mind:

A "Day" in MOVES is really best thought of as a "portion of the week". It does not have to represent a single 24 hour period. It may represent several 24-hour periods, or even the entire week. The default database for this version of DRAFT MOVES2009 divides the week into a 5-day "weekday" portion and a 2-day "weekend" portion. While calendar years in MOVES are intended to represent actual historical years, the finer time period classifications (months, portions of the week, and hours) are best thought of as being "generic" time classifications. For example MOVES does not attempt to model the fact that a holiday occurs on a weekend in one year but occurs on a weekday in another.

Another reason why MOVES should not normally be considered to model historical time periods smaller than a calendar year is the disconnection between "weeks" and "months." For example, there is no way to specify data for more than one week in a single historical month, such as May 2004, in a single MOVES database. The database is designed to store information about the year 2004 and about all weeks in May. Along the same lines, MOVES does not attempt to model facts such as that a given month may contain more weekend days in some years than others. MOVES does account for the different number of days in each month, dividing this by 7 to determine the number of weeks it is considered to contain, and MOVES does account for leap years.

In order to model an actual historical time period, data corresponding to the unique time period could be supplied by the user, and the user could make this association outside the model.   Depending on the accuracy and detail desired, multiple model runs might be necessary.

In most portions of the model the month, portion of the week, and hour time periods are simply categories, and do not even have an assumed sequence. Because estimation of evaporative emissions is based on an hourly "diurnal" temperature cycle all 24 hourly categories of the day must be included in the run specification when estimating evaporative emissions, and this area of the model does assume an hourly time sequence.

9.3. Characterizing Emission Sources (Vehicle Classification)

A long-standing challenge in the generation of on-road mobile source emission inventories is the disconnect between how vehicle activity data sources characterize vehicles and how emission and fuel economy regulations characterize vehicles. The crux of this issue is that there is a fundamental difference between factors influencing how vehicles are used, and their fuel consumption and emission performance. An example of this is how vehicles are characterized by the Highway Performance Monitoring System (HPMS) – by a combination of the number of tires and axles – and EPA's weight-based emission classifications such as LDV, LDT1, LDT2 etc.

This disconnect is fundamental to matching activity data and emissions data, and generally requires some "mapping" of activity data to emission data. The MOBILE series of models have traditionally grouped vehicles according to the EPA emission classifications, and provided external guidance on mapping these categories to the sources of activity data, such as HPMS. MOVES is designed to take these mappings into account internally, such that the casual user of MOVES will not have to deal with external mapping. Doing this, however, requires some complexity in the design. Vehicles are characterized both according to activity patterns and energy/emission performance, and are mapped internal to the model. Thus the model uses data for both the activity and energy/emission methods of characterization. On the activity side, vehicles are grouped into "*Source Use Types*," or "*Use Types*", which are expected to have unique activity patterns. Because the HPMS is a fundamental source of activity information, the MOVES use types are defined as subsets of the HPMS vehicle classifications. These use types are shown in Table 9-1.

Table 9-1. MOVES Source Use Type Definitions

HPMS Class	MOVES use type	Description
Passenger Cars	21. Passenger Car	
Other 2-axle / 4-tire Vehicles	31. Passenger Truck	Minivans, pickups, SUVs and other 2-axle / 4-tire trucks used primarily for personal transportation
	32. Light Commercial Truck	Minivans, pickups, SUVs and other trucks 2-axle / 4-tire trucks used primarily for commercial applications. Expected to differ from passenger trucks in terms of annual mileage, operation by time of day
Single Unit Trucks	51. Refuse Truck	Garbage and recycling trucks Expected to differ from other single unit trucks in terms of drive schedule, roadway type distributions, operation by time of day
	52. Single-Unit Short-Haul Truck	Single-unit trucks with majority of operation within 200 miles of home base
	53. Single-Unit Long-Haul Truck	Single-unit trucks with majority of operation outside of 200 miles of home base
	54. Motor Home	
Buses	41. Intercity Bus	Buses which are not transit buses or school buses, e.g. those used primarily by commercial carriers for city-to-city transport.
	42. Transit Bus	Buses used for public transit.
	43. School Bus	School and church buses.
Combination Trucks	61. Combination Short-Haul Truck	Combination trucks with majority of operation within 200 miles of home base
	62. Combination Long-Haul Truck	Combination trucks with majority of operation outside of 200 miles of home base
Motorcycles	11. Motorcycle	

Activity patterns which may differ between the use types are: annual mileage, distribution of travel by time of day or day of week, driving schedule (i.e. real time speed/accel profile), average speeds, and distribution of travel by roadway type. For example, refuse trucks are separated out because their activity patterns are expected to vary significantly from other single-unit trucks, and accurately accounting for these vehicles requires accounting for their unique activity.

Source use types are the principal method of vehicle characterization seen by the MOVES user. The user selects which use type and fuel combinations to model in the user interface, and results are best reported by use type.[1] However, emission rates contained in the model are not broken down by use type, for two (related) reasons: first, emission and fuel consumption data are not gathered according to use types or other activity-based classifications (e.g. HPMS). Second, the factors that influence fuel consumption and emission production are different from how vehicles are used. For example, with regard to fuel consumption, loaded vehicle weight is a predominant influence; a 2000 lb. compact car and 5000 lb. SUV will have very different fuel consumption levels, although these vehicles may have similar use patterns. It is necessary to account for these differences in fuel consumption and emission generation separately from activity patterns. To do this, the MOVES design has implemented the concept of "*Source Bins*." Unique source bins are differentiated by characteristics that significantly influence fuel (or energy) consumption and emissions – and because these vary by pollutant, they are allowed to vary by pollutant in MOVES. Table 9-2 shows the source bins fields used in MOVES, which vary by pollutant. Energy source bins are defined by fuel type, engine type, model year group, loaded weight and engine size. For most other polluants, source bins are defined by fuel type, engine type, model year group, and regulatory class. The definition of model year group can vary by pollutant-process.

[1] Because Source Classification Codes (SCCs) have been and are used extensively in emission inventories DRAFT MOVES2009 also offers the option of reporting results by SCC. There are currently 144 SCCs for mobile sources formed by the intersection of the 12 HPMS road types with the 12 vehicle classifications used in PART5 and NMIM. This scheme is not native to MOVEs, however, and the MOVEs modeling team discourages continued use of this classification scheme.

Table 9-2a. MOVES Source Bin Definitions (other than ModelYearGroup)

Fuel Type (All Pollutants)	**Engine Technology** (All Pollutants)	**Loaded Weight** (Energy)	**Engine Size** (Energy)	**Regulatory Class** (All pollutants except energy and evap permeation)
Gas	Conventional IC (CIC)	Null	Null	Null
Diesel	Advanced IC (AIC)	< 500 (for motorcycles)	< 2.0 liters	Motorcycle
CNG	Hybrid - CIC Moderate	500-700 (for motorcycles)	2.1-2.5 liters	LDV
LPG	Hybrid - CIC Full	> 700 (for motorcycles)	2.6-3.0 liters	LDT
Ethanol (E85)	Hybrid - AIC Moderate	<= 2000 lbs	3.1-3.5 liters	HD gasoline GVWR <= 14K lbs
Methanol (E85)	Hybrid - AIC Full	2001-2500	3.6-4.0 liters	HD gasoline GVWR > 14K llbs.
Gas H_2	Fuel Cell	2501-3000	4.1-5.0 liters	LHDD
Liquid H_2	Hybrid - Fuel Cell	3001-3500	> 5.0 liters	MHDD
Electric	Electric	3501-4000		HHDD
		4001-4500		Urban Bus
		4501-5000		
		5001-6000		
		6001-7000		
		7001-8000		
		8001-9000		
		9001-10,000		
		10,001-14,000		
		14,001-16,000		
		16,001-19,500		
		19,501-26,000		
		26,001-33,000		
		33,001-40,000		
		40,001-50,000		
		50,001-60,000		
		60,001-80,000		
		80,001-100,000		
		100,001-130,000		
		>=130,001		

Table 9-2b. MOVES Source Bin Definitions (ModelYearGroup)

Model Year Group					
Energy	CH_4, N_2O	HC - Evap	HC, CO, NOx, PM start, running	HC, CO, NOx, PM extended idle	Sulfate PM (ratios to energy)
1980 and earlier	1972 and earlier	1970 and earlier	1980 and earlier	1980 and earlier	1980 and earlier
1981-85	1973	1971-1977	1981-1982	1981-85	1981 and later
1986-90	1974	1978-1995	1983-1984	1986-90	
1991-2000	1975	1996-2003	1985	1991-2000	
2001-2010	.	2004	1986-1987	2001-2006	
2011-2020	.	2005	1988-1989	2007-2010	
2021 and later	.	.	1990	2011-2020	
	1999	.	1991-1993	2021 and later	
	2000	2019	1994		
	2001-2010	2020	1995		
	2011-2020	2021 and later	.		
	2021 and later		.		
			2019		
			2020		
			2021 and later		

Source bins are defined independently from use types, but are mapped to use types internal to MOVES by the SourceBinDistributionGenerator.

9.4. Emission Pollutants

MOVES estimates two fundamentally different kinds of results: energy consumption and mass emissions. For convenience, all these quantities are considered to be “emissions.” “Energy emissions” estimated by DRAFT MOVES2009 are total energy consumption, fossil fuel energy consumption, and petroleum fuel energy consumption. The more familiar mass emissions estimated by DRAFT MOVES2009 are total gaseous hydrocarbons (THC), carbon monoxide (CO), oxides of nitrogen (NOx), sulfate particulate matter, tire wear particles under 2.5 microns, brake wear particles under 2.5 microns, methane (CH4), nitrous oxide (N2O), carbon dioxide (CO2) on an atmospheric basis, and the “CO2-equivalent” of CO2 combined with N2O and CH4.

9.5. Emission Processes

On-road vehicles consume energy and produce mass emissions through several mechanisms or pathways, which are known within MOVES as “*emission processes*,” or just “*processes*,” and are accounted for and reported (if desired by the user) separately. The MOVES mechanisms for “pump-to-wheel” energy consumption are limited to operation of the engine and emissions from the tailpipe. The MOVES mechanisms for gaseous emissions, however, include other processes such as fuel evaporation, tire wear and brake wear, which merit treatment as separate emission processes. In addition to all these “pump-to-wheel” energy and exhaust emission processes MOVES also includes a “well-to-pump” emission process. The processes for DRAFT MOVES2009 are as follows:

Running Exhaust, meaning the energy consumed or the tailpipe emissions produced during vehicle operation over freeways and surface streets while the engine is fully warmed up.

Start Exhaust, meaning the additional energy consumed or tailpipe emissions produced during the period immediately following vehicle start-up. An important note is that this quantifies the energy consumed or emissions produced *in addition* to the “running” energy/emissions produced immediately following start-up. Start emissions represent the incremental emissions produced following vehicle start-up, after accounting for the baseline running emissions.

Extended Idle, meaning energy consumed or tailpipe emissions produced during long periods of engine idling off of the roadway network. This process applies only to combination long-haul trucks in the current version of DRAFT MOVES2009, and is meant to account for the issue of overnight "hoteling" at truck stops, although it could eventually be applied to idling of passenger vehicles in drive-thru lanes, etc.

Evaporative Fuel Permeation, meaning the migration of hydrocarbons through the various elastomers in a vehicle fuel system.

Evaporative Fuel Vapor Venting, meaning the expulsion into the atmosphere of fuel vapor generated from evaporation of fuel in the tank. Also includes evaporation into the atmosphere of fuel which has "seeped" to the surface of vehicle parts.

Evaporative Fuel Leaking, meaning the "gross" leaking of fuel, in liquid form, from the vehicle. This is assumed to subsequently evaporate, outside the vehicle, into the atmosphere.

Brakewear, meaning the formation of particles of brake components which are formed during operation of vehicle brakes.

Tirewear, meaning the formation of tire material particles during vehicle operation

Well-To-Pump, meaning the energy and emissions produced from processing and distributing vehicle fuel from raw feedstock to the fuel pump. These energy use and emission rates are produced by a version of Argonne National Laboratory's GREET model. DRAFT MOVES2009 has not been expanded in this area relative to MOVES-HVI or MOVES2004 and so only reports the well-to-pump energy consumption and greenhouse gas emissions. It should be noted that well-to-pump emissions have a different relationship to locations than those of the other processes. MOVES associates well-to-pump results with the locations (e.g. Counties and RoadTypes) whose activity *caused* the emissions; the emissions *are not produced* at these locations as they are for the other processes. It should also be noted that GREET in DRAFT MOVES2009 has been disabled and the emission rates of well-to-pump were set to 0 (zero).

An additional process, manufacture/disposal, would account for energy and emissions from vehicle production and disposal. This is not yet included in MOVES.

9.6. Vehicle Fuel Classifications

The top level vehicle fuel classifications are listed in Table 9-2, above, in the context of their role in vehicle source bin classification. Implicit in this source bin classification scheme is that a particular vehicle is designed to operate on one kind of fuel. The MOVES term for this top-level classification of vehicle fuels is “fuel type”. DRAFT MOVES2009 considers the following fuel types:

Gasoline
Diesel Fuel
Compressed Natural Gas (CNG)
Liquid Propane Gas (LPG)
Ethanol (E85)
Methanol (M85)
Gaseous Hydrogen
Liquid Hydrogen
Electricity

To facilitate modeling the effects of alternative fuels on greenhouse gas emissions, MOVES further divides these top level fuel types into fuel subtypes. In the default MOVES database, for example, the gasoline fuel type has three subtypes: conventional, reformulated, and gasohol (E10). Diesel fuel has three subtypes: conventional, biodiesel, and Fischer-Tropsch diesel. Fuel subtypes represent alternative ways of meeting the demand for a general type of fuel. MOVES assumes that vehicles designed to operate on a top level fuel type may be operated on any of its subtypes depending upon the fuel supply at a particular time and geographic location.

This fuel classification scheme was expanded further in DRAFT MOVES2009 to further divide fuel subtypes into more specific “fuel formulations” which may be thought of as a batch of fuel having specific values of measurable properties such as RVP, sulfur content, and oxygenate content. This additional breakdown is necessary because these fuel characteristics affect the emissions of pollutants added in DRAFT MOVES2009 and vary within a fuel subtype. MOVES assumes that vehicles designed to operate on a top level fuel type may be operated on any formulation of any of its fuel subtypes depending upon the fuel supply at a particular time and geographic location.

9.7. Emission Source Activity

The cornerstone of estimating mobile source energy usage and emission inventories is vehicle activity. Vehicle activity centers on two fundamental questions: what is the total amount of vehicle activity, and how is this activity subdivided into modes that are unique in regards to energy consumption and emissions. The first question is quantified in MOVES by the metric Total Activity. Total Activity, as the name implies, is the total amount of vehicle activity for source use types in the given location and time which the user has selected in the run specification. The basis of total activity depends on the emission process, as shown in Table 9-3. In DRAFT MOVES2009 the Total Activity Generator (TAG) estimates Total Activity for all emission processes except well-to-pump. A simplified version of the TAG is used for runs involving mesoscale table lookup.

Table 9-3. Total Activity Basis by Process

Emission Process	Total Activity Basis	Description
Running Tire wear Brake wear	Source Hours Operating (SHO)	Total hours, of all sources within a source type, spent operating on the roadway network for the given time and location of the run spec. The same as number of sources * per-source hours operating
Evaporative Fuel Permeation, Vapor Venting and Leaking	Source Hours	Total hours, of all sources within a source type for the given time and location of the run spec. This is equivalent to the population of the source type times the number of hours in the time period.
Start	Number of Starts	Total starts, of all sources within a source type, for the given time and location of the run spec. The same as number of sources * per-source starts
Extended Idle	Extended Idle Hours	Total hours, of all sources within a source type, spent in extended idle operation for the given time and location of the run spec.
Well-To-Pump	Pump-To-Wheel Energy Consumed	Total energy consumed, of all sources within a source type, for the given time and location of the run spec. The sum of running, start and extended idle.

The second piece of activity characterization is to define how this total activity may be subdivided into operating modes which produce unique energy consumption and emission rates. The operating mode concept is central to MOVES multi-scale analysis capability, and has been expanded in DRAFT MOVES2009. In the MOVES design these operating modes are allowed to vary by emission process and pollutant, and for some pollutant-processes Total Activity is not further divided into multiple operating modes.

For the running emission process for all pollutants except CH4 and N2O, the total source hours operating (SHO) activity basis is broken down into operating modes representing ranges of vehicle speed and vehicle specific power (VSP). The operating modes used for the running emission process are shown in Tables 9-4 and 9-5.

Table 9-4. Operating Mode Bin Definitions for Running Energy Consumption)

Braking (Bin 0)			
Idle (Bin 1)			
VSP \ Instantaneous Speed	**0-25mph**	**25-50**	**>50**
< 0 kW/tonne	Bin 11	Bin 21	-
0 to 3	Bin 12	Bin 22	-
3 to 6	Bin 13	Bin 23	-
6 to 9	Bin 14	Bin 24	-
9 to 12	Bin 15	Bin 25	-
12 and greater	Bin 16	Bin 26	Bin 36
6 to 12	-	-	Bin 35
< 6	-	-	Bin 33

Table 9-5. Operating Mode Bin Definitions for Running THC, CO, NOx

Braking (Bin 0)			
Idle (Bin 1)			
VSP \ Instantaneous Speed	**0-25mph**	**25-50**	**>50**
< 0 kW/tonne	Bin 11	Bin 21	
0 to 3	Bin 12	Bin 22	
3 to 6	Bin 13	Bin 23	
6 to 9	Bin 14	Bin 24	
9 to 12	Bin 15	Bin 25	
12 and greater	Bin 16		
12 to 18		Bin 27	Bin 37
18 to 24		Bin 28	Bin 38
24 to 30		Bin 29	Bin 39
30 and greater		Bin 30	Bin 40
6 to 12			Bin 35
< 6			Bin 33

The start exhaust process emissions of THC, CO, and NOx are distinguished into operating modes which represent the length of time the engine was off prior to starting as follows:

Table 9-6. Operating Modes for Start Process – THC, CO, and NOx

opModeID	**minSoakBound in minutes**	**maxSoakBound in minutes**
101	null	6
102	6	30
103	30	60
104	60	90
105	90	120
106	120	360
107	360	720
108	720	null

The evaporative processes (fuel tank vapor venting, fuel permeation and liquid leaking) have three operating modes: "operating", "hot soaking" and "cold soaking" where "hot soaking" is considered to be time (Source Hours) when the engine is not operating, but has not yet cooled to a point where it is near the temperature it would be if it had not been operating.

The brake wear process divides its Source Hours Operating activity basis into periods where the vehicle brakes are being applied and all other operating time, during which no brake wear emissions are considered to occur.

Other pollutant-processes are modeled without breakdown of their activity basis into more detailed operating modes.

9.8. Modeling Vehicle Inspection/Maintenance Programs

The MOVES model contains two basic classes for estimating Inspection and Maintenance program (I/M) benefits. One is the exhaust I/M calculation class and the other is evaporative I/M calculation class. Both classes have a similar basic algorithm where the I/M emission reduction fraction is the product of an I/M Coverage fraction and an Emission Reduction fraction.

9.8.1. <u>I/M Coverage</u>

Information about which pollutant-processes are "covered" by I/M programs in various counties and calendar years is contained in the MOVES database table IMCoverage. This coverage information is allowed to vary by pollutant - process, county, year, regulatory class, and fuel type. The principal piece of information contained in the IMCoverage table is the compliance factor. It attempts to represent (in practice) a particular I/M program's ability to achieve theoretical design benefits for their program. It may vary from 0 to 1.0 where zero would represent a totally failed program and 1.0 a perfectly successful program. Factors which tend to reduce the compliance factor are the systematic waiver of failed vehicles from program requirements, the existence of large numbers of motorists who completely evade the program requirements, technical losses from improperly functioning equipment or inadequately trained technicians.

Other data in the IMCoverage table includes the concept of I/M program sub-types (called IMProgramID). A particular county will likely have several IMProgramIDs that reflect different test types, test standards or inspection frequencies being applied to different regulatory classes, model year groups or pollutant- process combinations. For example, County A in calendar year 2007 may have an IMProgramID=1 that annually inspects pre-1981 model year cars using an Idle test, and an IMProgramID=2 that biennially inspects 1996 and later model year light-trucks using an OBD-II test.

The IMCoverage table also shows other important I/M parameters for each IMProgramID. These include the model year information as a model year range (i.e., beginning and ending model year), the frequency of inspection (i.e., annual, biennial and continuous (i.e., monthly)), and test type (Idle, IM240, ASM, OBD-II) and test standard.

9.8.2. <u>I/M Effectiveness</u>

Information about the theoretical effectiveness of an I/M program design is contained in the MOVES database table IMFactor. The effectiveness information varies by pollutant - process, inspection frequency, test type, test standards, regulatory class, fuel type, model year group and age group. All MOVES I/M effectiveness values (IMFactor variable) were empirically generated from MOBILE6.2 runs. The Arizona I/M program as generally operated from 1995 through 2002 was used as the reference case. It receives an IMFactor of unity. Arizona's I/M program was picked as the reference because the underlying emission factors in MOVES are based on Arizona I/M testing. The principal piece of information contained in the IMCoverage table is the IMFactor. It represents the comparative effectiveness of one particular I/M test to identify and reduce emissions.

The IMFactor and IMCompliance rate are multiplied together to compute the IMAdjustFract. It is used in MOVES in the following general equation, and is labeled as variable 'R'. This equation is used to weight the I/M and Non I/M emission rates together in MOVE to compute a composite result.

$$E_p \quad = \quad E_{im} * R \ + \ E_{noim}*(1 - R)$$

E_p is the Target I/M program emission rate from MOBILE6.2

E_{im} is the Reference I/M program (Arizona) emission rate from MOBILE6.2

E_{noim} is the Reference NON I/M program emission rate from MOBILE6.2

R is the I/M adjustment factor

Rearranging and solving for R equals:

$$R = (E_p - E_{noim}) / (E_{im} - E_{noim})$$

or

$$R = E_p / (E_{im} - E_{noim}) - Enoim / (E_{im} - E_{noim})$$

I/M-1 Merge IMCompliance and IMFactor

This section computes the product of IMCompliance and IMFactor

Input Variables:

PollutantProcessModelYear. polProcessID,
PollutantProcessModelYear. modelYearID,
IMFactor. fuelTypeID,
IMFactor. regClassID,
IMFactor. IMFactor
IMCoverage. complianceFactor

Output Variable:

IMAdjustFract

weightFactor

Joining and Conditional Variables

polProcessID
countyID
yearID
modelYearID
inspectFreq
testTypeID
testStandardsID
regClassID
fuelTypeID
begModelYearID
endModelYearID

Calculation:

IMAdjustFract = (IMFactor*complianceFactor*.01)

complianceFactor as weightFactor

I/M-2 Combine I/M and Non I/M Emission Rates

This section weights the I/M and Non I/M Emission Rates using the IMAdjustFract

Input Variables:

IMAdjustmentWithSourceBin. zoneID,

IMAdjustmentWithSourceBin, yearID,

EmissionRateByAge. polProcessID,

IMAdjustmentWithSourceBin. modelYearID,

EmissionRateByAge. sourceBinID,

EmissionRateByAge. opModeID

IMAdjustmentWithSourceBin. IMAdjustFract

EmissionRateByAge. meanBaseRate

Output Variable:

meanBaseRate

Joining and Conditional Variables

polProcessID
sourceBinID,
ageGroupID

Calculation:

meanBaseRate = (meanBaseRateIM * IMAdjustFract + meanBaseRate * (1.0-IMAdjustFract)

10. MOVES Functional Specifications

This chapter explains the functions, including calculations, performed by each portion of the MOVES software, as shown in figure 8-1, or indicates where such information can be found.

10.1. Graphical User Interface (GUI) / Run Specification Editor

The MOVES Graphical User Interface (GUI) is used to produce and modify MOVES run specifications. Its functionality is described in detail in the DRAFT MOVES2009 User Guide. Components whose principal functionality is evident from the GUI, such as the I/M Coverage Table Editor, are also documented in the User Guide.

10.2. Application Program Interface and Master Looping Mechanism

This basic component manages the overall execution of a MOVES model run. It includes an application program interface (API) callable by either the command line interface, or the MOVES GUI. This component invokes, directly or indirectly, the Input Data Manager, any required Database Preaggregation, and, when performing a mesoscale table lookup run, the Lookup Table Link Producer. These components execute before the master looping mechanism is invoked.

Once these components have run to completion, a master looping mechanism is executed. InternalControlStrategy, Generator, and EmissionCalculator objects have "signed up" with this MasterLoop to execute over portions of the modeling domain. (The modeling domain is defined by a RunSpec in terms of the emission processes, geographic locations and time periods being modeled.)

If uncertainty estimation is being performed in the run, which involves running multiple iterations, the looping process manages these iterations.

The MasterLooping mechanism uses a pool of control "threads" to bundle Emisson Calculator input data (and SQL scripts to be run on the data) for portions of the modeling domain and place them in the SharedWork directory. Another thread of control is established to unbundle the results placed in the SharedWork directory by MOVES Worker program(s). This thread also leads to the performance of the final result aggregation and units conversion functions. If a mesoscale table lookup run is being performed, an integrated post-processor is also invoked to create an additional table of emission rates in the output database.

This software component is obviously rather complicated and consists of a number of Java classes. Programmer level documentation is required to understand this component in greater detail.

10.3. Input Data Manager

The InputDataManager generates the MOVESExecution database from the MOVESDefault database, removing (or "filtering") records where possible based on the needs of the run specification (RunSpec). The InputDataManager runs to completion before any InternalControlStrategy objects, Generators or Calculators begin the MOVES model calculations. The MOVES GUI and RunSpecs specify a list of "user input databases" to be used in addition to the default database to construct the MOVESExecution database.

The InputDataManager filters input records based on the following RunSpec criteria: year, month, link, zone, county, state, pollutant, emission process, day, hour, day and hour combination, roadtype, pollutant-process combination, sourceusetype, fueltype, fuelsubtype, and monthgroup. This filtering is specified in a table-specific manner and need not be applied to every table having these key fields. Filtering is not performed on particular table columns where doing so would interfere with correct result calculation. (The exact table columns to filter are specified in the Java code for the InputDataManager class in the "tablesAndFilterColumns" array.) The reasons for not filtering a particular table by all possible criteria are documented with program source code comments.

Input databases have the same table contents and structure as the MOVESDefault database, but need not contain all tables. If a table is present, however, it must contain all the table columns. Records from user input databases add to or replace records in the MOVESDefault database. If the same record (i.e. a record having the same values of all primary key fields but generally different non-key field values) is present in more than one input database, the record from the user input database listed last is the one which ends up in MOVESExecution.

The InputDataManager addes a field to core model input tables in the MOVESExecution Database to indicate that the records came from a user input database so that Generators may avoid deleting such records.

MOVES verifies that all input tables present in user input databases contain the required columns. The MOVES GUI also checks that the same database is not specified for both input (either as the default input database or an additional input database) and for output, and ensures that MOVESExecution is not used as either an input or an output database. Otherwise, however, it remains the responsibility of the user to ensure that the ordered application of any additional input databases called for in the run specification to the MOVESDefault database results in a MOVESExecution database that is accurate, complete and consistent.

10.4. Database Pre-Aggregation

To improve execution run time performance, when geographic selections are made at the state or national level, MOVES “preaggregates” the MOVESExecution database so that each state selected, or the entire nation, appears in the database as if it were a single county. These geographic performance shortcuts are specified by the “STATE” and “NATION“ GeographicSelectionType values produced by the “Macroscale Geographic Selection” GUI screen and stored in MOVES run specifications. No database preaggregation is performed when geographic selections are made at the County level. County selections may still be used to produce results for broad geographic areas if the user can endure their execution time performance. Geographic preaggregation is not allowed for Mesoscale Lookup calculations.

Options are also available to improve execution run time performance by “preaggregating” time periods in the MOVESExecution database. These options are specified by the “Time Aggregation Level” item in the MOVES GUI and MOVES RunSpec. This can assume the following values:

HOUR - no time period aggregations are performed. This is required for evaporative emission calculations.

DAY - combine all hours of the day

MONTH - combine all portions of the week (though the default MOVES database may not divide the week into smaller portions).

YEAR - combine months of the year

All of these computational shortcuts (except COUNTY and HOUR) involve compromises to the accuracy of the results.

The MOVES GUI adjusts the levels of geographic and time period detail specified for the output if necessary so that levels of output detail which can no longer be produced due to data preaggregation are not requested by the RunSpec.

10.4.1. Sequence of the Database Pre-Aggregation Operations:

After creation of the MOVESExecution database by the input data manager the geographic and time period preaggregation operations are performed as follows:

a. If the GeographicSeletionType = NATION, the model creates an average county which represents the entire nation. To do this the MOVESExecution database is aggregated to a level where the nation consists of a single representative state and this “state” consists of a single “county”, and a single “zone”. For macroscale there is a single “link” for each road type in the RunSpec.

b. If the GeographicSeletionType = STATE, then the MOVESExecutionDatabase is aggregated to a level where each state selection in the RunSpec consists of a single “county” and a single “zone”. For macroscale there is a single “link” in each such state for each road type in the RunSpec.

c. if the Time Aggregation Level value is DAY, MONTH, or YEAR, all data pertaining to the 24 separate hours of the day in the MOVESExecution database is aggregated into a single “pseudo-hour” representing the entire day. Time period preaggregation is not allowed if evaporative emissions are being estimated.

d. if the Time Aggregation Level value is MONTH, or YEAR, all data pertaining to any day-based portions of the week in the MOVESExecution database is further aggregated into a single “pseudo-day” representing the entire week. If the Default MOVES Database divides the week into a 5 weekday portion and a 2 weekend day portion, MONTH or YEAR data preaggregation would remove this distinction.

e. if the Time Aggregation Level value is YEAR, all data pertaining to the 12 separate months of the year in the MOVESExecution database are further aggregated into a single “pseudo-month” representing the entire year.

f. Following any of the pre-aggregation operations performed in steps a. thru e., the set of ExecutionLocations used by the MasterLoop is recalculated based on the aggregated database.

g. If the DAY, MONTH, or YEAR aggregations have been performed all information derived from the run specification used throughout the remainder of the run is made consistent with the aggregated time periods.

These operations run to completion before any MOVES MasterLoopable objects (ControlStategy objects, Generators, and EmissionCalculators), are invoked.

10.4.2. How the Pre-aggregated Results are Reported

If either of the geographic computational shortcuts is taken, the output database produced does not contain any “real” county (or perhaps even “real” state) level detail, even though such detail is generally present in the MOVESDefault and user input databases. Instead, additional “pseudo” values of stateID, countyID, etc. appear in the output records when a geographic computational shortcut is taken.

If any one of the time period calculation shortcuts is taken, there may be only single representative “hour”, “day” or “month” time periods for the MasterLoop to loop over, (though no MasterLoopable objects currently sign up below the Month level), and the output database produced may not contain any “real” hour, day, or month level detail, even though such detail will generally be present in the MOVESDefault and user input databases. Instead “pseudo” time period-identifying values will now be present in the MOVEExecution and MOVESOutput databases.

10.4.3. Algorithms Used to Perform the NATION and STATE Pre-Aggregations

Table 10-1 describes the database aggregation algorithms used on a table-by-table basis for the NATION and STATE cases. Tables not listed contain no geographic identifiers and are therefore not affected by these aggregations. While some of these table aggregations are simple summations, others are “activity-weighted”. For these activity-weighted summations to be performed entirely correctly, something approaching the full execution of the control strategies and generators would have to be performed which would defeat the purpose of the pre-aggregation. So, instead, these “activity-weighted” aggregations involve compromise and simplification. Specifically, the activity weighting is based entirely upon “startAllocFactor” values in the Zone table. The variable startAllocFactor is the factor used within MOVES to allocate the total number of starts from the national to the county / zone level. In the default MOVES database for DRAFT MOVES2009 this allocation is based on vehicle miles traveled (VMT), hence the use of startAllocFactor for the pre-aggregation weightings is in essence a VMT weighting. More details on startAllocFactor and its derivation can be found in the report “MOVES2004 Highway Vehicle Population and Activity Data”.

Table 10-1. Database Aggregation Algorithms

MOVES Database Table	**GeographicSelectionType = NATION**	**GeographicSelectionType = STATE**
State	Single Record, stateID=0, stateName=Nation, stateAbbr=US	No action required. Already filtered by stateID
County	Single Record, countyID = 0. stateID=0, countyName=Nation, altitude=L barometric pressure and GPAFract are activity-weighted.	Single record per stateID, countyID=stateID*1000, countyName = stateName, altitude = L barometric pressure and GPAFract are activity-weighted.
Zone	Single Record, zoneID=0, countyID=0., startAllocFactor = 1.0 idleAllocFactor = 1.0 SHPAllocFactor = 1.0	Single record per stateID, zoneID=stateID*10000, countyID= stateID*1000 startAllocFactor = sum of old factors for state. Same for idleAllocFactor and SHPAllocFactor

Link	Single record for each roadTypeID in RunSpec.
	linkID=roadTypeID,
	countyID = 0.
	zoneID = 0.
	linkLength = NULL
	linkVolume = NULL
	grade = weighted national average
		Single record for each stateID - roadTypeID in RunSpec.
		linkID= stateID * 100000 + roadTypeID,
		countyID = stateID*1000,
		zoneID = stateID*10000
		linkLength = NULL
		linkVolume = NULL
		grade = weighted state average

CountyYear	Single record for each yearID, countyID = 0	Single record per stateID - yearID combination. countyID = stateID*1000.

ZoneMonthHour	Single record for each monthID-hourID combination in old table. (These have been filtered.) Calculate activity-weighted national average temperature and relative humidity. heatIndex and specific humidity are recalculated by Met generator.	Single record for each combination of new zoneID, month and hour. (These have been filtered.) Calculate activity-weighted state average temperature and relative humidity. heatIndex and specific humidity are recalculated by Met generator.
OpMode Distribution	(contents would be from user input.) Aggregate to national level roadType linkIDs, weighting by activity.	(contents would be from user input.) Aggregate to state level roadType linkIDs, weighting by activity.
ZoneRoadtype	Single record for each roadTypeID. SHOAllocFactor = sum of old SHOAllocFactors	Single record for each new zoneID, roadTypeID combination. SHOAllocFactor = sum of old SHOAllocFactors
FuelSupply (Default values are considered.)	Activity-weighted aggregation of all old counties to single new county. Since the fuel supply is in terms of fuel formulations, there may be many fuel formulations in the aggregate.	Activity-weighted aggregation of all old counties in state to single new county. Since the fuel supply is in terms of fuel formulations, there may be many fuel formulations in the aggregate.
IMCoverage	The NATION is considered to have IMCoverage with unknown (NULL) inspection frequencies, activity-weighted average IMAdjustFract values, and model year ranges which range from the earliest to the latest present at any location, for a given year, polProcess,fueltype and regclass. This is obviously an approximation.	Each STATE is considered to have IMCoverage with unknown (NULL) inspection frequencies, activity-weighted average IMAdjustFract values, and model year ranges which range from the earliest to the latest present at any location in the state, for a given year, polProcess,fueltype and regclass. This is obviously an approximation.
SHO	(contents would be from user input.) Combine all links having same roadtype. SHO and distance are simple summations.	(contents would be from user input.) Combine all links within state having same roadtype. SHO and distance are simple summations.
SourceHours	(contents would be from user input.) Combine all zones. starts field is a simple summation.	(contents would be from user input.) Combine all zones within state. starts field is a simple summation.
Starts	(contents would be from user input.) Combine all zones. starts field is a simple summation.	(contents would be from user input.) Combine all zones within state. starts field is a simple summation.
Extended IdleHours	(contents would be from user input.) Combine all zones. extendedIdleHours field is a simple summation.	(contents would be from user input.) Combine all zones within state. extendedIdleHours field is a simple summation.

SCCRoadType Distribution	Combine all zones, SCCRoadTypeFractions are activity-weighted.	Combine all zones in state, SCCRoadTypeFractions are activity-weighted.
AverageTank Temperature	(contents would be from user input.) Combine all zones, averageTankTemperature values are activity-weighted.	(contents would be from user input.) Combine all zones in state, averageTankTemperature values are activity-weighted.
SoakActivity Fraction	(contents would be from user input.) Combine all zones, soakActivityFraction values are activity-weighted.	(contents would be from user input.) Combine all zones in state, soakActivityFraction values are activity-weighted.
ColdSoakTank Temperature	(contents would be from user input.) Combine all zones, coldSoakTankTemperature values are activity-weighted.	(contents would be from user input.) Combine all zones in state, coldSoakTankTemperature values are activity-weighted.
ColdSoakInitial HourFraction	(contents would be from user input.) Combine all zones, coldSoakInitialHourFraction values are activity-weighted.	(contents would be from user input.) Combine all zones in state, coldSoakInitialHourFraction values are activity-weighted.
AverageTank Gasoline	(contents would be from user input.) Combine all zones, ETOHVolume and RVP values are activity-weighted.	(contents would be from user input.) Combine all zones in state, ETOHVolume and RVP values are activity-weighted.

10.4.4. Algorithms Used to Perform the Time Period Pre-Aggregations

Table 10-2 describes the database aggregation algorithms used on a table-by-table basis for the DAY (Portion of the Week), MONTH, and YEAR time period pre-aggregations. These must operate correctly whether or not one of the STATE or NATION aggregations has been performed. The MONTH-level preaggregation assumes that the DAY level has been performed and the YEAR-level assumes that the MONTH level has been performed. Tables not listed contain no time period identifiers and are therefore not affected by these aggregations.

While some of these table aggregations are simple summations, others are "activity-weighted". All activity-based weighting is in essence based on VMT, using allocations of VMT at the level necessary for the desired aggregation. For these activity-weighted summations to be performed entirely correctly, something approaching the full execution of the control strategies and generators would have to be performed which would defeat the purpose of the pre-aggregation. So instead these “activity-weighted” aggregations involve some compromise and simplification. Specifically, the activity weighting used for the DAY aggregation is based upon the values in the HourVMTFraction table; the weighting used for the MONTH aggregation is based upon the values in the DayVMTFraction table; and the activity weighting used for the YEAR aggregation is based upon the values in the MonthVMTFraction table. Because these activity fractions themselves depend upon other dimensions of the model, which do not always appear in the tables being aggregated, several variations of each aggregation are utilized, some of which are approximations:

For aggregating hours into Days, three activity-weighting variations are used. (The third and fourth are approximations.)

HourWeighting1: is based directly on the HourVMTFraction table itself, which is used to aggregate tables sharing its sourceTypeID, roadTypeID, and dayID primary keys.

HourWeighting2: uses RoadTypeDistribution to aggregate HourVMTFraction over Roadtype. This is used to aggregate tables having sourceTypeID and dayID, but not roadTypeID.

HourWeighting3: is a simple weighting by hourID used to aggregate tables sharing no keys with HourVMTFraction except hourID. It is produced from HourWeighting2 by using the data for the passenger car source type, and giving equal weight to all portions of the week.

HourWeighting4: is produced from HourWeighting2 by using the data for the passenger car source type. This is used to aggregate tables sharing no keys with HourVMTFraction except dayID and hourID.

For aggregating days (periods of the week) into Months, three activity-weighting variations are needed. The third is an approximation.

DayWeighting1: is based on the DAYVMTFraction table, with MonthID removed by using weights from MonthVMTFraction. Its key fields are sourceTypeID, roadTypeID, and dayID.

DayWeighting2: is based on DayWeighting1, with roadTypeID removed by using information from RoadTypeDistribution. Its key fields are sourceTypeID and dayID.

DayWeighting3: is based on DayWeighting2, and uses the distribution for sourceTypeID = 21 for all sourceTypes.

For aggregating months into years, only one activity-weighting is needed, and it is an approximation:

MonthWeighting: based on MonthVMTFraction information for passenger cars only.

The analogous technique is used to aggregate month groups into years. (Monthgroups are used in some MOVES Database tables and were intended to represent seasons of the year. As the MOVES default database is currently populated, however, monthgroups correspond exactly to months.

Table 10-2. Description of Time Period Aggregations to Be Performed

MOVES Database Table	**Aggregation of Hours to DAY (Portion of the week)**	**Aggregation of Days (Portions of the week) to MONTH (assumes DAY aggregation.)**	**Aggregation of Months to YEAR (assumes MONTH aggregation.)**
HourOfAnyDay	Single Record, hourID=0, hourName = "Entire day".		
DayOfAnyWeek		Single record. dayID=0. dayName = "Whole Week"; noOfRealDays=7	
HourDay	Record for each dayID, hourDayID = dayID; hourID=0	Single record. dayID=0. hourID=0.	
MonthOfAny Year			Single record. MonthID = 0 noOfDays = 365 monthGroupID=0
MonthGroup OfAnyYear			Single record. MonthGroupID=0 monthGroupName = "Entire Year".
HourVMT Fraction	Record for each SourceType- RoadType-Day combination. HourVMTFraction = 1.0	Record for each SourceType- RoadType combination. HourVMTFraction = 1.0	
DayVMT Fraction		Record for each SourceType-Month-RoadType combination. DayVMTFraction = 1.0	Record for each SourceType-RoadType combination. DayVMTFraction = 1.0
MonthVMT Fraction			monthVMTFraction = 1.0
AvgSpeed Distribution	Activity-weighted average avgSpeedFraction using HourWeighting1	Activity-weighted average avgSpeedFraction using DayWeighting1	

OpMode Distribution (contents would be from user input.)	Activity-weighted average opModeFraction using HourWeighting1	Activity-weighted average opModeFraction using DayWeighting1	
SourceType Hour	Hour-level idleSHOFactors are summed	Activity-weighted average idleSHOFactor using DayWeighting2	
SHO (contents would be from user input.)	Simple summation of SHO and distance	Simple summation of SHO and distance	Simple summation of SHO and distance
SourceHours	Simple summation of sourceHours	Simple summation of sourceHours	Simple summation of sourceHours
Starts (contents would be from user input.)	Simple summation of starts	Simple summation of starts	Simple summation of starts
Extended IdleHours (contents would be from user input.)	Simple summation of extendedIdleHours	Simple summation of extendedIdleHours	Simple summation of extendedIdleHours
ZoneMonthHour	Activity-weighted average temperature and relHumidity using HourWeighting3. MetGenerator will recalculate heatIndex and specific humidity.		Activity-weighted average temperature and relHumidity using MonthWeighting. MetGenerator will recalculate heatIndex and specific humidity.
MonthGroup Hour	Activity-weighted ACActivity terms using HourWeighting3.		Activity-weighted ACActivity terms using MonthGroupWeighting.
SampleVehicleTrip	Set hourID = 0 for all trips	Set dayID = 0 for all trips. Assumes vehIDs are unique across dayIDs.	
SampleVehicleDay		Set dayID = 0 for all vehicle days. Assumes vehIDs are unique across dayIDs.	
StartsPerVehicle (contents would be from user input.)	hourDayID = dayID Simple summation of startsPerVehicle	hourDayID = 0. Simple summation of startsPerVehicle	

FuelSupply			Activity-weighted marketshare using MonthGroupWeighting. (Note: default values produced by DefaultDataMaker, must be considered.)
AverageTank Temperature (contents would be from user input.)	Activity weighted averageTankTemperature using HourWeighting4.	Activity weighted averageTankTemperature using DayWeighting3.	Activity weighted averageTankTemperature using MonthWeighting
SoakActivity Fraction (contents would be from user input.)	Activity-weighted soakActivityFraction using HourWeighting2	Activity weighted average soakActivityFraction using DayWeighting2.	Activity weighted average soakActivityFraction using MonthWeighting

10.4.5. Calculation Inaccuracies Introduced by the Database Pre-Aggregations:

The simplified activity-weighted aggregations introduce the following approximations relative to having MOVES perform its calculations individually for each real county-location and for each hour of the day:

- Start-based activity, while appropriate for the start process, represents an approximation for other processes whose activity basis (SHO, etc.) may not exactly correspond to start activity.

- Direct user input to the CMIT tables may override the Zone.startAllocFactor values. Any effect on activity of direct user input to the CMIT tables is not taken into account.

- Control strategies may eventually be added to MOVES which adjust activity levels. Any such effects are not included.

- The potentially significant non-linear relationships of the emissions calculations to temperature and humidity are ignored. This may be especially serious at the national level.

- Activity weighted hourly averages are used when combining hourly temperature, humidity, AC activity information, and any user supplied average tank temperature values for the hours of the day, but differences in hourly activity levels between the passenger car source use type and other source use types are ignored in this calculation, as are differences in hourly activity levels by day of the week.

- The distribution of passenger car VMT to the periods of the week is used for all source types when aggregating any user-supplied average tank temperature data to the month level.

- Differences in monthly activity patterns between passenger cars and other source use types are ignored when calculating weighted average annual temperature, humidity, ACActivity, fuelSubtype marketshares, average fuel tank temperatures and soakactivityfractions.

- When calculating annual data from monthly data, all years are considered to be non-leap years.

An initial analysis of the sensitivity of MOVES results to levels of pre-aggregation was presented in the report “MOVES2004 Validation Results”. While DRAFT MOVES2009 did produce different results depending on the level of aggregation selected by the user, the magnitude of difference for energy consumption did not appear to be very large. For example, the difference in total energy results between a run where state / month pre-aggregation was selected was about 2 percent higher than the same run where nation / year pre-aggregation was selected.

10.5. Mesoscale Lookup Table Link Producer (LTLP)

This component is invoked when executing runs which specify the “Mesoscale Lookup” Scale. It reconstructs the contents of the Link table and populates the LinkAverageSpeed table based on the contents of the AverageSpeedBin and Zone tables. It is invoked early during the run execution after the InputDataManager has constructed the MOVESExecution database. It simplifies this situation that geographic preaggregation is not allowed in conjunction with Mesoscale Lookup. Within the MOVES program code it is implemented within a component called the GeographicExecutionLocationProducer which produces the list of locations looped over by the Master Looping mechanism.

The current default “Link” table has a record for each combination of county and road type. The LTLP populates the Link table with unique links for every combination of averageSpeedBinID value in the AverageSpeedBin table, each zoneID in the RunSpec, and each roadTypeID in the RunSpec.

a. The linkID is generated to be a unique value. (Current macroscale linkID * 100 + averageSpeedBinID.)
b. The zoneID and roadTypeID fields are populated appropriately.
c. The countyID field is populated based on the zoneID.
d. The linkLength, linkVolume and grade fields are not used and are set to Null.

The LTLP also populates the LinkAverageSpeed table with a single record for each of the new linkIDs. The average speed value used is the avgBinSpeed value from the AverageSpeedBin table for the AvgSpeedBin represented by the Link.

10.6. Total Activity Generator (TAG) for Macroscale

This generator calculates total activity pursuant to the run specification for each source use type in DRAFT MOVES2009 using the MOVESExecution database. This activity includes Start activity (number of starts), Extended Idle Hours, and Source Hours in addition to source hours operating (SHO). This activity information is categorized by time span, geographic location, source type and age. The product of the TAG is four core model input tables produced in MOVESExecution containing these results: SHO, Starts, ExtendedIdleHours, and SourceHours. The TAG also calculates an intermediate form of distance traveled information which is stored in the SHO table. This version of the TAG is not used for the Mesoscale Lookup scale.

The algorithm used to calculate total activity for a given time and location by source type and age is divided into ten steps (numbered 0 thru 9) referred to as TAGs (total activity generator steps) in this document. Each TAG step references the MOVESExecution database and implements a simple mathematical formula. The primary function of the TotalActivityGenerator is to convert commonly-available activity data such as vehicle miles traveled (VMT), age distribution, vehicle populations, sales and VMT growth rates, etc. to the MOVES activity parameters of source hours, source hours operating, starts, and extended idle hours. Externally-provided VMT is thus the primary driver of total activity. Steps 1-3 calculate allocations of total VMT by vehicle age and source type. Step 4 grows VMT. Steps 5 and 6 allocate VMT by time, space, vehicle age and source type. Step 7 converts the allocated VMT to the MOVES activity parameters of source hours operating (SHO), number of starts, and extended idle hours. Step 8 allocates activity to zones (e.g. counties for the default case), and Step 9 re-calculates distance for output reporting purposes. A brief description of each of the TAGs is shown in table 10-3. The HPMS acronym used in the table refers to the Highway Performance Management System.

Table 10-3. Overview of TAG Calculations

Step		**Description**
TAG-0	Determine the base year	Determine the appropriate base year to use in

		growth calculations
TAG-1	Calculate Base Year Vehicle Population	Establish base-year vehicle population in modeling domain by use type, age
TAG-2	Grow Vehicle Population to Analysis Year	If analysis year is in future, establish analysis year population through growth and scrappage
2a	*Age 0*	
2b	*Age 1 through 29*	
2c	*Age 30 and higher*	
TAG-3	Calculate Analysis Year Travel Fraction	Calculation travel fraction across domain as function of age mix and annual miles traveled, by use type and age
3a	*Calculate total population within HPMS vehicle class*	
3b	*Calculate fraction within HPMS vehicle class*	
3c	*Compute travel fraction*	
TAG-4	Calculate Analysis Year VMT	If analysis year is in future, establish analysis year aggregate VMT across domain from base year VMT and growth
TAG-5	Allocate Analysis Year VMT By Roadway Type, Use Type, Age	Allocate aggregate VMT to roadway type, use type and age
TAG-6	Allocate Annual VMT to Hour by Roadway Type, Use Type, Age	Allocate annual VMT to hourly VMT
TAG-7	Convert to Total Activity Basis By Process	Convert to total activity basis at domain level: SHO, Starts, Extended Idle Hours, Source Hours Parked (a portion of Source Hours)
7a	*Calculate average speed*	
7b	*Convert to SHO*	
7c	*Calculate Starts*	
7d	*Convert SHO to Exetended Idle Hours*	
7e	*Calculate Source Hours Parked (SHP)*	
TAG-8	Allocate Total Activity to Location	Allocate total activity to locations using geographic allocation factors
8a	*Allocate SHO to Links*	
8b	*Allocate Starts to Zones*	
8c	*Allocate Extended Idle Hours to Zones*	
8d	*Allocate SHP to Zones*	
8e	*Calculate Source Hours by Zone and Roadway*	
TAG-9	Calculate Distance Traveled	Calculate SHO.distance from SHO and average speeds

Some overall considerations when performing these calculations are:

1. The TotalActivityGenerator signs up for the Master Loop at the Year level which means the calculations are performed individually for each year at each location (i.e. link) for each emission process requiring SHO, start, extended idle hour, or source hour activity information.

2. The MOVES design allows the user to provide some or all of the values in core model input tables such as SHO, Starts, ExtendedIdleHours and SourceHours. The InputDataManager places any such user-supplied values in the MOVESExecution database before the TotalActivityGenerator is activated. The TotalActivityGenerator does not replace such user-supplied values.

3. When the Total Activity Generator encounters a missing value when performing a calculation, the result of the calculation is considered as missing. Records for which the results are missing are not represented by a value of zero but are left out of the database.

Detailed descriptions of the calculations in each TAG step follow. Each of the variables used in the TAG calculations either exists in the MOVESExecution database or is calculated by a previous TAG step. All of the database variables are described in the database documentation included in the database. The table in which each variable can be found is indicated in parentheses in the “Input Variables”portion of each TAG step description.

10.6.1. TAG-0: Determine the Base Year

Before any calculations can be done, the appropriate base year must be determined using the year of analysis and the isBaseYear information in the Year table.

Input Variables:

isBaseYear (Year)
calendar year (from RunSpec)

Output Variable:

baseYear

Calculation:

```
yearID = calendar year

IF isBaseYear(yearID) = "Y", then baseYear=yearID.

ELSE
```

baseYear = maximum value of yearID which is less than the calendar year and
for which isBase(Year)= "Y"

10.6.2. TAG-1: Calculate Base Year Vehicle Population By Age.

Input Variables:

sourceTypePopulation (SourceTypeYear)
ageFraction (SourceTypeAgeDistribution)
baseYear (from previous step)

Output Variable:

sourceTypeAgePopulation, (in new SourceTypeAgePopulation table)

Calculation:

yearID=baseYear
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) =
sourceTypePopulation (yearID, sourceTypeID) *
ageFraction (yearID, sourceTypeID, ageID)

10.6.3. TAG-2: Grow Vehicle Population from Base Year to Analysis Year

NOTE: This step is only required if the analysis year is in the future relative to the base year. For future projection, these TAG-2 steps are repeated in every year until the analysis year is reached.

TAG-2a: Age Zero (New Vehicles)

This calculation applies only to ageID=0.

Input Variables:

sourceTypeAgePopulation, from previous step,
for ageID=0 and yearID=yearID-1
salesGrowthFactor (SourceTypeYear)
migrationRate (for yearID and yearID-1) (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=0 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=0) =
[sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=0) /
migrationRate (yearID-1, sourceTypeID)] *
salesGrowthFactor (yearID, sourceTypeID) *
migrationRate (yearID, sourceTypeID)

Note: the full equation would divide through by age=0 survival rate to derive new sales in previous years prior to scrappage, and multiply by the same term to account for scrappage in the first year. Since scrappage is the same in all years, they cancel each other out, and these terms are excluded from the equation.

TAG-2b: Ages 1 through 29

This calculation loops through AgeID=x, where x is 1 through 29.

Input Variables:

sourceTypeAgePopulation (for ageID=x and yearID=yearID-1)
survivalRate (for AgeID=x) (SourceTypeAge)
migrationRate (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=1 through 29 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=x) =
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=x-1) *
survivalRate (sourceTypeID, ageID=x-1) *
migrationRate (yearID, sourceTypeID)

TAG-2c: Age 30

This calculation is for Age 30, which includes years 30 and higher.

Input Variables:

sourceTypeAgePopulation (for ageID=29 & 30 and yearID=yearID-1)
survivalRate (for ageID=29 & 30) (SourceTypeAge)
migrationRate (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=30 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=30) =
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=29) *
survivalRate (sourceTypeID, ageID=29) *
migrationRate (yearID, sourceTypeID) +
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=30) *
survivalRate (sourceTypeID, ageID=30) *
migrationRate (yearID, sourceTypeID)

10.6.4. TAG-3: Calculate Analysis Year Travel Fraction

TAG-3a: Calculate total population within Highway Performance Management System (HPMS) vehicle class.

Input Variables:

sourceTypePopulation (for each sourceTypeID within HPMSVtypeID)
(SourceTypeYear)
HPMSVtypeID (HPMSVtype)

Output Variable:

HPMSVtypePopulation

Calculation:

HPMSVTypePopulation (yearID, HPMSVtypeID) =
Sum of [SourceTypeAgePopulation (yearID, sourceTypeID, ageID)]
over all ageID,
for all sourceTypeID within each HPMSVtypeID.

TAG-3b: Calculate fraction within HPMS vehicle class

Input Variables:

sourceTypeAgePopulation (from step 2)
HPMSVtypePopulation (from previous step)

Output Variable:

fractionwithinHPMSVtype

Calculation:

fractionwithinHPMSVtype (yearID, sourceTypeID, ageID) =
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) /
HPMSVTypePopulation (yearID, HPMSVtypeID, sourceTypeID)

TAG-3c: Compute travel fraction

Input Variables:

fractionwithinHPMSVtype (from previous step)
relativeMAR (SourceTypeAge)

Output Variable:

travelFraction

Calculation:

travelFraction (yearID, sourceTypeID, ageID) =
(fractionwithinHPMSVtype (yearID, sourceTypeID, ageID=x) *
relativeMAR (sourceTypeID, ageID=x)) /
Sum of [fractionwithinHPMSVtype (yearID, sourceTypeID, ageID=x) *
relativeMAR (sourceTypeID, ageID=x)] over all ageID and for all
sourceTypeID within each HPMSVtype.

10.6.5. TAG-4: Calculate Analysis Year VMT

Determine the total VMT within each HPMS vehicle type, accounting for VMT growth. This calculation is repeated in every yearID until the analysis year is reached.

TAG-4a: Base year.

In the base year, the analysisYearVMT is the same as the HPMSBaseYearVMT.

Input Variables:

HPMSBaseYearVMT (yearID=BaseYear) (HPMSVtypeYear)

Output Variable:

analysisYearVMT

Calculation:

analysisYearVMT (yearID, HPMSVTypeID) =
HPMSBaseYearVMT (yearID, HPMSVTypeID)

TAG-4b: All years following the base year.

In the any years following the base year, the analysisYearVMT is calculated from the previous year's value for analysisYearVMT.

Input Variables:

analysisYearVMT (yearID-1)
VMTGrowthFactor (HPMSVtypeYear)

Output Variable:

analysisYearVMT

Calculation:

analysisYearVMT (yearID, HPMSVTypeID) =
analysisYearVMT (yearID-1, HPMSVTypeID) *
VMTGrowthFactor (yearID, HPMSVTypeID)

10.6.6. TAG-5: Allocate Analysis Year VMT by Roadway Type, Use Type, Age

Input Variables:

analysisYearVMT (from previous step)
roadTypeVMTFraction (RoadTypeDistribution)
travelFraction (from step 3)

Output Variable:

annualVMTbyAgeRoadway

Calculation:

annualVMTbyAgeRoadway (yearID, roadTypeID, sourceTypeID, ageID) =
analysisYearVMT (yearID, HPMSVTypeID) *
roadTypeVMTFraction (roadTypeID, sourceTypeID) *
travelFraction (yearID, sourceTypeID, ageID)

10.6.7. TAG-6: Allocate Annual VMT to Hour

Input Variables:

annualVMTbyAgeRoadway (from previous step)
monthVMTFraction (MonthVMTFraction)
dayVMTFraction (DayVMTFraction)
hourVMTFraction (HourVMTFraction)
noOfDays (MonthOfAnyYear)

Output Variable:

VMTbyAgeRoadwayHour

Calculation:

VMTbyAgeRoadwayHour (yearID, roadTypeID, sourceTypeID, ageID,
monthID, dayID, hourID) =
annualVMTbyAgeRoadway (yearID, roadTypeID, sourceTypeID, ageID) *
monthVMTFraction (sourceTypeID, isLeapYear, monthID) *
dayVMTFraction (roadTypeID, sourceTypeID, monthID, dayID) *
hourVMTFraction (roadTypeID, sourceTypeID, dayID, hourID)
/ (noOfDays/7)

10.6.8. TAG-7: Convert to Total Activity Basis

Tag-7a: Compute average speed by roadway type

Input Variables:

avgBinSpeed (AvgSpeedBin)
averageSpeedFraction (AverageSpeedDistribution)

Output Variable:

averageSpeed

Calculation:

averageSpeed (roadTypeID, sourceTypeID, dayID, hourID) =

Sum of [avgBinSpeed(avgSpeedBin) * averageSpeedFraction(roadTypeID, sourceTypeID, dayID, hourID, avgSpeedBin)] over all avgSpeedBin

Tag-7b: Convert VMT to SHO

Input Variables:

VMTbyAgeRoadwayHour (from step 6)
averageSpeed (from previous step)

Output Variable:

SHObyAgeRoadwayHour

Calculation:

SHObyAgeRoadwayHour (yearID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) =
VMTbyAgeRoadwayHour(yearID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) /
averageSpeed (roadTypeID, sourceTypeID, dayID, hourID)

Tag-7c: Calculate Starts

Preliminary Calculation

For each combination of sourceTypeID, hourID, and dayID

startsPerVehicle = (number of records in INNER JOIN of SampleVehicleDay and SampleVehicleTrip with non null keyOnTimes)
/
(number of records in SampleVehicleDay)

This calculation excludes “Marker Trips”.

Input Variables:

startsPerVehicle (sourceTypeID, hourID, dayID)
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) (from step 2)

Output Variable:

startsByAgeHour

Calculation:

startsByAgeHour (yearID, sourceTypeID, ageID, dayID, hourID) =
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) *
startsPerVehicle (sourceTypeID, hourID, dayID)

Tag-7d: Convert SHO to Extended Idle Hours

Input Variables:

SHObyAgeRoadwayHour (from step 7b)
idleSHOFactor (SourceTypeHour)

Output Variable:

idleHoursbyAgeHour

Calculation:

idleHoursbyAgeHour (yearID, sourceTypeID, ageID, monthID, dayID, HourID)
=
(Sum over 24 hours and over all Roadtypes (SHObyAgeRoadwayHour(yearID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID))) *
idleSHOFactor(sourceType, dayID, hourID)

Tag-7e: Calculate Source Hours Parked (SHP)

Input Variables:

SHObyAgeRoadwayHour (yearID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) (from step 7b)
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) (from step 2)
noOfRealDays (DayOfAnyWeek)

Output Variable:

SHPbyAgeHour (yearID, sourceTypeID, ageID, monthID, dayID, hourID)

Calculation:

SHPbyAgeHour = sourceTypeAgePopulation * noOfRealDays-
Sum over roadtypes(SHObyAgeRoadwayHour)

10.6.9. TAG-8: Allocate Total Activity to Location

Tag-8a: Allocate SHO to Links

Input Variables:

SHOByAgeRoadwayHour (from step 7b)
SHOAllocationFactor (ZoneRoadwayType)

Output Variable:

SHO (SHO)
This result populates the SHO field in SHO table.

Calculation:

SHO(yearID, zoneID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) =
SHOByAgeRoadwayHour(yearID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) *
SHOAllocationFactor(zoneID, roadTypeID)

Tag-8b: Allocate Starts to Zones

Input Variables:

startsByAgeHour (from step 7c)

startAllocFactor (Zone)

Output Variable:

Starts (Starts)
Result of TAG-8b populates “starts” field of Starts Table

Calculation:

starts(yearID, zoneID, sourceTypeID, ageID, monthID, dayID, hourID) =
startsByAgeHour(yearID, sourceTypeID, ageID, dayID, hourID) *
startAllocFactor(zoneID))

Tag-8c: Allocate Extended Idle Hours to Zones

Input Variables:

idleHoursByAgeHour (from step 7d)
idleAllocFactor (Zone)

Output Variable:

extendedIdleHours(ExtendedIdleHours)
Result of TAG-8c populates “extendedIdleHours” field of ExtendedIdleHours Table

Calculation:

extendedIdleHours(yearID, zoneID, sourceTypeID, ageID, monthID, dayID, hourID) =
idleHoursByAgeHour(yearID, sourceTypeID, ageID, monthID, dayID, hourID) * idleAllocFactor(zoneID)

Tag-8d: Allocate SHP to Zones

Input Variables:

SHPByAgeHour (from step 7e)
SHPAllocFactor (Zone)

Output Variable:

SHP (in an intermediate table)

Calculation:

SHP(yearID, zoneID, sourceTypeID, ageID, monthID, dayID, hourID) =
SHPByAgeHour(yearID, sourceTypeID, ageID, monthID, dayID, hourID) *
SHPAllocFactor(yearID, zoneID)

Tag-8e: Calculate Source Hours by Zone and Roadway

Input Variables:

SHP (from step 8d)
SHO (from step 8a)

Output Variable:

sourceHours(SourceHours) result of this step populates sourceHours field in SourceHours table

Calculation:

sourceHours(yearID, zoneID, roadTypeID <> OffNetwork, sourceTypeID, ageID, monthID, dayID, hourID) = SHO(yearID, zoneID, roadTypeID, sourceTypeID, ageID, monthID, dayID, hourID) sourceHours(yearID, zoneID, roadTypeID = OffNetwork, sourceTypeID, ageID, monthID, dayID, hourID) = SHP(yearID, zoneID, sourceTypeID, ageID, monthID, dayID, hourID)

10.6.10. TAG-9: Calculate Distance Traveled Corresponding to Source Hours Operating

If the “Distance Traveled” output is requested by the RunSpec, (which also implies that some pollutant has been selected for the Running process), the distance field in the SHO table is calculated. Otherwise this distance field is output by this generator as NULL. The method used to produce this distance information involves multiplying the number of source hours operating (SHO) by the average vehicle speed associated with them. These average speeds have been calculated in Step 7a.

Input Variables:

SHO from step 8a
averageSpeed from step 7a

Output Variable:

distance (in SHO table)

Calculation:

distance (yearID, monthID, linkID (zoneID with roadTypeID), hourID, dayID, ageID, aourceTypeID) =
SHO((yearID, monthID, linkID (zoneID with roadTypeID), hourID, dayID, ageID, sourceTypeID) * averageSpeed(roadTypeID, sourceTypeID, dayID, hourID)

10.7. Total Activity Generator (TAG) for Mesoscale Lookup

This version of the TAG is used for the Mesoscale Lookup scale and is a simplified version of the “Macroscale” TAG described in the preceding section. The TAG can be simplified for Mesoscale lookup calculations because only running activity is needed and because artificial VMT values can be used. Because emissions are reported in grams/mile or other “per distance” units, the absolute values of VMT and SHO don’t matter for this use case, but their proportional allocations among sourcetypes, ages, and time periods must be maintained.

This generator calculates total activity pursuant to the run specification for each source use type in DRAFT MOVES2009 using the MOVESExecution database. This activity includes Source Hours in addition to source hours operating (SHO). This activity information is categorized by time span, geographic location, source type and age. This TAG produces two core model input tables in MOVESExecution containing these results: SHO, and SourceHours. The TAG also calculates an intermediate form of distance traveled information which is stored in the SHO table.

Some overall considerations when performing these calculations are:

1. The TotalActivityGenerator signs up for the Master Loop at the Year level which means the calculations are performed individually for each year at each location (i.e. link) for each emission process requiring SHO or source hour activity information.

2. The MOVES design allows the user to provide some or all of the values in core model input tables such as SHO, and SourceHours. The InputDataManager places any such user-supplied values in the MOVESExecution database before the TotalActivityGenerator is activated. The TotalActivityGenerator does not replace such user-supplied values.

3. When the Total Activity Generator encounters a missing value when performing a calculation, the result of the calculation is considered as missing. Records for which the results are missing are left out of the database.

The steps for the Mesoscale Lookup TAG are summarized here and described in more detail below.

1) Travel fractions by Age & SourceType are determined using population growth and relative mileage accumulations.
2) Travel fractions are treated as VMT and allocated to specific months, days & hours using factors that vary by roadType & sourceType.
3) SHO is set to equal the hourly VMT. The same SHO is used for each link of a roadtype.
4) Source Hours are set equal to SHO.
5) Distance is calculated from the SHO and the link speed.

By way of general background information, a number of database fields that are important for the Macroscale TAG are not used in Mesoscale Lookup:

HPMSBaseYearVMT
VMTGrowthFactor
roadTypeVMTFraction
averageSpeedFraction
idleSHOFactor
SHOAllocationFactor
startAllocFactor
idleAllocFactor

Detailed descriptions of the calculations in each TAG step follow. Each of the variables used in the TAG calculations either exists in the MOVESExecution database or is calculated by a previous TAG step. All of the database variables are described in the database documentation included in the database. The table in which each variable can be found is indicated in parentheses in the "Input Variables" portion of each TAG step description.

10.7.1. TAG-0: Determine the Base Year

Before any calculations can be done, the appropriate base year must be determined using the year of analysis and the isBaseYear information in the Year table.

Input Variables:

isBaseYear (Year)
calendar year (from RunSpec)

Output Variable:

baseYear

Calculation:

```
yearID = calendar year

IF isBaseYear(yearID) = "Y", then baseYear=yearID.

ELSE

baseYear = maximum value of yearID which is less than the calendar year for
    which  isBase(Year)= "Y”
```

10.7.2. TAG-1: Calculate Base Year Vehicle Population By Age.

Input Variables:

sourceTypePopulation (SourceTypeYear)
ageFraction (SourceTypeAgeDistribution)
baseYear (from previous step)

Output Variable:

sourceTypeAgePopulation, (in new SourceTypeAgePopulation table)

Calculation:

```
yearID=baseYear
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) =
sourceTypePopulation (yearID, sourceTypeID)  *
ageFraction (yearID, sourceTypeID, ageID)
```

10.7.3. TAG-2: Grow Vehicle Population from Base Year to Analysis Year

NOTE: This step is only required if the analysis year is in the future relative to the base year. For future projection, these TAG-2 steps are repeated in every year until the analysis year is reached.

TAG-2a: Age Zero (New Vehicles)

This calculation applies only to ageID=0.

Input Variables:

sourceTypeAgePopulation, from previous step,
    for ageID=0 and yearID=yearID-1
salesGrowthFactor (SourceTypeYear)
migrationRate (for yearID and yearID-1) (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=0 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=0) =
[sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=0) /
migrationRate (yearID-1, sourceTypeID)] *
salesGrowthFactor (yearID, sourceTypeID) *
migrationRate (yearID, sourceTypeID)

Note: the full equation would divide through by age=0 survival rate to derive new sales in previous years prior to scrappage, and multiply by the same term to account for scrappage in the first year. Since scrappage is the same in all years, they cancel each other out, and these terms are excluded from the equation.

TAG-2b: Ages 1 through 29

This calculation loops through AgeID=x, where x is 1 through 29.

Input Variables:

sourceTypeAgePopulation (for ageID=x and yearID=yearID-1)
survivalRate (for AgeID=x) (SourceTypeAge)
migrationRate (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=1 through 29 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=x) =
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=x-1) *
survivalRate (sourceTypeID, ageID=x-1) *
migrationRate (yearID, sourceTypeID)

TAG-2c: Age 30

This calculation is for Age 30, which includes years 30 and higher.

Input Variables:

sourceTypeAgePopulation (for ageID=29 & 30 and yearID=yearID-1)
survivalRate (for ageID=29 & 30) (SourceTypeAge)
migrationRate (SourceTypeYear)

Output Variable:

sourceTypeAgePopulation (for ageID=30 in current yearID)

Calculation:

sourceTypeAgePopulation (yearID, sourceTypeID, ageID=30) =
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=29) *
survivalRate (sourceTypeID, ageID=29) *
migrationRate (yearID, sourceTypeID) +
sourceTypeAgePopulation (yearID-1, sourceTypeID, ageID=30) *
survivalRate (sourceTypeID, ageID=30) *
migrationRate (yearID, sourceTypeID)

10.7.4. TAG-3: Calculate Analysis Year Travel Fraction

TAG-3a: Calculate total population within Highway Performance Management System (HPMS) vehicle class.

Input Variables:

sourceTypePopulation (for each sourceTypeID within HPMSVtypeID)
(SourceTypeYear)
HPMSVtypeID (HPMSVtype)

Output Variable:

HPMSVtypePopulation

Calculation:

HPMSVTypePopulation (yearID, HPMSVtypeID) =
Sum of [SourceTypeAgePopulation (yearID, sourceTypeID, ageID)]
over all ageID,
for all sourceTypeID within each HPMSVtypeID.

TAG-3b: Calculate fraction within HPMS vehicle class

Input Variables:

sourceTypeAgePopulation (from step 2)
HPMSVtypePopulation (from previous step)

Output Variable:

fractionwithinHPMSVtype

Calculation:

fractionwithinHPMSVtype (yearID, sourceTypeID, ageID) =
sourceTypeAgePopulation (yearID, sourceTypeID, ageID) /
HPMSVTypePopulation (yearID, HPMSVtypeID, sourceTypeID)

TAG-3c: Compute travel fraction

Input Variables:

fractionwithinHPMSVtype (from previous step)
relativeMAR (SourceTypeAge)

Output Variable:

travelFraction

Calculation:

travelFraction (yearID, sourceTypeID, ageID) =
(fractionwithinHPMSVtype (yearID, sourceTypeID, ageID=x) *
relativeMAR (sourceTypeID, ageID=x)) /
Sum of [fractionwithinHPMSVtype (yearID, sourceTypeID, ageID=x) *
relativeMAR (sourceTypeID, ageID=x)] over all ageID and for all
sourceTypeID within each HPMSVtype.

10.7.5. TAG-4: (reserved)

10.7.6. TAG-5: Allocate Analysis Year VMT by Roadway Type, Use Type, Age

Because VMT is divided out at the end, real VMT is not needed and is removed from this calculation. However, the model does need the distribution of VMT among sourcetypes & ages. RoadType distinctions are needed for the next step.

Input Variables: TravelFraction(yearID, sourceTypeID, ageID)

Output Variable: AnnualVMTbyAgeRoadway

Calculation:

AnnualVMTby AgeRoadway(YearID, roadTypeID, sourceTypeID, AgeID) =
TravelFraction(YearID, sourceTypeID, AgeID)

10.7.7. TAG-6: Temporally Allocate Annual VMT to Hour

Input Variables:

annualVMTbyAgeRoadway (from previous step)
monthVMTFraction (MonthVMTFraction)
dayVMTFraction (DayVMTFraction)
hourVMTFraction (HourVMTFraction)
noOfDays (MonthOfAnyYear)

Output Variable:

VMTbyAgeRoadwayHour

Calculation:

VMTbyAgeRoadwayHour (yearID, roadTypeID, sourceTypeID, ageID,
monthID, dayID, hourID) =
annualVMTbyAgeRoadway (yearID, roadTypeID, sourceTypeID, ageID) *
monthVMTFraction (sourceTypeID, isLeapYear, monthID) *
dayVMTFraction (roadTypeID, sourceTypeID, monthID, dayID) *
hourVMTFraction (roadTypeID, sourceTypeID, dayID, hourID)
/ (noOfDays/7)

10.7.8. TAG-7: Convert to Total Activity Basis

Tag-7: Convert VMT to SHO

Because "Distance" is calculated from SHO and is divided out in the end, the actual SHO doesn't matter. But the proportional distribution of SHO among ages, sourcetypes and times must be preserved. This step sets SHO = VMT.

Input Variables:

VMTbyAgeRoadwayHour (from step 6)

Output Variable:

SHObyAgeRoadwayHour

Calculation:

SHObyAgeRoadwayHour (yearID, roadTypeID, sourceTypeID, ageID,
monthID, dayID, hourID) =
VMTbyAgeRoadwayHour(yearID, roadTypeID, sourceTypeID, ageID,
monthID, dayID, hourID)

10.7.9. TAG-8: Allocate Total Activity to Location

Tag-8a: Allocate SHO to Links

This step assigns SHO to multiple links of the same zone/roadtype. Because geographic aggregation is forbidden for Lookup Output, allocation to Zones can be uniform. Similarly, allocation to links is not important as long as the distribution among ages, sourcetypes & times is preserved. So we set SHO(link) = SHO(roadType).

Input Variables:

SHOByAgeRoadwayHour (from step 7)
Link(linkID, countyID, zoneID, roadTypeID) (Created by the LTLP)

Output Variable:

SHO (SHO)
This result populates the SHO field in SHO CMIT table.

Calculation:

SHO(yearID, linkID, sourceTypeID, ageID, monthID, dayID, hourID) =
SHOByAgeRoadwayHour(yearID, roadTypeID, sourceTypeID, ageID,
monthID, dayID, hourID)

Tag-8b: Calculate Source Hours by Zone and Roadway

Input Variables:

SHO (from step 8a)
LinkAverageSpeed(linkID)

Output Variable:

sourceHours(SourceHours) result of this step populates sourceHours field in
SourceHours table

Calculation:

sourceHours(yearID, linkID, sourceTypeID, ageID, monthID, dayID, hourID) =
SHO(yearID, linkID, sourceTypeID, ageID, monthID, dayID, hourID)

10.7.10. TAG-9: Calculate Distance Traveled Corresponding to Source Hours Operating

The method used to produce this distance information involves multiplying the number of source hours operating (SHO) by the average vehicle speed associated with the Link.

Input Variables:

SHO from step 8a
averageSpeed from step 7a

Output Variable:

distance (in SHO table)

Calculation:

distance (yearID, monthID, linkID (zoneID with roadTypeID), hourID, dayID, ageID, aourceTypeID) =
SHO((yearID, monthID, linkID (zoneID with roadTypeID), hourID, dayID, ageID, sourceTypeID) * averageSpeed(linkID)

10.7A. Total Activity Generator (ProjectTAG) for Project Level Modeling

This version of the ProjectTAG is used for the Project Level Domain Scale and is analogous, but also substantially different from the “Macroscale” and “Mesoscale” TAG described in the preceding sections. A MOVES project-level analysis is the modeling of one or more individual roadway links which are spatially connected to each other. It also extends the definition of project to off-network common areas within the project boundaries where vehicle starts, extended idling and evaporative emissions are allocated. This generator calculates and allocates total activity basis, starts, source-hours parked (SHP) and source hours pursuant to the run specification for each user specified link / roadway type and source type. The ProjectTAG produces core model input tables in MOVESExecution containing these results: SHO, SHP and SourceHours. The TAG does not calculate distance traveled information because it is a user input for each link.

Some overall considerations when performing these calculations are:

1. The ProjectTAG signs up for the Master Loop at the Year level which means the calculations are performed individually for the specified year, location, hour for each emission process requiring SHO, SHP or source hour activity information.

2. The MOVES design requires the user to provide all of the values basic inputs for the calculator

3. When the Total Activity Generator encounters a missing value when performing a calculation, the result of the calculation is considered as missing. Records for which the results are missing are left out of the database.

Detailed descriptions of the calculations in each TAG step follow. Each of the principal variables used in the TAG calculations are entered by the user into a special Project database or are calculated by a previous TAG step. The table in which each variable can be found is indicated in parentheses in the “Input Variables” portion of each

ProjectTAG step description. The ProjectTAG implementation can be found in the Java method “ProjectTAG.allocateTotalActivityBasis”.

10.7A.1 Determine Project Context

The MOVES Runspec and the Project Data Manager input must have the same context in terms of MOVES countyID, zoneID, yearID, and hourID. Only one value of each may be specified in a Project Level Runspec. Both DayIDs, any and all sourcetypeIDs, pollutantID and processID may be chosen in the Runspec, but these must be consistent between the Runspec and the Project Data Manager.

The ProjectTAG allocates activity according the Runspec and Project Data Manager inputs. The allocations are according to:

Age	SourceTypeAgeDistribution as a function{year} – either a user input or calculated by ProjectTAG.
SHO	source hours operating
SHP	source hours parked
Starts	vehicle starts / soak times
ExtIdle	extended idle operation
SH	source hours

The possible process allocations for ProjectTAG include:

evapPermeationProcess
evapFuelVaporVentingProcess
evapFuelLeaksProcess
evapNonFuelVaporProcess
runningExhaustProcess
extendedIdleProcess
exhaustStartProcess
brakeWearProcess

ProjectTAG-2a ***Total Population Calculation***

Calculate the total population for ages 1 through 30 for the given year and sourcetypes

Input Variables:

sourceTypeAgePopulation.population,

where yearid = year

sourcetypeID = sourcetype

Output Variable:

sourceTypeAgePopulationSummary. totalPopulation

Calculation:

Sum [sourceTypeAgePopulation.population[ageID]] AgeID=x, where x is 1 through 29

ProjectTAG-2b ***Age Fraction Calculation***

Calculate the age fraction for the given year, age and sourcetype

Input Variables:

sourceTypeAgePopulation.population,

sourceTypeAgePopulationSummary. totalPopulation

where yearid = year

sourcetypeID = sourcetype

Output Variable:

sourceTypeAgeDistribution. ageFraction

Calculation:

p.population / s.totalPopulation) as ageFraction"

Where

sourceTypeAgePopulationSummary as s"

sourceTypeAgePopulation as p "

ProjectTAG-3 ***Calculation Source Hours Operating (SHO)***

This section calculates the sourcehours operating if a roadway link is selected by the user. Context variables are from the Runspec.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

dayOfAnyWeek. noOfRealDays

link. linkAvgSpeed

link. linkVolume

link. linkLength

linkSourceTypeHour. sourceTypeHourFraction

sourceTypeAgeDistribution. ageFraction

Output Variable:

SHO, distance

Calculation:

SHO = (linkVolume * sourceTypeHourFraction * noOfRealDays * ageFraction) * linkLength / linkAvgSpeed

distance = linkVolume * sourceTypeHourFraction * noOfRealDays * ageFraction * linkLength

ProjectTAG-4 ***Calculation Source Hours Parked (SHP)***

This section calculates the sourcehours parked if an off-network link is selected by the user. Context variables are from the Runspec.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

offNetworkLink. vehiclePopulation

offNetworkLink. parkedVehicleFraction

dayOfAnyWeek. noOfRealDays

sourceTypeAgeDistribution. ageFraction

Output Variable:

SHP

Calculation:

SHP = (vehiclePopulation * parkedVehicleFraction * ageFraction)

ProjectTAG-5 ***Calculation Starts (STARTS)***

This section calculates the number of starts if an off-network link is selected by the user. Context variables are from the Runspec.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

offNetworkLink. vehiclePopulation

offNetworkLink. startFraction

sourceTypeAgeDistribution. ageFraction

Output Variable:

STARTS

Calculation:

STARTS = (vehiclePopulation * startFraction * ageFraction)

ProjectTAG-6 ***Calculation Extended Idle (ExtIdle)***

This section calculates the fraction of extended idle operation if an off-network link is selected by the user. Context variables are from the Runspec.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

offNetworkLink. vehiclePopulation

offNetworkLink. extendedIdleFraction

sourceTypeAgeDistribution. ageFraction

Output Variable:

extendedIdleFraction

Calculation:

extendedIdleFraction = (vehiclePopulation * extendedIdleFraction * ageFraction)

ProjectTAG-7a ***Calculation Source Hours (SH)***

This section calculates the source hours if an off-network link is selected by the user. Context variables are from the Runspec. This variable is used in the calculation of evaporative processes.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

offNetworkLink. vehiclePopulation

offNetworkLink. parkedVehicleFraction

dayOfAnyWeek. noOfRealDays

sourceTypeAgeDistribution. ageFraction

Output Variable:

SH

Calculation:

SH = (vehiclePopulation * parkedVehicleFraction * ageFraction)

ProjectTAG-7b ***Calculation Source Hours (SH)***

This section calculates the source hours if a roadway link is selected by the user. Context variables are from the Runspec. This variable is used in the calculation of evaporative processes.

Input Variables:

runSpecSourceType. Context.sourceTypeID

runSpecDay. Context (dayID)

runSpecMonth. Context (monthID)

hourDay. Context (hourDayID)

dayOfAnyWeek. noOfRealDays

link. linkAvgSpeed

link. linkVolume

link. linkLength

linkSourceTypeHour. sourceTypeHourFraction

sourceTypeAgeDistribution. ageFraction

Output Variable:

SH

Calculation:

SH = (linkVolume * sourceTypeHourFraction * noOfRealDays * ageFraction) * linkLength / linkAvgSpeed

10.7B. Link Operating Mode Distribution Generator (ProjectTAG) for Project Level Modeling

The Link Operating Mode Distribution Generator is used in the calculation of project domain emission inventories, and is closely associated with the ProjectTAG generator. This class called “LinkOperatingModeDistributionGenerator.java” builds an operating mode distribution for a particular project domain link from a speed-time driving cycle when it is supplied by the user. The generator is not utilized in cases where the user chooses to enter the operating mode distribution for the particular link directly as a distribution.

10.7B.1 Calculate VSP Second by Second

This section calculates the vehicle specific power (VSP) in units of kW/tonne using first principals of vehicle dynamics. The code recursively joins ‘itself’ (i.e., driveScheduleSecondLink) to create acceleration and previous second acceleration values for the special case of the ‘braking’ opmode (opmodeID = 0).

Input Variables:

sourceUseType. sourcetypeID

driveScheduleSecondLink. linkID

driveScheduleSecondLink. secondID

driveScheduleSecondLink. speed

driveScheduleSecondLink. grade

sourceUseType. sourceMass

sourceUseType. rollingTermA

sourceUseType. rotatingTermB

sourceUseType. dragTermC

constantTerm1	0.44704 – conversion miles/hour to meters/second
constantTerm2	9.81 – gravitational constant meters/sec^2

Output Variable:

VSP

Calculation:

VSP = [((speed*0.44704)*(rollingTermA+ (speed*0.44704)*(rotatingTermB+ dragTermC*(speed*0.44704)))/sourceMass] + (speed*0.44704)*(a.speed-b.speed)*0.44704 + (9.81*sin(atan(grade/100.0))*(speed*0.44704)))

10.7B.2 Count the Total Seconds in the Link

This section counts the total seconds in the link. It is grouped by source type and link.

Input Variables:

tempDriveScheduleSecondLink. sourceTypeID

tempDriveScheduleSecondLink. linkID

Output Variable:

tempDriveScheduleSecondLinkTotal. secondTotal

Calculation:

Count(*) group by sourceTypeID and linkID

10.7B.3 Count the Seconds per Individual opModeID

This section counts the total seconds in the link and opmodeID combination. It is grouped by source type, opmodeID and link.

Input Variables:

tempDriveScheduleSecondLink. sourceTypeID

tempDriveScheduleSecondLink. linkID

tempDriveScheduleSecondLink. opModeID

Output Variable:

tempDriveScheduleSecondLinkCount. secondCount

Calculation:

Count(*) group by sourceTypeID, linkID and opModeID

10.7B.4 Calculate the opMode Fraction for each opMode

This section calculates the operating mode fraction by source type, operating mode and link.

Input Variables:

tempDriveScheduleSecondLinkCount. sourceTypeID

tempDriveScheduleSecondLinkCount. linkID

tempDriveScheduleSecondLinkCount. opModeID

Output Variable:

tempDriveScheduleSecondLinkFraction. opModeFraction

Calculation:

opModeFraction = (secondCount*1.0/secondTotal)

10.7B.5 Calculate the opMode Distribution Linking hourdayID and polprocessID

This section calculates the operating mode distribution and creates the opmode distribution table. It is grouped by sourceTypeID, hourDayID, linkID, and polProcessID.

Input Variables:

tempDriveScheduleSecondLinkFraction. sourceTypeID

tempDriveScheduleSecondLinkFraction. linkID

tempDriveScheduleSecondLinkFraction. opModeID

tempDriveScheduleSecondLinkFraction. opModeFraction

opModePolProcAssoc. polProcessID

RunSpecHourDay. hourDayID

Output Variable:

opModeDistribution. opModeFraction

Calculation:

opModeFraction

10.8. Running OperatingModeDistributionGenerator (OMDG) for Macroscale

The OperatingModeDistributionGenerator is used for the running and brakewear emission processes and is only relevant for pollutant-processes which have multiple operating modes. For these pollutant-processes the OMDG calculates the distribution of operating modes for each source type on each roadway type modeled using data from the MOVESExecution database. The resulting distributions are added to the OperatingModeDistribution core model input table. This version of the OMDG is used at Macroscale.

The method used to generate these operating mode distributions in MOVES is a refinement of the method described in section 7.1.3 of the *Draft Design and Implementation Plan for MOVES*. The task of the OMDG is to produce operating mode distributions, in terms of a set of VSP-and-speed-range-based operating modes, for each combination of source type, road type, hour of the day, and day of the week using as input average speed distribution information for each such combination. Driving schedules representing typical operation at different average speeds for each source type operating on each road type play an intermediate role in translating average speed information into VSP distributions. Each average speed bin used in the input average speed distributions is represented by a pair of “bracketing” driving schedules one of which has a slightly higher average speed and one of which has a slightly lower average speed. VSP is calculated on a second by second basis for the source use type operating over these two schedules and the results are weighted appropriately to represent the average speed distribution. A full discussion of the operating mode definitions and the use of vehicle specific power (VSP) and driving schedules in MOVES is contained in a separate report, *MOVES2004 Energy and Emissions Inputs*, downloadable from the MOVES web site.

This algorithm is divided into seven steps referred to as OMDGs (operating mode distribution generator steps) in this document. Most OMDGs reference the MOVESExecution database and all implement a simple mathematical formula. Steps 1-3 take on-road driving schedules stored in the MOVESExecution database and determine the mix of these schedules to use based on the mix of average speeds and roadway types. These driving schedules may each consist of multiple "snippets" or sections of driving which are disconnected in time. Step 4 calculates VSP; Step 5 determines the operating mode bin (based on speed and VSP); and Steps 6 and 7 determine the overall mix of operating mode bins based on the mix of roadway types and average speed.

A brief description of each of the eight OMDGs is shown in table 10-4:

Table 10-4. Overview of Calculation Steps

Step	**Description**
OMDG-1	Determine bracketing drive schedules
OMDG-2	Determine proportions for bracketing drive schedules
OMDG-3	Determine distribution of drive schedules, including a sub-step to reflect separation of ramp and non-ramp freeway driving
OMDG-4	Calculate second-by-second vehicle specific power (VSP)
OMDG-5	Determine operating mode bin for each second
OMDG-6	Calculate operating mode fractions for each drive schedule
OMDG-7	Calculate overall operating mode fractions

The Operating Mode Distribution Generator signs up for the Master Loop at the Year level which means that it executes for each year in the run specification for each Link location for the running emission process.

The MOVES design allows the user to directly provide some or all of the operating mode distribution values in core model input tables such as OpModeDistribution. The InputDataManager places these user-supplied values in the MOVESExecution database OpModeDistribution table before the Operating Mode Distribution Generator is activated. The Operating Mode Distribution Generator does not replace any such user supplied values.

When the Operating Mode Distribution Generator encounters a missing value when performing a calculation, the result of the calculation is considered as missing.

Records for which the results are missing are not represented by a value of zero but are left out of the database.

The detailed descriptions of the calculations in each OMDG step are as follows: Each of the variables used in the OMDG calculations either exists in the MOVESExecution database or is calculated by a previous OMDG step. All of the MOVESExecution variables are described in the database documentation.

10.8.1. OMDG-1: Determine bracketing drive schedules

Each average speed bin lies between (is bracketed) by the average speeds of two drive schedules. This step determines which two drive schedules bracket the average speed bin and stores the identity and average speeds of the two bins. This is done for each source type, and roadway type for each average speed bin. However, it is not done for ramp drive schedules. A separate algorithm that does not include bracketing is used for ramps.

Input Variables:

AvgBinSpeed (avgSpeedBinID) from the AvgSpeedBin table.
AverageSpeed(driveScheduleID) from the DriveSchedule table.

Output Variables:

BracketScheduleLo (sourceTypeID, roadTypeID, avgSpeedBinID)
LoScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)
BracketScheduleHi (sourceTypeID, roadTypeID, avgSpeedBinID)
HiScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)

Calculation:

For each sourceTypeID, roadTypeID, and avgSpeedBinID:

The output variables are determined using table DriveSchedule, where for each combination of sourceTypeID, roadTypeID, avgSpeedBinID, bracketScheduleLo and bracketScheduleHi are determined as the drive schedules associated with the source type and road type with the next lowest and next highest average speeds compared to the value of avgBinSpeed. loScheduleSpeed and hiScheduleSpeed are the values of averageSpeed for the bracket schedules. The DriveScheduleAssoc table contains the information as to which driveScheduleID's are associated with which sourceTypeID's and which roadTypeID's.

10.8.2. OMDG-2: Determine proportions for bracketing drive schedules

This step determines the proportion of each of the bracketing drive schedules such that the combination of the average speeds of drive schedules equals the nominal average speed of each average speed bin. The results are then weighted by the fraction of all operating time that are represented by the time spent in that average speed bin. This is done for each source type, roadway type, day of week and hour of day.

Input Variables:

AvgBinSpeed (avgSpeedBinID) from the AvgSpeedBin table.
AvgSpeedFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID) from the AvgSpeedDistribution table.

with

BracketScheduleLo (sourceTypeID, roadTypeID, avgSpeedBinID)
loScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)
bracketScheduleHi (sourceTypeID, roadTypeID, avgSpeedBinID)
hiScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)

Output Variable:

loScheduleFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID)
hiScheduleFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID)

Calculation:

```
For each sourceTypeID, roadTypeID, hourDayID and avgSpeedBinID:
loScheduleFraction =
( hiScheduleSpeed – avgBinSpeed) /
(hiScheduleSpeed – loScheduleSpeed)
If bracketScheduleHi=bracketScheduleLo (meaning the average speeds of the
    "Lo and "Hi" schedule is the same):
    loScheduleFraction = 1
    hiScheduleFraction = (1 – loScheduleFraction)
Weight the results by the fraction of all speeds that are represented by that
    average speed bin.
loScheduleFraction = loScheduleFraction *  avgSpeedFraction
hiScheduleFraction = hiScheduleFraction *  avgSpeedFraction
```

10.8.3. OMDG-3: Determine distribution of drive schedules

This step includes a preliminary sub-step which accounts for the effect of ramp driving. The rampFraction field in the RoadType table indicates the fraction of time on each roadway type which is spent on ramps. The "isRamp" field in the DriveScheduleAssoc table determines whether a schedule is to be associated with ramp driving or with driving on roadways exclusive of ramps. Currently, the MOVESDefault contains ramp activity with a constant ramp fraction of 8 percent for the two roadtypes with "isRamp = 'Y'.

User attempting to modify the database in this area should keep in mind several data constraints: There must always be at least one driving schedule not indicated as a ramp which is associated with each combination of source type and roadway type in the DriveScheduleAssoc table

OMDG-3a: Determine distribution of ramp schedules

This step accounts for ramps, based on the input fields RampFraction and opModeFraction contained in table RoadType and RoadOpmodeDistribution tables, respectively. For any driving schedule which is associated with the source type and roadway type which has an IsRamp value of "Y", the following methodology is used to calculate the ramp contribution.

The calculation starts, by obtaining from the RoadOpmodeDistribution table, the operating mode distribution for each sourcetype, average speed bin and roadtype (only those which contain ramp driving are selected). Next, the algorithm determines the road type's average speed for each sourcetype and hour. The avgSpeedFraction is taken from the avgSpeedDistribution table and the avgBinSpeed is from the avgSpeedBin table.

Input Variables:

avgSpeedFraction(sourceTypeID, roadTypeID, hourDayID,
avgSpeedBinID
avgBinSpeed(avgSpeedBinID)

roadTypeSourceTypeAvgSpeedBinID(sourceTypeID, roadTypeID, hourDayID)

Output Variable:

avgSpeed(sourceTypeID, roadTypeID, hourDayID)

Calculation:

avgSpeed = sum(avgSpeedFraction * avgBinSpeed)

Once the average speed is determined using the table roadTypeSourceTypeAvgSpeedBinID, its bin must be assigned by taking minimum avgSpeedBinID from avgSpeedBin table such that avgBinSpeed >= avgSpeed.

Ramps are not considered separate roadtypes in MOVES, but their contribution is weighted into the two restricted entry roadtypes (i.e., rural and urban freeways, non-freeway types contain no ramp fractions) using the weighed ramp opmode fraction and the rampfraction (i.e, 8% for all ramps). A separate ramp cycle is now available for each avgSpeedBin in MOVES. A total of 16 ramp cycles are present and differentiated by average speeds which range from 2.5 to 75 mph. Each ramp cycle contains a different opmode fraction distribution due the different speeds, acceleration and VSP values of each cycle. The opModeFraction is obtained from the RoadOpmodeDistribution table using avgSpeedBinID entries that match roadTypeSourceTypeAvgSpeedBinID for each hour and day. The opmode fraction from the table weightedRampOpmodeFraction is used in subsequent calculations for roadtypes which contain on and off ramps.

Input Variables:

rampFraction(roadTypeID)

opModeFraction(sourceTypeID, roadTypeID, opModeID, avgSpeedBinID)

Output Variable:

weightedRampOpModeFraction(sourceTypeID, roadTypeID, hourDayID, opModeID)

Calculation:

weightedRampOpModeFraction = rampFraction* opModeFraction

OMDG-3b: Determine distribution of non-ramp schedules

This calculation adjusts for the RampFraction on the given roadway. This step determines the distribution of drive schedules which represents the sum of all of the average speed bins. This is done for each source type, roadway type, day of week and hour of day for all driving schedules which are associated with the source type and roadway type which have an IsRamp value of "N".

Input Variables:

loScheduleFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID)
HiScheduleFraction (sourceTypeID, roadTypeID, hourDayID, AvgSpeedBinID)
RampFraction(roadTypeID)
driveScheduleID (sourceTypeID, roadTypeID, isRamp=N)

Output Variable:

DriveScheduleFraction (sourceTypeID, roadTypeID, hourDayID,)

Calculation:

For each sourceTypeID, roadTypeID, hourDayID and driveScheduleID:
driveScheduleFraction =
[(Sum Of loScheduleFraction over all cases where bracketScheduleLo = driveScheduleID) + (Sum Of hiScheduleFraction over all cases where BracketScheduleHi = driveScheduleID)] * (1-rampFraction)

10.8.4. OMDG-4: Calculate second-by-second vehicle specific power (VSP)

This step calculates the vehicle specific power (VSP) for each drive schedule for each source type. This is done for each source type, drive schedule and second.

Input Variables:

Speed (driveScheduleID, second) from the DriveScheduleSecond table.
rollingTermA (sourceTypeID) from the SourceUseType table.
rotatingTermB (sourceTypeID) from the SourceUseType table.
dragTermC (sourceTypeID) from the SourceUseType table.
sourceMass (sourceTypeID) from the SourceUseType table.

Output Variable:

VSP (sourceTypeID, driveScheduleID, second)
Accel (sourceTypeID, driveScheduleID, second)

Calculation:

For each sourceTypeID, driveScheduleID and second:
Preliminary Calculations:
speed (meters/second) = speed (mph) * 0.447 m/s per mph.
accel (meters/second2)= speed (t) – speed (t-1), with speed in meters/second.
VSP =
[rollingTermA*speed
+ rotatingTermB * speed^2
+ dragTermC * speed^3
+ sourceMass * speed * accel]
/ sourceMass
(speed in meters/second, accel in meters/second^2)

10.8.5. OMDG-5: Determine operating mode bin for each second

This step accounts for VSP, speed and accel. The VSP value for each second is compared to the upper and lower bounds for the operating mode bins and a bin ID is assigned to each second. This is done for each source type, drive schedule and second.

Input Variables:

VSPLower(opModeID) from the OperatingMode table.
VSPUpper(opModeID) from the OperatingMode table.
SpeedLower(opModeID) from the OperatingMode table.
SpeedUpper(opModeID) from the OperatingMode table.
BrakeRate1Sec(opModeID) from the OperatingMode table.
BrakeRate3Sec(opModeID) from the OperatingMode table.
Speed (sourceTypeID, driveScheduleID, second) from DriveScheduleSecond
with
VSP (sourceTypeID, driveScheduleID, second) from OMDG-4
accel (sourceTypeID, driveScheduleID, second) from OMDG-4

Output Variable:

opModeIDbySecond (sourceTypeID, driveScheduleID, second)

Calculation:

For each sourceTypeID, driveScheduleID and second:

An opModeID is assigned for each second in each drive schedule based on where the VSP, speed and accel values in that second falls. VSP and speed are compared against the VSP and speed bounds to determine to appropriate bins. Then the braking (ID=0) and idle bins (ID=1) should be assigned based on accel and speed, respectively. This sequence is important since there is overlap in the definitions between the non-braking/idle bins and the braking/idle bins.

10.8.6. OMDG-6: Calculate operating mode fractions for each drive schedule

Once all the seconds in each operating mode bin are known, the distribution of the bins is determined. The sum of the operating mode fractions sums to one for each source type and drive schedule combination. This is done for each source type and schedule.

Input Variables:

opModeIDbySecond (sourceTypeID, driveScheduleID, second)

Output Variable:

OpModeFractionbySchedule (sourceTypeID, driveScheduleID, opModeID)

Calculation:

For each sourceTypeID, driveScheduleID and opModeID:

OpModeFractionbySchedule =
(Number of Seconds in opModeID during DriveSchedule) /
(Total number of seconds in DriveSchedule)

10.8.7. OMDG-7: Calculate overall operating mode fractions

This step calculates the overall operating mode fractions by weighting the operating mode fractions of each drive schedule by the drive schedule fractions. This is done for each source type, road type, day of the week, hour, and operating mode.

Input Variables:

OpModeFractionbySchedule (sourceTypeID, driveScheduleID, opModeID)
DriveScheduleFraction (sourceTypeID, roadTypeID, hourDayID, driveScheduleID)

Output Variable:

OpModeFraction (sourceTypeID, roadTypeID, hourDayID, polProcessID, opModeID)

Calculation:

For each sourceTypeID, roadTypeID, hourDayID and opModeID:
OpModeFraction =
Sum of OpModeFractionbySchedule*DriveScheduleFraction over all DriveSchedules

The Results of OMDG-7 populate the OpModeDistribution table of the Execution Location Database.

The opModeFraction in the OpModeDistribution table is:

opModeFraction (sourceTypeID, linkID, hourDayID, polProcessID, opModeID)

The value of linkID needed for this table is determined from roadTypeID and zoneID.

10.9. Running OperatingModeDistributionGenerator (OMDG) for Mesoscale Lookup

The OperatingModeDistributionGenerator is used for the running, and brakewear emission processes and is only relevant for pollutant-processes which have multiple operating modes. For these pollutant-processes the OMDG calculates the distribution of operating modes for each source type on each roadway type modeled using data from the MOVESExecution database. The resulting distributions are added to the OperatingModeDistribution core model input table. This version of the OMDG is used for Mesoscale Lookup and takes advantage of the duplication of LinkAverageSpeeds produced by the LookupTableLinkProducer. Some calculations that are done at the roadtype level for Macroscale must be done at the link level for Mesoscale Lookup.

The method used to generate these operating mode distributions in MOVES is a refinement of the method described in section 7.1.3 of the *Draft Design and Implementation Plan for MOVES*. The task of the OMDG is to produce operating mode distributions, in terms of a set of VSP-and-speed-range-based operating modes, for each combination of source type, link, hour of the day, and period of the week using as input average speed information for each link. Driving schedules representing typical operation at different average speeds for each source type operating on each road type play an intermediate role in translating average speed information into VSP distributions. Each average speed bin used as a link average speed is represented by a pair of "bracketing" driving schedules one of which has a slightly higher average speed and one of which has a slightly lower average speed. VSP is calculated on a second by second basis for the source use type operating over these two schedules and the results are weighted appropriately to represent the link average speed. A full discussion of the operating mode definitions and the use of vehicle specific power (VSP) and driving schedules in MOVES is contained in a separate report, *MOVES2004 Energy and Emissions Inputs*, downloadable from the MOVES web site.

This algorithm is divided into seven steps referred to as OMDGs (operating mode distribution generator steps) in this document. Most OMDGs reference the MOVESExecution database and all implement a simple mathematical formula. Steps 1-3 take snippets of actual on-road driving schedules stored in the MOVESExecution database, and determine the mix of these schedules to use based on the mix of average speeds and roadway types; Step 4 calculates VSP, Step 5 determines the operating mode bin (based on speed and VSP), and Steps 6 and 7 determine the overall mix of operating mode bins based on the mix of roadway types and average speed. Each OMDG calculation step is also described table 10-4 in the immediately prior chapter.

The Operating Mode Distribution Generator signs up for the Master Loop at the Year level which means that it executes for each year in the run specification for each Link location for the running emission process.

The MOVES design allows the user to directly provide some or all of the operating mode distribution values in core model input tables such as OpModeDistribution. The InputDataManager places these user-supplied values in the MOVESExecution database OpModeDistribution table before the Operating Mode Distribution Generator is activated. The Operating Mode Distribution Generator does not replace any such user supplied values.

When the Operating Mode Distribution Generator encounters a missing value when performing a calculation, the result of the calculation is considered as missing. Records for which the results are missing are not represented by a value of zero but are left out of the database.

The detailed descriptions of the calculations in each OMDG step are as follows: Each of the variables used in the OMDG calculations either exists in the MOVESExecution database or is calculated by a previous OMDG step. The MOVESExecution variables are described in the database documentation.

10.9.1. OMDG-1: Determine bracketing drive schedules

Each average speed bin lies between (is bracketed) by the average speeds of two drive schedules. This step determines which two drive schedules bracket the average speed bin and stores the identity and average speeds of the two bins. This is done for each source type, and roadway type for each average speed bin.

Input Variables:

AvgBinSpeed (AvgSpeedBinID) from the AvgSpeedBin table.

AverageSpeed(driveScheduleID) from the DriveSchedule table.

Output Variables:

BracketScheduleLo (sourceTypeID, roadTypeID, AvgSpeedBinID)
LoScheduleSpeed (sourceTypeID, roadTypeID, AvgSpeedBinID)
BracketScheduleHi (sourceTypeID, roadTypeID, AvgSpeedBinID)
HiScheduleSpeed (sourceTypeID, roadTypeID, AvgSpeedBinID)

Calculation:

For each sourceTypeID, roadTypeID, and AvgSpeedBinID:

The output variables are determined using table DriveSchedule, where for each combination of sourceTypeID, roadTypeID, AvgSpeedBinID, BracketScheduleLo and BracketScheduleHi are determined as the drive schedules associated with the source type and road type with the next lowest and next highest average speeds compared to the value of AvgBinSpeed. LoScheduleSpeed and HiScheduleSpeed are the values of AverageSpeed for the bracket schedules. The DriveScheduleAssoc table contains the information as to which driveScheduleID's are associated with which sourceTypeID's and which roadTypeID's.

10.9.2. OMDG-2: Determine proportions for bracketing drive schedules

This step determines the proportion of each of the bracketing drive schedules such that the combination of the average speeds of drive schedules equals the average speed of the average speed bin. This calculation also takes advantage of the fact that the LTLP only uses the speeds in the AvgSpeedBin table.

Input Variables:

AvgBinSpeed (avgSpeedBinID) from the AvgSpeedBin table.

with

BracketScheduleLo (sourceTypeID, roadTypeID, avgSpeedBinID)
LoScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)
BracketScheduleHi (sourceTypeID, roadTypeID, avgSpeedBinID)
HiScheduleSpeed (sourceTypeID, roadTypeID, avgSpeedBinID)

Output Variable:

LoScheduleFraction (sourceTypeID, roadTypeID, avgSpeedBinID)
HiScheduleFraction (sourceTypeID, roadTypeID, avgSpeedBinID)

Calculation:

```
For each sourceTypeID, hourDayID and avgSpeedBinID:
IF (BracketScheduleLo=BracketScheduleHi)
    loScheduleFraction=1.0
ELSE
    loScheduleFraction =
    ( hiScheduleSpeed – avgBinSpeed) /
    (hiScheduleSpeed – loScheduleSpeed)
hiScheduleFraction = (1 – loScheduleFraction)
```

10.9.3. OMDG-3: Determine distribution of drive schedules

This step includes a preliminary sub-step which accounts for the effect of ramp driving. The rampFraction field in the RoadType table indicates the fraction of time on each roadway type which is spent on ramps. The “isRamp” field in the DriveScheduleAssoc table determines whether a schedule is to be associated with ramp driving or with driving on roadways exclusive of ramps. The MOVESDefault database provided with the Demonstration version of DRAFT MOVES2009 has no ramp activity (rampFraction = 0.0).

Users attempting to modify the database in this area should keep in mind several data constraints: There must always be at least one driving schedule not indicated as a ramp which is associated with each combination of source type and roadway type in the DriveScheduleAssoc table. Additionally, for every ramp fraction which is greater than zero in the RoadType table, there must be exactly one driving cycle which is indicated as a ramp for each source type using that roadtype. (Conversely Ramp fractions can be zero even if there is an associated ramp driving schedule; in this case the ramp schedule is simply not used.)

At Macroscale, the distribution of ramp and non-ramp schedules is determined for each roadtype; at Mesoscale Lookup, although the ramp fraction is assigned to all links of a given roadtype, the calculation must be done for each link.

OMDG-3a: Determine distribution of ramp schedules

This step is accounts for freeway and interstate ramps, based on the input field RampFraction, contained in table RoadType.

Input Variables:

RampFraction(roadTypeID)
driveScheduleID (sourceTypeID, roadTypeID, isRamp=Y)

Output Variable:

DriveScheduleFraction (sourceTypeID, hourDayID, driveScheduleID)

Calculation:

driveScheduleFraction = rampFraction(roadTypeID)

OMDG-3b: Determine distribution of non-ramp schedules

This calculation adjusts for the RampFraction on the given roadway. This step determines the distribution of drive schedules which represents the average speed bin for the links. Since the LTLP assigns only the averageSpeeds in the AvgSpeedBin table, this is simpler than for Macroscale.

Input Variables:

LoScheduleFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID)
HiScheduleFraction (sourceTypeID, roadTypeID, hourDayID, avgSpeedBinID)
RampFraction(roadTypeID)
driveScheduleID (sourceTypeID, roadTypeID, isRamp=N)

Output Variable:

DriveScheduleFraction (sourceTypeID, avgSpeedBinID, driveScheduleID)

Calculation:

```
For each sourceTypeID, avgSpeedBinID, and driveScheduleID:
driveScheduleFraction =
    [loScheduleFraction (if bracketScheduleLo = driveScheduleID) +
    hiScheduleFraction  (if bracketScheduleHi = driveScheduleID)] * (1-
    RampFraction)
```

10.9.4. OMDG-4: Calculate second-by-second vehicle specific power (VSP)

This step calculates the vehicle specific power (VSP) for each drive schedule for each source type. This is done for each source type, drive schedule and second.

Input Variables:

speed (driveScheduleID, second) from the DriveScheduleSecond table.
rollingTermA (sourceTypeID) from the SourceUseType table.
rotatingTermB (sourceTypeID) from the SourceUseType table.
dragTermC (sourceTypeID) from the SourceUseType table.
sourceMass (sourceTypeID) from the SourceUseType table.

Output Variable:

VSP (sourceTypeID, driveScheduleID, second)
accel (sourceTypeID, driveScheduleID, second)

Calculation:

```
For each sourceTypeID, driveScheduleID and second:
Preliminary Calculations:
speed (meters/second) = speed (mph) * 0.447 m/s per mph.
accel (meters/second2)= speed (t) – speed (t-1), with speed in meters/second.
VSP =
[  rollingTermA*speed
+ rotatingTermB * speed$^2$
+ dragTermC * speed$^3$
+ sourceMass * speed * accel ]
/ sourceMass
(speed in meters/second, accel in meters/second$^2$)
```

10.9.5. OMDG-5: Determine operating mode bin for each second

This step accounts for VSP, speed and accel. The VSP value for each second is compared to the upper and lower bounds for the operating mode bins and a bin ID is assigned to each second. This is done for each source type, drive schedule and second.

Input Variables:

VSPLower(opModeID) from the OperatingMode table.
VSPUpper(opModeID) from the OperatingMode table.
speedLower(opModeID) from the OperatingMode table.
speedUpper(opModeID) from the OperatingMode table.
brakeRate1Sec(opModeID) from the OperatingMode table.
brakeRate3Sec(opModeID) from the OperatingMode table.
speed (sourceTypeID, driveScheduleID, second) from DriveScheduleSecond with

VSP (sourceTypeID, driveScheduleID, Second) from OMDG-4
accel (sourceTypeID, driveScheduleID, Second) from OMDG-4

Output Variable:

opModeIDbySecond (sourceTypeID, driveScheduleID, second)

Calculation:

For each sourceTypeID, driveScheduleID and second:
An opModeID is assigned for each second in each drive schedule based on where the VSP, speed and accel values in that second falls. VSP and speed are compared against the VSP and speed bounds to determine to appropriate bins. Then the braking (ID=0) and idle bins (ID=1) should be assigned based on accel and speed, respectively. This sequence is important since there is overlap in the definitions between the non-braking/idle bins and the braking/idle bins.

10.9.6. OMDG-6: Calculate operating mode fractions for each drive schedule

Once all the seconds in each operating mode bin are known, the distribution of the bins can be determined. The sum of the operating mode fractions sums to one for each source type and drive schedule combination. This is done for each source type and drive schedule.

Input Variables:

opModeIDbySecond (sourceTypeID, driveScheduleID, second)

Output Variable:

OpModeFractionbySchedule (sourceTypeID, driveScheduleID, opModeID)

Calculation:

For each sourceTypeID, driveScheduleID and opModeID:

opModeFractionbySchedule =
(Number of seconds in opModeID during driveSchedule) /
(Total number of seconds in driveSchedule)

10.9.7. OMDG-7: Calculate overall operating mode fractions

This step calculates the overall operating mode fractions by weighting the operating mode fractions of each drive schedule by the drive schedule fractions. This is done for each source type, link, day of the week, hour of the day and operating mode. The opmodeFractions created should vary only by sourceType, roadType, and link speed. They should be the same for each zone, day, and hour.

Input Variables:

OpModeFractionbySchedule (sourceTypeID, driveScheduleID, opModeID)
DriveScheduleFraction (sourceTypeID, roadTypeID, avgtSpeedBinID, driveScheduleID)
Link(linkID, countyID, zoneID, roadTypeID)
LinkAverageSpeed(linkID, hourDayID, sourceTypeID)

Output Variable:

OpModeFraction (sourceTypeID, linkID, hourDayID, polProcessID, opModeID)

Calculation:

For each sourceTypeID, linkID, hourDayID and opModeID:
opModeFraction =
Sum of OpModeFractionbySchedule*DriveScheduleFraction over all DriveSchedules where the roadTypeID of the Link equals the roadTypeID of the driveScheduleFraction and the averageSpeed of the Link equals the averageSpeed of the avgSpeedBinID.

The Results of OMDG-7 populate the OpModeDistribution table of the Execution Location Database.

The OpModeFraction in the OpModeDistribution table is:

OpModeFraction (sourceTypeID, linkID, hourDayID, polProcessID, opModeID)

10.10. Source Bin Distribution Generator (SBDG)

The Source Bin Distribution Generator produces the distribution of source bins by source type and model year. This information provides the mapping between the activity elements of MOVES (total activity and operating modes), which are based on source use type, and the emission rates, which are based on source bin. The SBDG takes as input fleet distributions of source bin categories (e.g. weight class, engine size, fuel type etc.) by model year.

Data about the characteristics of the existing and projected vehicle fleet are stored in several of the tables within the MOVESExecution database as shown in table 10-5. The Source Bin Distribution Generator uses information in the first seven of these tables to populate the last two: SourceBin and SourceBinDistribution which are core model input tables.

Table 10-5. Tables used by SourceBinGenerator

Table Name	**Key Fields**	**Additional Fields**	**Notes**
SourceTypePolProcess	sourceTypeID polProcessID	isSizeWeightReqd isRegClassReqd isMYGroupReqd	Indicates which pollutant-processes the source bin distributions may be applied to and indicates which discriminators are relevant for each sourceType and polProcess
FuelEngFraction	sourceTypeModelYearID fuelTypeID engTechID	fuelEngFraction	Joint distribution of vehicles with a given fuel type and engine technology. Sums to one for each sourceType & modelYear
SizeWeightFraction	sourceTypeModelYearID fuelTypeID engTechID weightClassID engSizeID	sizeWeightFraction	Joint distribution of engine size and weight. Sums to one for each sourceType, modelYear and fuel/engtech combination.

RegClassFraction	sourceTypeModelYearID fuelTypeID engTechID regClassID	regClassFraction	Fraction of vehicles in a regulatory class. Sums to one for each sourceType, modelYear and fuel/engtech combination.
PollutantProcessModel Year	polProcessID modelYearID	modelYearGroupID	Defines model year groups.
ModelYearGroup	modelYearGroupID	shortModYrGroupID modelYearGroupName	Defines short model year group Ids.
SourceTypeModelYear	sourceTypeModelYearID	modelYearGroupID modelYearID sourceTypeID	Decodes ID field into modelYearID and sourceTypeID
SourceBin	SourceBinID	engSizeID fuelTypeID engTechID regClassID modelYearGroupID weightClassID	List of sourceBins
SourceBinDistribution	SourceTypeModelYearID polProcessID sourceBinID	sourceBinActivityFract ion	Stores source bin distributions.

The SourceBinGenerator uses information from the run specification to determine which sourcetypes, modelyears, fuel-types, pollutants and processes are relevant, and limits the Generator action to the relevant sources.

"Source bin discriminators" refer to characteristics that are used to distinguish the source bins. The MOVES source bin discriminators are:

fuelType*
modelYearGroup
engTech*
regClass
engSize*
weight Class*
(* fuelType and engTech are distinct but highly correlated, and, thus, handled together; similarly engSize and weightClass are handled together)

a. Because it would be awkward to have all these separate fields in each database table related to source bins, this information is combined into a single unique BIGINT identifier 1ffttrryyssssswwww00, where the digits represent the bin discriminators as follows:

Leading 1 (1)
Fuel (ff)
EngineTech (tt)
Regulatory Class (rr)
Model Year Group (yy)
Engine Size (ssss)
Engine Weight (wwww)
Extra zeros (00)
With the exception of the ModelYear group, each bin discriminator is coded as for the discriminator ID (see the database attribute table), with leading zeros added as needed. ModelYear group is coded using the shortModYrGroupID as defined in the database table ModelYearGroup.

The ordering of the subfields within this identifier is essentially arbitrary. The SourceBin table contains both the individual fields and the combined source bin identifier and can be used to "decode" the combined identier values.

b. The SourceTypePolProcess table identifies which discriminators are relevant for each source type and pollutant/emission process (polProcess). If a discriminator is not relevant for a given source type & polProcess, the discriminator ID is set to 0 (which means that the discriminator value doesn't matter) and the fraction of vehicles to which this value of this discriminator applies is set to 1 for all model years.

c. The SourceBinGenerator "signs up" with the MOVES MasterLoop at the PolProcess level which means it executes only once for each relevant process (running exhaust, start exhaust, and extended idle exhaust).

d. For each RunSpec-required source type, polProcess, fuel type and model year (as implied by the selected year and the largest valid ageID), and for each relevant/existing combination of sourcebin discriminators (as determined by the input tables), the sourceBinActivityFraction is calculated as follows:

$$\text{sourceBinActivityFraction} = \text{fuelEngFraction} * \text{regClassFraction} * \text{sizeWeightFraction}$$

e. The SourceBinDistribution table lists the sourceBinActivityFractions for each sourceBin, sourceType, modelYear and polProcess. The set of such records for a given sourceType, modelyear, and polProcess constitute a source bin distribution which sums to unity.

f. The SourceBin table provides a list of unique source bins that have a SourceBinActivityFraction > 0 in at least one source bin distribution. The SBDG adds sourceBinID records to this table if necessary, but does not duplicate records already present.

g. If data is not available for all the model years implied by the calendar year in the run specification, source bin distributions for the missing years prior to the first populated for the same sourceType and polProcess are generated which are equal to the distribution from that first populated model year. Source bin distributions are also generated for missing model years after the last populated for the same sourceType and polProcess which are equal to the distribution from that last populated model year. Years extrapolated are only those earlier than the earliest model year for which data is provided or later than the latest model year for which data is provided.

h. If any data is already present in the CMIT SourceBinDistribution table for a combination of sourceType and polProcess, which would mean that the user had entered this information directly, this Generator does not produce SourceBinDistribution output for that combination.

The approach to performing the calculation steps implied by these specifications is shown in table 10-6:

Table 10-6. SourceBin Generator Calculation Steps

Step/Query	**Description**
1 populate sb_fractions	Select data from the SourcBinGenerator tables to fill a temporary table with identifying and distribution records for each relevant combination of sourceType, pollutant-process and bin identifier.
2_1 sb_fractions update regulatory class.	For each record, for pollutant-processes that do not require regulatory class information, set regClassID to 0 and regClass Fraction to 1.
2_2 sb_fractions update MY GroupID	For each record, for pollutant-processes that do not require model year group information, set MYGroupID to 0.
2_3 sb_fractions update size and weight	For each record, for pollutant-processes that do not require size and weight information, set engSizeID and weightClassID to 0 and sizeWeightFraction to 1.
2_5 calculate sb_fractions	For each record, calculate sb_fractions as the product of fuelEngFraction, emisTechFraction, and sizeWeightFraction.
2_7 populate sb_fractions no dups	Remove duplicate records from sb_fractions; save as a second temporary table, “sb_fractions no dups”.
4 populate SourceBin	Select unique sourceBinIDs with fractions >0 to fill SourceBin table.
4_5 generate bin IDs	Generate sourceBinIDs from bin discriminators.
5 append source bin distributions	Use SourceBin table and “sb_fractions no dups” to fill SourceBinDistribution table.

10.11. Meteorology Generator

Basic meteorological parameters such as temperature, humidity and heat index are stored in the MOVESExecution database table ZoneMonthHour. This generator uses the temperature and relative humidity information in the ZoneMonthHour Table and performs the necessary calculations to populate the heatIndex and specific humidity (or humidity ratio) fields in this table.

This generator subscribes to the MOVES master loop mechanism at the process level for the running and extended idling emission processes.

The Meteorological Generator uses the temperature and relative humidity data in the ZoneMonthHour table to populate the Heat Index column.

For each RunSpec-required zone, month and hour, the Heat Index is calculated by following the algorithm:

```
HI = -42.379 + 2.04901523*T + 10.14333127*RH + -0.22475541*T*RH +
     -6.83783 *0.001*T*T + -5.481717 * 0.01 * RH*RH +
     1.22874*0.001*T*T*RH + 8.5282*0.0001*T*RH*RH +
     -1.99*.000001*T*T*RH*RH
```

where HI is the heat index, T is temperature in degrees F, and RH is the relative humidity in percent. This formula is a recent heat index algorithm used by the National Weather Service.

The MetGenerator has been expanded in DRAFT MOVES2009 to also calculate the specific humidity field of the ZoneMonthHour table. The equations used to convert from relative humidity in percent to specific humidity (or humidity ratio) in units of grains of water per pound of dry air were taken from CFR section 86.344-79.

Inputs:

T_F is the temperature in degrees F. (from temperature field in ZoneMonthHour)
P_b is the barometric pressure. (from barometricPressure field in County)
H_{rel} is the relative humidity (from relHumidity field in ZoneMonthHour)

$$T_K = \left(5\!\!\diagup\!\!9\right)\left[T_F - 32\right] + 273$$

$$T_0 = 647.27 - T_K$$

$$H_{ratio\ or\ specific humidity} = 4347.8 * P_V / (P_b - P_V)$$

$$P_V = \left(H_{rel}\!\!\diagup\!\!100\right) P_{db}$$

$$P_{db} = 29.92 * 218.167 * 10^{(-T_0/T_K)\left[\frac{(3.2437+0.00588T_0+0.0000000117T_0^3)}{1+0.00219T_0}\right]}$$

$$= 6527.557 * 10^{(-T_0/T_K)\left[\frac{(3.2437+0.00588T_0+0.0000000117T_0^3)}{1+0.00219T_0}\right]}$$

10.12. Start OperatingModeDistributionGenerator (StartOMDG)

The StartOMDG signs up with the master loop mechanism at the Zone level. Its substantive calculations are independent of geographic location and depend in time only upon hour and day.

The operating modes used for the start process of the criteria pollutants represent ranges of the amount of time vehicles have been parked before being started and are listed near the end of section 9.7.

The steps of this calculation can be logically specified as follows:

1. Compute the soak length of each trip in the SampleVehicleTrip table. This equals the keyOnTime of the trip minus the keyOffTime of the previous trip. Trips having no prior trip (priorTripID = NULL) are disregarded in the calculation.

2. Assign a start opModeID to the trip by comparing its soak length with the minSoakBound and maxSoakBound values in the OperatingMode table.

3. Then for each sourceTypeID and hourDayID, the opModeFraction of each opModeID equals:

   the count of records having the opModeID
   divided by
   the number of keyOnTime records.

   Trips having no prior trip are excluded from both the numerator and the denominator of this ratio.

   Steps 1 through 3 are performed at most once for each model run.

4. During each Zone level iteration of the StartOMDG masterloopable, the operating mode distribution (which is by sourceTypeID, hourDayID and opModeID) resulting from steps 1-3 is stored into the OpModeDistribution CMIT for the off-highway-network link location in the zone for each pollutant whose start process emissions are required by the run specification and whose calculation requires an operating mode distribution. (Estimating the start process emissions of the “greenhouse gas” pollutants in DRAFT MOVES2009 does not require an operating mode distribution, whereas estimating the start process emission of the “criteria” pollutants added in DRAFT MOVES2009 does involve using an operating mode distribution.)

Records already existing in the CMIT are not overwritten.

10.13. Tank Temperature Generator (TTG)

10.13.1. Functional Characteristics

The TTG signs up at the ZONE master looping level. It must execute before the Evap Operating Mode Distribution Generator.

The TTG uses the Sample Vehicle and SampleVehicleTrip tables, in conjunction with ambient temperature information in the ZoneMonthHour table, and temperature effect information from the TankTemperatureRise table to calculate the contents of the AverageTankTemperature, SoakActivityFraction tables, ColdSoakTankTemperature and ColdSoakInitialHourFraction tables which are CMITs.

No records are added to the AverageTankTemperature table for a combination of tankTemperatureGroupID, zoneID, and monthID if any record pertaining to this combination is already present. No records are added to the SoakActivityFraction table for a combination of sourceTypeID, zoneID, and monthID if any records are already present for it. No records are added to the ColdSoakInitialHourFraction table for a combination of sourceTypeID, zoneID, and monthID if any records are already present for it.

Because Mesoscale Lookup does not compute the emissions of parked cars, some TTG steps are skipped to reduce computing time when Mesoscale Lookup is selected.

10.13.2. Detailed Calculation Steps

TTG-1 Calculate ColdSoakTankTemperature

Inputs:
- Temperature (zoneMonthHour)

Output:
- coldSoakTankTemperature (zoneID, monthID, hourID). This is saved in the ColdSoakTankTemperature CMIT.

Preliminary Calculation: 15minuteTemperature - Create intermediate table of temperatures in 15 minute time steps, with key fields hourID, timeStep (1 through 4 with 1 representing the top of the hourID). Within each hourly temperature, set time step 1 to the corresponding value from zoneMonthHour. Then perform linear interpolation with hourID+1 timeStep 1 to fill in hourID TimeStep 2-4 . For the “highest” hour ID interpolate (default = 24) with the “lowest” hour ID (default = 1).

Preliminary Calculation: 15minuteTankTemperature - Performed on the 15 minutes temperature table produced in the preliminary calculation. 15minuteTankTemperate and tempDelta need to be calculated at each time step before performing calculations on the next time step.

```
- for hourID =1, timeStep=1
        15minuteTankTemperature
                = 15minuteTemperature (hourID=1, timeStep=1)

        tempDelta
                = 15minuteTemperature – 15minuteTankTemperature

- for all other hourID, timeSteps:
        15minuteTankTemperature
                = 1.4*SUM(tempDelta hourID=1, timeStep=1 through
                most recent hourID, timeStep)+ 15minuteTankTemperature
                (hourID 1, timeStep 1)

        tempDelta
                = 15minuteTemperature - 15minuteTankTemperature
```

Calculation: coldSoakTankTemperature (hourID) = 15minuteTankTemperature (hourID, timeStep=1)

TTG-2 Create sampleVehicleTripByHour

Trips in the sampleVehicleTrip table may span the top of the next hourDayID. This step is needed to parse the trips into segments for which the start and finish are within the same hourID. “Marker Trips”, which have a keyOffTime, but null value of keyOnTime are ignored in this calculation.

Inputs:

- SampleVehicleTrip

Outputs:

- Intermediate table SampleVehicleTripByHour (vehicleID, tripID, hourDayID, endOfHour, keyOnTime, KeyOffTime, startOfTrip, endOfTrip)

Preliminary calculation: calculate "endOfHour" for each record not representing a "Marker Trip" = next highest multiple of 60

Calculation: Create a new "trip" record when keyOffTime for a trip > endOfHour. The tripID will remain the same, but the hourID will increment accordingly. The fields "startOfTrip" or "endOfTrip" will be added to table to identify whether the trip record is an actual trip start or trip end. The specific steps are as follows:

- Set startOfTrip=1 for each existing trip in SampleVehicleTrip
- Records need to be split when keyOffTime > endOfHour
    - New keyOffTime for existing record = endOfHour
    - Create new record for tripID, with hourID = hourID+1 and endOfHour = endOfHour + 60
    - keyOnTime for new record = endOfHour (hourID-1) + 1
    - keyOffTime is the same as original record, unless it is > endOfHour – in which case repeat these steps until a record is created where keyOffTime < endOfHour
- "endOfTrip" = 1 for record where keyOffTime<endOfHour (i.e. last split).

TTG-3 Create hotSoakEventByHour

Inputs:
- SampleVehicleTripByHour (vehicleID, tripID, hourDayID, endOfHour, keyOnTime, KeyOffTime, startOfTrip, endOfTrip)

Output:
- Intermediate table hotSoakEventByHour (vehicleID, tripID, hourDayID, endOfHour, hotSoakBegin, hotSoakEnd, startOfSoak, endOfSoak)

Calculation:
- Select records from sampleVehicleTripByHour where endOfTrip=1
- hotSoakBegin = keyOffTime of TripID; startOfSoak=1
- if keyOnTime of "next trip" (define by priorTripID=TripID and startOfTrip=1) < endOfHour for TripID, then hotSoakEnd = keyOnTime of next trip; endOfSoak=1
- Otherwise, if there is a next trip
    - hotSoakEnd = endOfHour for TripID; endOfSoak=0
    - Create new record for tripID, with hourID = hourID+1 and endOfHour = endOfHour + 60

        - hotSoakBegin for new record = endOfHour (hourID-1) + 1
        - hotSoakEnd is the same as original record, unless it is > endOfHour – in which case repeat these steps until a record is created where keyOffTime < endOfHour; at which point set endOfSoak=1
- Otherwise, if there is no next trip
    - hotSoakEnd = endOfHour for TripID; endOfSoak=0
    - Repeat these steps until hourID=24 is reached
        - Create new record for tripID, with hourID = hourID+1 and endOfHour = endOfHour + 60
        - hotSoakBegin for new record = endOfHour (hourID-1) + 1
        - hotSoakEnd = endOfHour for TripID; endOfSoak=0

TTG-4 Calculate Hot Soak and Operating Tank Temperature by Parsed Trip

This step computes operating tank temperatures for the beginning and end times of the parsed trips in sampleVehicleTripByHour, and for each minute for hot soaks (necessary because hot soak is modeled as a decay function). The calculations must be performed in tandem since the initial temperature for each trip is a function of the final temperature from the preceding hot soak, and vice versa.

Inputs:

- sampleVehicleTripbyHour (TTG-2)
- hotSoakEventByHour (TTG-3)
- hourlyColdSoakTankTemperature (TTG-1)
- temperature (ZoneMonthHour)
- tankTemperatureRiseTermA (TankTemperatureRise)
- tankTemperatureRiseTermB (TankTemperatureRise)

Outputs:

- intermediate table operatingTemperature, which adds the fields keyOnTemp and keyOffTemp to sampleVehicleTripByHour; and adds key field tankTemperatureGroup
- intermediate table hotSoakTemperature, which stores the tank temperatures for each hot soak event minute-by-minute; and adds key field tankTemperatureGroup

Calculation: operatingTemperature

For each vehicle:

for first (non-marker) trip in sampleVehicleTripByHour:

- keyOnTemp = hourlyColdSoakTemperature for that hour (linking of hourDayID in sampleVehicleTripByHour and hourID in hourlyColdSoakTemperature required)
- keyOffTemp = keyOnTemp + [(tankTemperatureRiseTermA + tankTemperatureRiseTermB * (95-hourlyColdSoakTemperature))/(1.2)]*((keyOffTime – keyOnTime)/60)

for subsequent trips:

if startOfTrip=0 (i.e. continuation of a TripID in a new hour):
- keyOnTemp = keyOffTemp from previous segment (defined as the same TripID in the previous hour)
- keyOffTemp calculated as above

if startOfTrip=1 (i.e. new trip)
- keyOnTemp = final soakTankTemperature from hotSoakTemperature where TripID of hotSoakTemperatures = priorTripID of operatingTemperature (i.e. the record in which hotSoakTime = keyOnTime for Trip ID)
- keyOffTemp calculated as above

Repeat with each new vehicle

Calculation: SoakTemperature this intermediate table expands hourlySoakEventByHour to store the hot soak tank temperature for each hot soak event minute-by-minute. Construct hotSoakTemperature as follows:

- key fields vehicleID, tripID, hourDayID from hourlySoakEventByHour
- using values of hotSoakBegin and hotSoakEnd for a given tripID, expand records to minute-by-minute with key field hotSoakTime (i.e. initial hotSoakTime = hotSoakBegin, final hotSoakTime = hotSoakEnd)
- coldSoakTemperature = hourlyColdSoakTemperature for the hour
- initialTankTemperature = keyOffTemp for Trip ID where endOfTrip=1, from operatingTemperature table
- For initial record in each hot soak event (defined by tripID):
    - soakTankTemperature = initialTankTemperature
    - tempDelta = temperature (this is ambient temperature from zoneMonthHour for that hour) – soakTankTemperature
    - opModeID = 150 if soakTankTemperature > coldSoakTemperature + 3, otherwise end

- For subsequent records in the same event:
  - soakTankTemperature = 1.4*SUM(tempDelta from initial record through most recent record)/60+ initialTankTemperature
  - tempDelta = temperature (this is ambient temperature from zoneMonthHour for that hour) – soakTankTemperature
  - opModeID = 150 if soakTankTemperature > coldSoakTemperature + 3, otherwise end

TTG-5 Calculate averageTankTemperature CMIT

This step is not required for mesoscale lookup.

Inputs:
- operatingTemperatures
- hotSoakTemperatures
- hourlyColdSoakTankTemperature

Output:
- averageTankTemperature CMIT (zoneID, monthID, hourDayID, tankTemperatureGroupID, opModeID, averageTankTemperature)

Calculation:
- averageTankTemperature, for a given hourDayID:
  - for opModeID=151 (cold soak) = hourlyColdSoakTankTemp for that hour
  - for each zoneID, monthID, hourDayID, and tankTemperatureGroupID averageTankTemperature can be calculated from the OperatingTemperature table as:

    averageTankTemperature =

        SUM((keyOffTime-keyOnTime) * (keyOnTemp + keyOffTemp)/2.0)
        /
         SUM(keyOffTime-keyOnTime)

  - for opModeID=150 (hot soak) = average of all soakTankTemperatures for trips with the same hourDayID

TTG-6 Calculate soakActivityFraction CMIT

Inputs:
- intermediate table HotSoakTemperature
- SampleVehicleDay table

Outputs:

- SoakActivityFraction CMIT

Calculation:

Fraction of cold soaking = cMinutes / (cMinutes + hMinutes)
Fraction of hot soaking = hMinutes/ (cMinutes + hMinutes)

Where:

cMinutes, on a dayID = total minutes of cold soaking for all vehicles in SampleVehicleDay = (60 * number of sample vehicles existing on the dayID) – oMinutes – hMinutes

oMinutes, on a dayID = total minutes of vehicle operation in the hourDayID = sum (keyOffTime-keyOnTime) for all trips in the hourDayID from VehicleTripByHour

hMinutes, on a dayID = total minutes of vehicle hot soaking in the hourDayID = count of records in HotSoakTemperature in the hour

TTG-7: Calculate Cold Soak Initial Hour Fractions

This step is not required for mesoscale lookup

Inputs:

- SampleVehicleTrip table
- SampleVehicleDay table
- VehicleTripByHour(TTG-2)
- HotSoakTemperature(TTG-4)

Outputs:

- ColdSoakInitialHourFraction CMIT table

Calculations:

First, make an intermediate table, ColdSoakInitialHourMinutes:

Key fields:

zoneID
monthID
tankTemperatureGroupID
vehID
dayID
hourID
initialHourID

Data field:
coldSoakInitialHourMinutes

The produce ColdSoakInitialHourMinutes as follows:

```
For each tankTemperatureGroupID
    For each vehID and dayID in SampleVehicle
        Initial Hour = 1
        For Each Hour

            Write two records into coldSoakInitialHourMinutes, one for
            minutes of cold soaking which began in the initial hour, which we
            will denote as X , and one for minutes of cold soaking which began
            in the current hour, which we will denote as Y.   The idea here is
            that all minutes of cold soaking in the current hour must either be
            an extension of cold soak periods which began at some single prior
            hourID, or must have begun in this hour.   This single prior hourID
            is what has been denoted above as Initial Hour.

            Find the earliest record pertaining to the vehicle, day, and hour in
            either VehicleTripByHour (TTG-2), based on keyOnTime, or in
            HotSoakTemperature (TTG-4) based on hotSoakTime.    There are
            three possibilities:

            1.  There is no first record because there are no records.

                X = 60.
                Y = 0.

            2.  The first record is a VehicleTrip.  (This may be the only record
                in the hour or there may be any number of subsequent hot
                soaks, and trips, the last of which may or may not run to the
                end of the hour.)

                X = keyOnTime of this first record – endOfHour + 60
                Y = 60-X-(number of HotSoakTemperature records) –
                sum(keyOffTime-keyOnTime) for all trips in the hour.
                Reset Initial Hour for subsequent calculations to be the current
                hour.

            3.  The first record is a hot soak minute (This may be the only
                record, or there may be any number of subsequent trips and hot
                soak minutes, the last of which may or may not run to the end
                of the hour.)

                X = 0
                Y = 60-(number of HotSoakTemperature records) –
                sum(keyOffTime-keyOnTime) for all trips in the hour.
                Reset InitialHour for subsequent calculations to be the current
                hour.

        Next Hour
```

Next Vehicle - Day

Next TankTemperatureGroup

TTG-7 can now produce the ColdSoakInitialHourFraction table by summing the ColdSoakInitialHourMinutes table across tankTemperatureGroups and sampleVehicles within SourceType, and normalizing to distributions which sum to unity.

The format of the ColdSoakInitialHourFraction table remains:

Key Fields:

sourceTypeID
zoneID
monthID
hourDayID
initialhourDayID

Data Field:

coldSoakInitialHourFraction

ignoring the detail that hourIDs and dayIDs are combined into hourDayIDs

coldSoakInitialHourFraction =
(sum of all minutes for vehIDs belonging to sourceTypeID in the zoneID, monthID, and dayID having the hourID and initialHourID) /
(sum of all minutes for vehIDs belonging to sourceTypeID in the zoneID, monthID and dayID having the hourID)

The calculation deliberately ignores the tankTemperatureGroup distinction, giving them all equal weight.

This key assumption behind this approach is that vehicles are soaking at all times when they are not hot soaking or operating. The predominant model of time embodied in the previous calculations, which this approach seeks to make rigorous, is that the world begins on the first hour of the day, and ends at the end of the day. E.g. there is no effort in the cold soak tank temperature calculations to “wrap-around” from the last hour of the day to the first hour of the day, and doing so now would result in a temperature discontinuity. (The single exception to this is that the TAG does consider some history from previous “real-world sampling days”, by allowing soak times for the first trip to be longer than if they had begun at midnight. This exception is quite limited and corresponds exactly to the use of the “marker trips” which are used for this single purpose.)

10.14. Tank Fuel Generator (TFG)

10.14.1. Functional Characteristics

This component executes at the County master looping level.

This component uses the fuel supply information from the MOVES database (which pertains to Fuel Formulations dispensed to sourceUseTypes in proportion to their FuelSupply marketShares) to produce the AverageTankGasoline table. It accounts for the effects of "comingling" ethanol with non-ethanol gasoline and for the "weathering" effect on RVP for in-use fuel.

The AverageTankGasoline table produced by this component is a CMIT. Any records already present in the MOVESExecution database, are not overwritten. This is done on an individual record basis.

10.14.2. Detailed Calculation Steps

TFG-1a: Calculate Average Pump Gasoline and Ethanol Blend Type

Inputs:

marketShare from FuelSupply (county, fuelYear, monthGroup, fuelFormulation)
ETOHvolume from FuelFormulation (fuelFormulation)
RVP from FuelFormulation (fuelFormulation)
fuelSubTypeID from FuelFormulation
fuelTypeID from FuelSubType

Outputs:

averageRVP (county, fuelYear, monthGroup)
tankAverageETOHVolume (county, fuelYear, monthGroup) This is stored as the ETOHVolume field of AverageTankGasoline.

Calculations:

averageRVP = For all FuelFormulations in county, fuel year & monthGroup where fuelType = "gasoline" (ie fuelTypeID = 1))
(Sum (RVP*marketshare)) / (Sum (marketshare))

tankAverageETOHVolume = For all Fuel Formulations in county, fuel year & monthgroup where fuelType = "gasoline" (ie fuelTypeID = 1))
(Sum (ETOH Volume*marketshare)) / (Sum (marketshare))

TFG-1b: Calculate Ethanol Market Share and Ethanol BlendType

Inputs:

marketShare from FuelSupply (county, fuelYear, monthGroup, fuelFormulation)
fuelSubTypeID from FuelFormulation
fuelTypeID from FuelSubType

Outputs:

gasoholMarketShare (countyID, fuelYearID, monthGroupID)
ethanolBlendType (county, fuelYear, monthGroup)

Calculation:

```
gasoholMarketShare:    For all FuelFormulations in county, fuelyear &
     monthgroup where ETOHVolume >= 4
             gasoholMarketShare =Sum (marketShare)

lowETOHRVP:   For all FuelFormulations in county, fuel year & monthgroup
     WHERE fuelType = “gasoline” (ie fuelTypeID = 1) and ETOHVolume <4

             IF ( sum (marketshare) = 0,
                  lowETOHRVP=AverageRVP

             ELSE
                      lowETOHRVP=(Sum (RVP*marketshare)) / (Sum
             (marketshare))


highETOHRVP:   For all FuelFormulations in county, fuel year &
     monthgroup WHERE fuelType = “gasoline” (ie fuelTypeID = 1)) and
     ETOHVolume >= 4

             IF gasoholMarketShare = 0,
                  highETOHRVP = AverageRVP

             ELSE
                      highETOHRVP =(Sum (RVP*marketshare)) /
             gasoholMarketShare


ethanolBlendType:
             IF absolute value (highETOHRVP –lowETOHRVP) <= 0.2,
                      ethanolBlendType =“Match”
             ELSE  ethanolBlendType = “Splash”
```

TFG-1c: Calculate Commingled Tank Fuel RVP

Inputs:

gasoholMarketShare (countyID, fuelYearID, monthGroupID) from TFG-1b
averageRVP (countyID, fuelYearID, monthGroupID) from TFG-1a

Commingling Lookup (stored in program)

LookupMarketShare	Commingling RVP Factor
0.0	1.000
0.1	1.016
0.2	1.028
0.3	1.035
0.4	1.039
0.5	1.040
0.6	1.038
0.7	1.034
0.8	1.027
0.9	1.018
1.0	1.000

Outputs:

commingledRVP (countyID, fuelYearID, monthGroupID)

Calculation:

comminglingFactor (countyID, fuelyearID, monthgroupID) = lookup from table using smallest value of "LookupMarketShare" that is greater than or equal to the gasoholMarketShare.

commingledRVP = averageRVP * comminglingFactor

TFG-2: Weathered RVP

TFG-2a: Calculate "EvapTemp" by zoneID, MonthGroupID

Inputs:

temperature (zoneID, hourID monthgroupID)
zoneID from masterloopcontext

Outputs:

zoneEvapTemp (zoneID, monthgroupID)

Calculation :

```
zoneMin(zoneID, monthgroupID) = MIN (temperature(zoneID,
    monthgroupID, hourID))

zoneMax (zoneID, monthgroupID) = MAX(temperature(zoneID,
    monthgroupID, hourID)

zoneEvapTemp =
    IF zoneMax <40  or zoneMax-zoneMin <=0, (zoneMin+zoneMax)/2
    ELSE  zoneEvapTemp(zoneID, monthGroupID) =
        -1.7474+1.029*zoneMin+ 0.99202* (zoneMax-zoneMin)-
        0.0025173*zoneMin* (zoneMax-zoneMin)
```

TFG-2b: Calculate ratio of weathering loss for gasoline by Zone, Year & Month at actual ambient temperatures relative to a diurnal swing of 72-96 F

Inputs:

zoneEvapTemp (zoneID, monthgroupID) from previous step
commingledRVP (countyID, fuelYearID, monthgroupID) from TFG-1c
zone(countyID, zoneID)

Outputs:

ratioGasolineRVPLoss(zoneID, fuelYearID, monthgroupID)

Calculation :

```
ratioGasolineRVPLoss =MAX (0,  [-2.4908 + 0.026196 * zoneEvapTemp +
    0.00076898 * zoneEvapTemp * commingledRVP]/[-0.0860 + 0.070592 *
    commingledRVP ] )
```

TFG-2c: Calculate weathering loss for average fuel for standard temperatures

Inputs:

ethanolBlendType (county, fuelYear, monthGroup) from TFG-1a
gasoholMarketShare (countyID, fuelYearID, monthGroupID) from TFG-1b

Outputs:

avgWeatheringConstant (countyID, fuelYearID, monthGroupID)

Calculations:

```
IF ethanolBlendType = “Match”, avgWeatheringConstant = 0.049 – 0.0034 *
    gasoholMarketShare
ELSE  avgWeatheringConstant = 0.049 – 0.0116 * gasoholMarketShare
```

TFG-2d: Calculate weathered RVP for county-average fuel adjusted for zone temperatures

Inputs:

ratioGasolineRVPLoss (zoneID, fuelYearID, monthgroupID) from TFG-2b
avgWeatheringConstant (countyID, fuelYearID, monthGroupID)
from previous step
commingledRVP (countyID, fuelYearID, monthGroupID) from TFG-1c
zone(countyID, zoneID)

Outputs:

tankAverageGasolineRVP(zoneID, fuelYearID, monthgroupID)

Calculation :

tankAverageGasolineRVP (zoneID) = commingledRVP(countyID) * (1 – ratioGasolineRVPLoss (zoneID) * avgWeatheringConstant (countyID))
This is stored as the RVP field of AverageTankGasoline

10.15. Evaporative OperatingModeDistributionGenerator (EvapOMDG)

The evaporative operating mode distribution generator populates the core model OpModeDistribution table for the evaporative processes (Tank Vapor Venting, Fuel Liquid Leaking and Fuel Permeation)   While the Permeation process does not distinguish emission rates by these operating modes it applies the temperature adjustment separately by operating mode and therefore needs this operating mode distribution to weight values together.

Inputs to this calculation are the SHO and SourceHours tables and the SoakActivityFraction table produced by the TTG. Its only output is the OpModeDistribution table.

Since the operating mode distributions for the principal evaporative processes depend upon ambient temperature and therefore upon monthID, but the OperatingModeDistribution table does not include monthID as a key field, it is logically necessary, apart from any performance considerations, for this MasterLoopable component to execute at or below the MONTH level, and as implemented it does sign up at the MONTH level.   This means, as regards locations, that it executes for each Link. The algorithm used is intended to operate correctly if some vehicle operation is allocated to the off-network roadtype.

For links which represent actual highways the operating mode distribution is always “100% operating”.   If running for mesoscale lookup this is all that needs to be done.

When links which represent off highway locations are included in the run specification, this OMDG determines fractions for “operating”, and for the other evaporative process operating modes, which in the current database are “hot soaking” and “cold soaking”, which sum to unity as follows:

1. Determine the fraction of operating as a ratio of SHO in the SHO table to sourceHours in the SourceHours table. Since these tables have the same structure, and since this component is executing for a single linkID and monthID, this calculation is straightforward. The ageID, which is present in the SHO and SourceHours tables but not in OperatingModeDistribution, just needs to be summed out. If the source hours denominator is missing or zero, then no output distribution is produced.

2. Convert the soakActivityFractions for the operating modes other than "operating" (currently "hot soaking" and "cold soaking") to opModeFractions which take into account the fraction of operating.

$$\text{opModeFraction}_{\text{opModeID}} = \text{soakActivityFraction}_{\text{opModeID}} * (1.0\text{-fraction of operating})$$

The special treatment of the "operating" mode is "hard-coded" into this generator. The other modes, however, are treated in a general fashion, e.g.. the TTG does not assume that there are only two other modes, or that these modes have certain names.

10.16. Alternative Vehicle Fuels and Technologies (AVFT) Strategy

The Alternative Vehicle Fuels & Technologies Strategy is the first implementation of an internal control strategy. This control strategy allows the user to input penetration rates (i.e. sales fractions) by model for a broad range of advanced technologies, through model year 2050. It is thus a key element in the ability of MOVES to perform "what-if" analysis.

The AVFT has two components. One component is incorporated into the MOVES GUI and allows the user to create specifications for replacement inputs for the FuelEngFraction table and, indirectly, the SizeWeightFraction and RegClassFraction tables. A second portion, the actual InternalControlStrategy MasterLoopable object, uses these specifications to produce these replacement tables prior to the running of the SourceBinGenerator. This causes changes to the SourceBinDistributions and the eventual MOVES output.

The user has the option to save the specifications (AVFTspecs) for the replacement input tables. Most of the content of these specifications is the data to fill a large FuelEngFraction table in the MOVES database. The user is able to save the AVFTSpecs and to load and modify a saved AVFTSpec. A RunSpec may have an associated AVFTSpec; but AVFTspecs may also exist independently of RunSpecs.

10.16.1. Functional Characteristics of the AVFT GUI Component:

a. "Strategies" appears on the RunSpec Navigation List immediately after "Manage Input Data Sets". "Alternative Vehicle Fuels & Technologies" is a sub-section. Control Strategies are not required, thus the initial status icon for Strategies (and its sub-sections) is the green check or the yellow squiggles depending on whether the Run Spec currently includes a control strategy that modifies the default data.

b. An AVFT file management panel appears when the user selects "Alternative Vehicle Fuels & Technologies" from the navigation list. The panel includes:

The name of the associated AVFTSpec (if one exists), or an indicator that no AVFTSpec exists and that default Fuel/Technology fractions are being used. The name displayed changes in response to the buttons below. No more than one AVFTSpec may be associated with a given RunSpec.

A “New” button to create a new AVFTSpec. (This has the same effect as returning to default and editing.)

An “Edit” button to view & modify the associated AVFTSpec.

An “Import” button to load/associate an existing AVFTSpec. This allows the user to browse for exported AVFTSpecs and to select one.

An “Export” button allows the user to save a AVFTSpec independent of the RunSpec. (Note: Saving the RunSpec itself includes saving the associated AVFTSpec. Executing the RunSpec after editing an AVFTSpec but without saving includes execution of the new AVFTSpec.)

A “Delete” button to remove an associated AVFTSpec and revert to MOVES defaults.

A “Cancel” button to close the panel without changes.

c. An AVFT Edit panel appears next to the AVFT file management panel when the user selects “Edit” or “New” in that panel. The Edit panel contains a description button and a details subpanel.

Description Button: The user may (optionally) enter a short text description of the spec.

Details Panel:

1. The user may select a SourceType from a drop down list of all SourceTypes. The panel then displays a table of the Fuel/Engine Technology fractions from the associated AVFTSpec (or MOVES default DB FuelEngFraction table) for that SourceType for model years 2001-and-later. (The default FuelEngFraction table need not include all model years and the AVFTSpec may include similarly limited model years.   The panel shows the model years provided and allows users to add additional years if desired (see below).   In any case, the MOVES

SourceBinGenerator extrapolates needed future years by repeating values for the last year provided.)

2. Fuel/EngineTechnologies are listed as columns. Model years are listed as rows. The first value in each row is the model year. Fractions are sorted into categories as listed in the in the FuelEngTechAssoc table for that SourceType. The categories are displayed in the order indicated by the CategoryDisplayOrder field of the FuelEngTechAssoc table. Within a category, fractions are listed in ascending order by FuelTypeID*100 +EngTechID. The final value in each row is the sum of all FuelEngFractions for that source/type model year (so the user can see if the values sum to 1 as desired.) This panel normally fills the entire screen, with scroll bars to display any portions that don't fit.

3. The initial view displays fractions aggregated in the categories listed in the FuelEngTechAssoc table for that SourceType. The user can display or hide detailed views that display the fractions for individual Fuel/EngineTechnology combinations. Categories with only one member are not considered "aggregate" (ie, they may be modified in all views and there is no "detailed view" to display or hide). Columns of aggregate fractions should be labelled with the category name. Columns of non-aggregrate fractions should be labeled with the name of the FuelType and the Engine Technology.

4. The user may select an "Add Model Years" button that adds one or more model year rows at the bottom. The user has an option to specify how many rows to add with this feature but the sytem currently will not add model years beyond 2050. In each new row, the model year field is the next consecutive model year. The initial values for the FuelEngFractions in the new row equal those of the previous model year.

5. The user may replace any (non-aggregate) fraction in the table. Category aggregate fractions are grayed-out to indicate that they must be modified in a detail view. Fractions must be between 0 and 1, inclusive. The GUI does not allow the user to enter values <0 or >1.

6. The user may manually assure that the fractions sum to 1 for a sourcetype/model year or may choose a "Normalize" button that will keep the proportions of the input fractions but adjust so they sum to 1. The AVFTSpec does not "Export" and the RunSpec does not "Save" or "Execute" until all sourcetypes/model years sum to 1. Error messages are provided to the user as needed.

7. If the FuelEngTechAssoc table lists only one FuelType/EngineTechnology combination for the SourceType (currently true for motorcycles), there is no action for the user to take. The GUI displays the appropriate detail screen (which should have only one Fuel/EngineTechnology column), and displays a message: "Only one FuelType/EngineTechnology combination is supported for (SourceType Name). No alternate values allowed."

10.16.2. Functional Characteristics of the AVFT MasterLoopable

a. The AVFT processor is MasterLoopable; executing after the InputDataManager and before the Generators. It is instantiated under the same conditions as the SourceBinGenerator and executes at the emission process master loop level.

b. The AVFT FuelEngFractions are processed to replace the default fractions in FuelEngFractions in the Execution Location Database. In particular, if the specification provides inputs for a given SourceType/ModelYear, those inputs replace all values for that SourceType/ModelYear. Empty records in both the original and replacement tables (ie, records for which the implied fuelEngFraction =0) are taken into account.

c. The default RegClassFractions and SizeWeightFractions are modified such that the aggregate (across FuelType/EngineTechnology) RegClassFractions for the SourceType/ModelYear remain the same after application of the AVFT. The new RegClassFractions and SizeWeightFractions conserve the original fractions for vehicles with unchanged FuelEng characteristics and proportionally distribute the remaining fractions among the "changed vehicles".

   This calculation is specified in detail below for RegClassFraction. The algorithm for SizeWeightFractions is analogous, except the individual Regulatory Classes are replaced by combinations of EngSizeID and WeightClassID.

Compute New Reg Class Fractions

For each source type and model year, and for each supported FuelEngTech combination (i), either there exists an original **FuelEngFraction(i)** and a **NewFuelEngFraction(i)**, or it is implied that the FuelEngFraction(i) or NewFuelEngFraction(i) =0.

```
    Compute DeltaFuelEngFraction(i) as NewFuelEngFraction(i)-
    FuelEngFraction(i).

    Compute ProportionOfNew(i) as
        IF  DeltaFuelEngFraction(i)>=0
            Proportion =0
        ELSE
            ProportionOfNew =
                DeltaFuelEngFraction(i) /
                Sum (over all i where DeltaFuelEngFraction <0)
                (DeltaFuelEngFraction(i)).

    Compute PortionOld(i) as
        IF NewFuelEngFraction(i)=0,
            PortionOld(i) =0
        ELSE PortionOld = Min (1,
        FuelEngFraction(i)/NewFuelEngFraction(i))
```

For each SourceType & Model Year there is a set of original RegClassFractions(i,j), where “i” indicates the associated FuelEng combination and “j” indicates the RegClass. These are used with the results of the above calculations to create NewRegClassFractions(i,j,).

```
For each j
    For each i
        Compute NewRegClassFraction(i,j) as:

            IF NewFuelEngFraction(i)= 0,
                NewRegClassFraction(i,j) =0
                (Doesn't need to be in database)
            ELSE
                NewRegClassFraction(i,j) =
                    PortionOld(i) * RegClassFraction (i,j) +
                    [1-PortionOld(i)) *
                    (Sum(over all i) (ProportionOfNew(i)
                    *RegClassFraction(i,j))]
```

10.17. Energy Consumption Calculator (ECC)

The energy consumption calculator calculates energy consumption (total, petroleum-based and fossil-based) for four processes: running, start, extended idle and well-to-pump, for each source type on each roadway type in DRAFT MOVES2009. It uses input from CMITs produced by the total activity, operating mode distribution, source bin, and meteorology generators, and calculates the quantities of energy consumed in the form of the MOVES output database.

Note: This functional calculator is actually implemented as two MOVES EmissionCalculator classes: an "EnergyConsumptionCalculator" for the running, start, and extended idle processes, and a "WellToPumpProcessor" which "chains" onto the EnergyConsumptionCalculator if theWellToPump process is required.

10.17.1. Overview of Calculation Steps

Table 10-7. Overview of Emission Consumption Calculator

Step	
ECCP-1	Preliminary calculations
ECCP-1a	*Calculate petroleum and fossil energy fractions*
ECCP-1b	*Convert age to model year for analysis year*
ECCP-2	Calculate adjustments
ECCP-2a	*Calculate air conditioning adjustment*
ECCP-2b	*Calculate temperature adjustment*
ECCP-2c	*Calculate fuel adjustment*
ECCP-3	Calculate running energy consumption
ECCP-4	Calculate start energy consumption
ECCP-5	Calculate extended idle energy consumption
ECCP-6	Calculate well-to-pump energy consumption
ECCP-6a	*Calculate pump-to-wheel energy consumption*
ECCP-6b	*Interpolate well-to-pump factor by FuelSubType in analysis year*
ECCP-6c	*Calculate well-to-pump factor by FuelType in analysis year*
ECCP-6d	*Calculate well-to-pump energy consumption*

The level of aggregation for each calculation is dictated by the key fields of the input and output variables. The result of energy quantities in steps 2, 3, 4 and 5 are at the level dictated by the DRAFT MOVES2009 output database design: calendar year, month, day, hour, county, zone, link, pollutant, process, source type, fuel type, model year, and road type.

If the EnergyConsumptionCalculator encounters a missing value when performing a calculation, the result of the calculation is considered as missing. Records for which the results are missing are not represented by a value of zero but are left out of the database.

The EnergyConsumptionCalculator signs up for the Master Loop at the year level which means that it executes for each location (link) for each calendar year. It signs up for the running, start and extended idle processes to the extent they are called for by the run specification.

10.17.2. Detailed Steps

ECCP-1: Preliminary Calculations

Step 1a: Calculate petroleum and fossil energy fractions by fuel type

Since petroleum and fossil energy fractions vary by fuel subtype, this step is required to aggregate these fractions up to the fuel type level.

Input Variables:

MarketShare (County, Year, MonthGroup, FuelSubType,)
From FuelSupply table
Note: If record not found in database, default value of 1.0 is used if fuelSubtypeID ends with 0, otherwise default value of 0.0 is used.
FuelSubtypePetroleumFraction (FuelSubType)
From FuelSubtype table
FuelSubtypeFossilFraction (FuelSubType)
From FuelSubtype table

Output Variable:

PetroleumFraction (County, Year, Month, FuelType)
FossilFraction (County, Year, Month, FuelType)

Calculation:

PetroleumFraction=

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{FuelSubtypePetroleumFraction}$$

FossilFraction =

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{FuelSubtypeFossilFraction}$$

Step 1b: Convert Age to Model Year For Analysis Year

Input Variables:

YearID
AgeID

Output Variable:

ModelYearID

Calculation:

ModelYearID = YearID - AgeID

ECCP-2: Calculate adjustments

Step 2a: Calculate air conditioning adjustment

Preliminary calculation (1): ACOnFraction

This step calculates the fraction of time the AC compressor is engaged, as a function of Heat Index

Input Variables:

HeatIndex (Zone, Month, Hour)
  From ZoneMonthHour table
ACActivityTermA (MonthGroup, Hour)
  From MonthGroupHour table
ACActivityTermB (MonthGroup, Hour)
  From MonthGroupHour table
ACActivityTermC (MonthGroup, Hour)
  From MonthGroupHour table

Calculation:

ACOnFraction (Zone, Month, Hour) =
  (ACActivityTermA+ACActivityTermB*HeatIndex
  +ACActivityTermC*HeatIndex^2)

If ACOnFraction<0, set to 0
If ACOnFraction>1, set to 1

Preliminary calculation (2): ACActivityFraction

This step calculates the overall A/C activity fraction, which accounts for the A/C on fraction, the penetration of A/C in the fleet and the fraction of those A/C systems that are functioning.

Input Variable

ACOnFraction (Zone, Month, Hour) from previous calculation

ACPenetrationFraction (SourceType, ModelYear)
From SourcetypeModelYear table
FunctioningACFraction (SourceType, Age)
From SourceTypeAge table

Calculation

ACActivityFraction (Zone, Month, Hour, SourceType, ModelYear) =
ACOnFraction
* ACPenetrationFraction
*FunctioningACFraction

Calculate AC Adjustment

Input Variables:

ACActivityFraction (Zone, Month, Hour, SourceType, ModelYear)
From previous calculation
FullACAdjustment (PolProcess, SourceType, OpMode)
From FullACAdjustment table

Calculation:

ACAdjustment (Zone, Month, Hour, SourceType, ModelYear, PolProcess, OpMode) = 1 + ((FullACAdjustment –1)*ACActivityAdjustment)

Step-2b: Calculate temperature adjustment

Input Variables:

Temperature (Zone, Month, Hour)
From ZoneMonthHour table
TempAdjustTermA (PolProcess, SourceType, FuelType)
From TemperatureAdjustment table
TempAdjustTermB (PolProcess, SourceType, FuelType)
From TemperatureAdjustment table
TempAdjustTermC (PolProcess, SourceType, FuelType)
From TemperatureAdjustment table
Note: If temp adjustment terms are not found in database, default values of 0.0 are used.

Output Variables:

TempAdjustment (PolProcess, Zone, Month, Hour, SourceType, FuelType)

Calculation:

TempAdjustment =
1 + TempAdjustTermA * (Temperature-75)
+ TempAdjustTermB * $(\text{Temperature-75})^2$

Step 2c: Calculate fuel adjustment

Input Variables:

FuelAdjustment (SourceType, PolProcess, FuelSubType)
MarketShare (County, Year, MonthGroup, FuelSubType)

Output Variable:

FuelAdjustmentbyType (County, Year, MonthGroup,SourceType, PolProcess, FuelType)

Calculation:

FuelAdjustmentbyType =

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{FuelAdjustment}$$

ECCP-3: Calculate running energy consumption

Step 3a: Aggregate Base Emission Rates to SourceType/ Fuel Type/Model Year/ Operating Mode level

Input Variables:

MeanBaseRate (SourceBin, PolProcess, OpMode)
From EmissionRate table
SourceBinActivityFraction (SourceType, ModelYear, SourceBin, PolProcess)
From SourceBinDistribution Table

Calculation:

MeanBaseRatebyType (SourceType, FuelType, ModelYear, PolProcess, OpMode) =

$$\sum_{SourceBin=1}^{No.SourceBinswithinFuelType,ModelYear} SourceBinActivityFraction * MeanBaseRate$$

Step 3b: Aggregate emission rates to SourceType level, Apply A/C Adjustment

Input Variables:

MeanBaseRatebyType (SourceType, FuelType, ModelYear, PolProcess, OpMode) From previous calculation
OpModeFraction (SourceType, Link, HourDay, PolProcess, OpMode) From OPModeDistribution table
ACAdjustment (Zone, Month, Hour, SourceType, ModelYear PolProcess, OpMode) From ECCP- 2a

Calculation:

Total Energy:

SourceTypeEnergy (Zone, Month, HourDay, SourceType, FuelType, ModelYear, Link, PolProcess) =

$$\sum_{OpModeBin=1}^{No.OpModeBins} OpModeFraction * MeanBaseRatebyType * ACAdjustment$$

Step 3c: Calculate Total Energy

Input Variables:

SourceTypeEnergy (Zone, Month, HourDay, SourceType, FuelType, ModelYear, Link, PolProcess) from previous calculation
SCCVtypeFraction(SourceType, ModelYear, FuelType, SCCVtype) From SCCVtypeDistribution table
SHO (SourceType, Age, Link, Hour, Day, Month, Year) From SHO Table
IMOBDAdjustment (SourceType, County, Year, Age, FuelType) From IMOBDAdjustment table.
Note: If IMOBD adjustment value not found in database, a default value of 1.0 is used.
TempAdjustment (ECCP-2b, existing code)
FuelAdjustment (ECCP-2c, existing code)
PetroleumFraction (ECCP-1a, existing code)
FossilFraction (ECCP-1a, existing code)

Calculation:

Total Energy

EmissionQuant = SCCVtypeFraction *

$$\sum_{SourceBin=1}^{No.SourceTypes} SHO * SourceTypeEnergy * FuelAdjustment * TempAdjustment * IMOBDAdjustment$$

Petroleum Energy:

EmissionQuant =
EmissionQuant (Running, Total Energy)
* PetroleumFraction

Fossil Energy:

EmissionQuant =
EmissionQuant (Running, Total Energy)
* FossilFraction

ECCP-4 & 5: Calculate start and extended idle energy consumption

The equations are exactly the same as is done for ECCP-3 above. The only difference is that, under Step c, "SHO" is replaced by "Starts" for Start and "ExtendedIdleHours" for Extended Idle.

ECCP-6: Calculate total, petroleum, and fossil energy consumption for well-to-pump (WTP)

For all pollutants, well-to-pump energy and emissions are calculated as a function of pump-to-wheel total energy consumption (i.e. the sum of running, start and extended idle energy)

Step-6a: Calculate total pump-to-wheel(PTW) energy consumption

Inputs:

EmissionQuant(…FuelType, PolProcess = TotalEnergy; Running, Start, Extended Idle)

Outputs:

PTWEmissionQuant (FuelType, Total Energy)

Calculation:

PTWEmissionQuant =
EmissionQuant (Running, Total Energy)
+ EmissionQuant (Start, Total Energy)
+ EmissionQuant (Extended Idle, Total Energy)

Step-6b: Interpolate well-to-pump factorby FuelSubType in analysis year

Inputs:

WTPFactor (Pollutant, FuelSubType for next highest and next lowest in WTP factor database, relative to YearID)
(Note : WTPFactor represents the field EmissionRate located in the GreetWellToPump table of the MOVES database.)

Outputs:

WTPFactor (Pollutant, FuelSubType, YearID)

Calculation:

Linear interpolation using "bracketing" years in WTP database:
WTPFactor = WTPFactor (LoYearID) +
(WTPFactor (HiYearID) – WTPFactor (LoYearID)) *
((YearID – LoYearID) / (HiYearID – LoYearID))

Step-6c: Calculate well-to-pump factor by FuelType in analysis year

Inputs Variables:

WTPFactor (Step 6a)
MarketShare (County, Year, MonthGroup, FuelSubType,)

Output Variables:

WTPFactorbyFuelType(Pollutant, FuelType, YearID)

Calculation:

WTPFactorbyFuelType =

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{WTPFactor}$$

Step-6d: Calculate well-to-pump energy consumption

Inputs:

PTWEmissionQuant (Step 6a)
WTPFactorbyFuelType (Step 6c)

Outputs:

EmissionQuant (PolProcess = Well-To-Pump; Total Energy, Petroleum Energy, Fossil Energy)

Calculation:

EmissionQuant =
PTWEmissionQuant * WTPFactorbyFuelType

10.18. Distance Calculator

The TotalActivityGenerator produces distance information in the SHO table. The DistanceCalculator reports vehicle travel distance information based on these SHO table values, if requested by the RunSpec, in the MOVESActivityOutput table of the MOVES output database. Note that MOVES stores distance within the MOVESMesoscaleActivityOutput table for runs using the Mesoscale Lookup scale

10.18.1. Algorithm Overview

The "total" distance in the SHO table must be disaggregated to the often more detailed level of the MOVES output. This is a "calculator-like" function and so is performed by an EmissionCalculator class despite the fact that "distance" is not a "pollutant".

This calculator is instantiated whenever distance output is requested by the RunSpec, which implies that some pollutant-process involving the running process has been selected, and signs up for the "Running Exhaust" emission process at the "Year" level of the MasterLoop, the same level at which the TAG operates.

10.18.2. Distance calculation

If the "Distance Traveled" output is requested by the RunSpec, (which also implies that some pollutant has been selected for the Running process), the distance field in the SHO table is calculated by the TotalActivityGenerator (TAG). Otherwise distance is output by that generator as NULL.

Within the distance calculator itself, there is a single calculation step:

DC-1 : Allocate Distance to Finest MOVES Output Level

Input Variables:

SCCVtypeID(SCC) from SCC table
SCCVtypeFraction(sourceTypeModelYearID, fuelTypeID, SCCVtypeID)
From SCCVtypeDistribution table
sourceBinActivityFraction (sourceTypeModelYearID, polProcessID, sourceBinID) from SourceBinDistribution table
distance(sourceTypeID, linkID, ageID, yearID, monthID, hourDayID) from SHO table

Output Variable:

MOVESActivityOutput.distance(runID, yearID, monthID, dayID, hourID,
stateID, countyID, zoneID, linkID, sourceTypeID, fuelTypeID,
modelYearID, roadTypeID, SCC)

Conceptual Calculation:

MOVESActivityOutput.distance =
SCCVtypeFraction* distance *
[sum over all sourceBins in fuelTypeID (sourceBinActivityFraction)]

Several detailed considerations must be kept in mind as these calculations are performed.

One consideration is just that model years are converted to ageID values by the relationship: modelYearID = yearID - ageID.

Another consideration is that SHO table values are reduced by the SCCVtypeFraction, because the raw MOVESOutput data is for SCC-SourceUseType intersections. This factor is looked up in the SCCVtypeDistributionTable based on sourceTypeID, modelYearID, and fuelTypeID. In general this yields multiple values of SCCVtype and SCCVtypeFraction. These values of SCCVtype are used in combination with RoadType to determine SCC values from the SCC table.

The most significant complexity is that SHO table distance values are reduced by a fraction representing the portion of this “total” activity occurring for the specific fuelTypeID. This is accomplished by determining the portion of the SourceBinDistribution which involves the given fuelTypeID.

The table 10-8 illustrates the structural differences between the input to this calculation and the output it must produce:

Table 10-8. Structural Differences between DistanceCalculator Input/Output

Key Field	**SHO.distance**	**Most detailed <u>MOVESOutput</u>**
MOVESOutputRowID		Yes, autoincremented
runID		Yes, but constant for a run
yearID	Yes	Yes
monthID	Yes	Yes
dayID	Yes	Yes
hourID	Yes	Yes
stateID		Yes, but redundant with county
countyID		Yes, but redundant with zone
zoneID	Yes	Yes
roadTypeID	Yes	Yes
pollutantID		Yes, but constant for this calculation
processID		Yes, but constant for this calculation
sourceTypeID	Yes	Yes
fuelTypeID	No	Yes
modelYearID	as ageID	Yes
SCC	No	Yes

10.19. Methane (CH4) and Nitrous Oxide (N2O) Calculator

MOVES includes two EmissionCalculator classes, CH4N2OrunningStartCalculator and CH4N2OWTPCalculator, which calculate Methane (CH_4) and Nitrous Oxide (N_2O) emissions for three processes (running, start, and well-to-pump) for each source type on each roadway type modeled. These calculators use input from CMITs produced by the total activity and source bin distribution generators, and calculate the quantities of these emissions in the form required by the MOVESOutput database.

These calculators subscribe to the MasterLoop at the year level, which means they are executed once for each location (linked) for each year.

When these EmissionCalculators encounter a missing input value in the MOVES database, the result of the calculation are considered as missing. Such results are left out of the database and are not represented in the MOVES output by a value of zero or some other numeric value or character.

The calculation steps are described below. These steps are the same for CH_4 and N_2O.

10.19.1. Step 1: Calculate Running Emissions

Input:

SHO (SourceType, Age, Link, Hour, Day, Month, Year)
SourceBinActivityFraction (SourceType, ModelYear, PolProcess , SourceBin)
MeanBaseRate (PolProcess, SourceBin, OpMode)

Output:

EmissionQuant (SourceType, Pollutant, Process, FuelType….see MOVES output database design)

Calculation:

EmissionQuant = SHO * SourceBinActivityFraction * MeanBaseRate

10.19.2.Step 2: Calculate Start Emissions

Input:

Starts (SourceType, Age, Zone, Hour, Day, Month, Year)
SourceBinActivityFraction (SourceType, ModelYear, PolProcess , SourceBin)
MeanBaseRate (PolProcess, SourceBin, OpMode)

Output:

EmissionQuant (SourceType, Pollutant, Process, FuelType….see MOVES output database design)

Calculation:

EmissionQuant = Starts * SourceBinActivityFraction * MeanBaseRate

10.19.3. Step 3: Calculate Well-To-Pump Emissions

Step 3a: Calculate well-to-pump factor by FuelType in analysis year

Inputs Variables:

EmissionRate from GREETWellToPump (Year, Pollutant, FuelSubType)
MarketShare (County, Year, MonthGroup, FuelSubType,)

Output Variables:

WTPFactorbyFuelType(Year, Pollutant, FuelType)

Calculation:

WTPFactorbyFuelType =

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{WTPFactor}$$

Step 3b: Calculate Well-To-Pump Emissions

Input:

PTWEmissionQuant for total energy (Step 6a of Total Energy Calculator)
WTPFactorbyFuelType for CH4/N2O (Step 3a above)

Output:

EmissionQuant (SourceType, Pollutant, Process, FuelType….see MOVES output database design)

Calculation:

EmissionQuant = PTWEmissionQuant * WTPFactorbyFuelType

10.20. Atmospheric CO2 and CO2-Equivalent Calculator

10.20.1. General Description

This task involves performs the calculation of two pollutants: Atmospheric CO_2 and CO_2 Equivalent for all of the exhaust emission processes: Start, Running and Extended Idling. The calculations depend on the previous calculation of Total Energy, CH_4 and N_2O in their respective calculators. So this calculator is “chained” to those.

10.20.2. Detailed Calculation Steps

Step 1: Calculate CO2 and CO2-Equivalent from Total Energy

Preliminary Calculation: Compute Carbon Content and Oxidation Fraction by FuelType

Inputs:

- CarbonContent (grams per KJ) by fuel subtype
- OxidationFraction by fuel subtype
- MarketShare (County, Year, MonthGroup, FuelSubType) From FuelSupply table

Outputs:

- CarbonContent by FuelType
- OxidationFraction by FuelType

Calculations:

CarbonContent by FuelType

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{CarbonContent}(\textit{FuelSubtype})$$

OxidationFraction by FuelType =

$$\sum_{n=1}^{no.ofFuelSubTypeswithinFuelType} \text{MarketShare} * \text{OxidationFraction(FuelSubtype)}$$

Calculate Atmospheric CO2 from Total Energy

Inputs:

- Total Energy (KJ)
- CarbonContent by FuelType (previous calculation)
- Oxidation Fraction by FuelType (previous calculation)

Output:

- Atmospheric CO_2 (grams)

Calculation:

Atmospheric CO_2 = Total Energy * Oxidation Fraction * Carbon Content * (44/12)

Step 2: Calculate CO_2 Equivalent

Inputs:

- Atmospheric CO_2, CH_4, N_2O (grams)
- 100 year Global Warming Potentials by pollutant (GWP)

Output:

- CO_2 Equivalent (grams)

Calculation:

CO_2 Equivalent = $CO_2 * GWP_{CO2} + CH_4 * GWP_{CH4} + N_2O * GWP_{N2O}$

10.21. Criteria Pollutant Running EmissionCalculator (CREC)

10.21.1. General Description

This EmissionCalculator signs up with the MOVES master looping mechanism at the Year level for the running process, which means that a single execution produces results for one Link and one Year.

The logical steps of the calculation can be described as follows:

CREC-1 Calculate emission rates which account for I/M programs

This step has two substeps:

The first substep calculates an intermediate IMAdjustment table from the contents of the IMCoverage table for a particular zone and year. It puts the information into a usable form. The form of the IMCoverage table, in particular the use of model year ranges as non-key fields and the presence of information only where I/M programs exist, was highly constrained by overall table size considerations. To be used in subsequent steps the information needs to be into a more usable form, which this substep accomplishes.

The second sub-step combines information from the EmissionRateByAge table and the IMAdjustment table produced in the first substep to produce an intermediate EmissionRateWithIM table. The function here is to weight the meanBaseRate and meanBaseRateIM fields together using the IM fractions calculated in the previous substep.

CREC-2 Calculate fuel-supply-weighted fuel adjustment factors

This step has two substeps:

The first substep uses the FuelAdjustment and County tables to produce an intermediate CountyFuelAdjustment table specific to the county containing the link being executed. It resolves the GPA and non-GPA fuel adjustment factors into a single factor based on the GPAFract value in the County table. It is still at the fuelFormulation level.

The second substep uses CountyFuelAdjustment and the FuelSupply table to produce an intermediate FuelSupplyAdjustment table of fuel adjustment factors at the fuelTypeID level.

CREC-3 Calculate temperature adjustment factors

This step uses the TemperatureAdjustment table along with temperature information from the ZoneMonthHour table to produce an intermediate METAdjustment table.

CREC-4 Calculate air conditioning (AC) adjustment factors

This step involves four substeps.

The first substep uses the ACActivity terms from the MonthGroupHour table and the heat index information from the ZoneMonthHour table to calculate the fraction of time the AC compressor is engaged.

The second substep calculates the AC activity fraction, which accounts for the AC on fraction from the first substep, the penetration of AC in the fleet (from the SourceTypeModelYear table) and the fraction of those AC systems that are functional (from the SourceTypeAge table.)

The third substep calculates operating-mode-weighted FullACAdjustment factors from the contents of the FullACAdjustment table and the OperatingModeDistribution table.

The fourth substep uses the results of the second and third substeps to calculate the AC adjustment factors used in subsequent calculations.

CREC-5 Weight emission rates by source bin

This step applies the source bin distributions to the EmissionRateWithIM table produced in step 1, storing the results in a SBWeightedEmissionRate table. FuelType distinctions from the source bin classification are preserved but other source bin discriminators are aggregated out. The modelYearID distinction contained in the EmissionRateWithIM table is preserved in SBWeightedEmissionRate, effectively subsuming the role of modelYearGroupID as a source bin discriminator.

CREC-6 Weight emission rates by operating mode

This step applies the operating mode distributions to the SBWeightedEmissionRate table calculated in the previous step, to produce an intermediate FullyWeightedEmissionRate table.

CREC-7 Apply fuel, temperature, and opmode-weighted-AC adjustment factors to weighted emission rates

This step applies the fuel supply adjustment factors from step 2, the temperature adjustment factors from step 3, and the ACAdjustment factors from step 4 to the results of step 6.

CREC-8 Calculate and Apply Humidity Correction Factor to NOx running emissions

This step calculates a multiplicative correction factor based on the specific humidity value in ZoneMonthHour and a fuel type-dependent coefficient and applies it to any NOx emission results from step 7.

CREC-9 Multiply fully weighted and adjusted emission rates by source hours operating (SHO) activity

> This step multiplies the result of step 8 by the activity from the SHO table. Since SHO is stored by Link, the results of this step are specific to a Link and a Roadtype and are essentially now at the level of the MOVESOutput when reported by sourceTypeID.

CREC-10 Convert results by source type into results by SCC

> This step is performed only when the run specification requests output by SCC code.
>
> The SCCVTypeDistributions and the SCCRoadType distributions are applied to the results of step 9 in a single processing step.

10.21.2. Detailed Calculation Steps

CREC-1: Calculate emission rates which account for I/M programs

CREC 1-a: Complete I/M adjustment fraction information

Input Variables:

begModelYearID, endModelYearID, IMAdjustFract (IMCoverage table)
countyID, zoneID, yearID (from Master Loop Context).
ageID (AgeCategory table)
regClassID (RegulatoryClass table)

Output Variable:

IMAdjustFract (in intermediate IMAdjustment table)

Calculation:

```
For zoneID, yearID in Master Loop Context,
for running process (which is also the Master Loop Context process)
            of all pollutants in runSpec which this calculator calculates
for all ageID in AgeCategory table,
for all regClassID values in RegulatoryClass except regClassID=0


modelYearID = yearID-ageID
IMAdjustFract (zoneID, yearID, polProcessID, modelYearID, ageID,
    fuelTypeID, regClassID)
= IMAdjustFract if IMCoverage record exists
            with begModelYearID <= modelYearID <= endModelYearID
= 0.0 otherwise
```

CREC 1-b: Combine I/M and non I/M rates

Input Variables:

meanBaseRate and meanBaseRateIM (EmissionRateByAge table)
fuelTypeID, regClassID (from SourceBin table)
modelYearGroupID (from PollutantProcessModelYear table)
IMAdjustFract (from step 1-a)

Output Variable:

meanBaseRate (in new intermediate table EmissionRateWithIM)

Calculation:

```
For all records in join of IMAdjustment Table to EmissionRateByAge table
      WHERE

IMAdjustment.polProcessID=EmissionRateByAge.polProcessID

IMAdjustment.fuelTypeID = SourceBin.fuelTypeID AND
IMAdjustment.regClassID = SourceBin.regClassID AND
IMAdjustment.modelYearID= PollutantProcessModelYear.modelYearID AND
IMAdjustment.polProcessID = PollutantProcessModelYear.polProcessID AND
PollutantProcessModelYear.modelYearGroupID =
      SourceBin.modelYearGroupID   AND
EmissionRateByAge.sourceBinID=SourceBin.sourceBinID AND
EmissionRateByAge.ageGroupID=AgeCategory.ageGroupID AND
IMAdjustment.ageID=AgeCategory.ageID

meanBaseRate (zoneID, yearID, polProcessID, modelYearID, sourceBinID,
      opModeID) =  meanBaseRate + (IMAdjustFract * (meanBaseRateIM-
      meanBaseRate))

if meanBaseRate < 0.0 then set meanBaseRate = 0.0 (since IMAdjustFract might
      be greater than 1.0.)
```

CREC-2: Calculate fuel-supply-weighted Fuel Adjustment Factors

CREC 2-a: Combine GPA and non GPA fuel adjustment factors

Input Variables:

fuelAdjustment and fuelAdjustmentGPA (from FuelAdjustment table)
GPAFract (from County table)
countyID (from MasterLoop Context)

Output Variable:

fuelAdjustment (in intermediate CountyFuelAdjustment table)

Calculation:

```
For countyID from Master Loop Context,
For all records in FuelAdjustment
for running process (which is also the Master Loop Context process)
            of all pollutants in runSpec which this calculator calculates

fuelAdjustment (countyID, polProcessID, fuelMYGroupID, sourceTypeID,
      fuelFormulationID )
= fuelAdjustment + GPAFract*(fuelAdjustmentGPA-fuelAdjustment)
```

NOTE: An internal MOVES model component, the DefaultDataMaker, has set fuelAdjustment=1 for cases which are not populated in fuelAdjustment.

CREC 2-b: Aggregate county fuel adjustments to fuel type

Input Variables:

fuelAdjustment (from CountyFuelAdjustment table from step 2-a)
marketShare (from FuelSupply table)
fuelSubTypeID (from FuelFormulation table)
fuelTypeID (from FuelSubtype table)
monthID (from MonthOfAnyYear table)
yearID (from MasterLoopContext)
fuelYearID (from Year table)
modelYearID (from PollutantProcessModelYear table)

Output Variable:

fuelAdjustment (in intermediate FuelSupplyAdjustment table)

Calculation:

```
    For all records in join of CountyFuelAdjustment and FuelSupply where
CountyFuelAdjustment.countyID=FuelSupplycountyID and
Year.yearID = yearID from MasterLoopContext and
Year.fuelYearID = FuelSupply.fuelYearID and
FuelSupply.monthGroupID = MonthOfAnyYear.monthGroupID and
CountyFuelAdjustment.monthGroupID=FuelSupply.monthGroupID and
CountyFuelAdjustment.fuelFormulationID=FuelFormulation.fuelFormulationID and
FuelFormulation.fuelSubtypeID=FuelSubType.fuelSubtypeID  and
        PollutantProcessModelYear.polProcessID=
            CountyFuelAdjustment.polProcessID and
PollutantProcessModelYear.fuelMYGroupID=
            CountyFuelAdjustment.fuelMYGroupID

    fuelAdjustment (countyID, yearID, monthID, polProcessID, modelYearID,
                sourceTypeID, fuelTypeID )

    = sum (fuelAdjustment * marketShare) over all fuel formulations in each fuel
        type.
```

CREC-3: Calculate temperature adjustment factors

Input Variables:

tempAdjustTermA, tempAdjustTermB (from TemperatureAdjustment table)
temperature (from ZoneMonthHour table)
zoneID (from MasterLoop context)

Output Variable:

temperatureAdjustment (in intermediate METAdjustment table)

Calculation:

```
For zoneID in Master Loop Context,
For running process (which is also the Master Loop Context process)
            of all pollutants in runSpec which this calculator calculates
For all records in cross join of ZoneMonthHour and TemperatureAdjustment

temperatureAdjustment(zoneID, monthID, hourID, polProcessID, fuelTypeID)
= 1.0 + tempAdjustTermA * (temperature-75) + tempAdjustTermB
    $(temperature-75)^2$
```

CREC-4: Calculate Air Conditioning (AC) Adjustment Factors

CREC 4-a: Calculate AC On Fraction

Input Variables:

heatIndex (zoneID, monthID, hourID)
From ZoneMonthHour table
ACActivityTermA (monthGroupID, hourID)
From MonthGroupHour table
ACActivityTermB (monthGroupID, hourID)
From MonthGroupHour table
ACActivityTermC (monthGroupID, hourID)
From MonthGroupHour table

Output Variable:

ACOnFraction (in intermediate ACOnFraction table)

Calculation:

From join of ZoneMonthHour and MonthGroupHour tables where
MonthGroupHour.monthGroupID = MonthOfAnyYear.monthGroupID and
MonthOfAnyYear.monthID = ZoneMonthHour.monthID and
MonthGroupHour.hourID = ZoneMonthHour.hourID

ACOnFraction (zoneID, monthID, hourID) =
(ACActivityTermA+ACActivityTermB*heatIndex
+ACActivityTermC*heatIndex^2)

If ACOnFraction<0, set to 0
If ACOnFraction>1, set to 1

CREC 4-b: Calculate AC Activity Fraction

Input Variables:

ACOnFraction (zoneID, monthID, hourID) from previous calculation
ACPenetrationFraction (sourceTypeID, modelYearID) from
SourceTypeModelYear table
functioningACFraction (sourceTypeID, ageID) from SourceTypeAge table

Output Variable:

ACActivityFraction (in intermediate ACActivityFraction table)

Calculation:

From join of ACOnFraction, SourceTypeModelYear and SourceTypeAge tables
where
yearID = yearID value from master loop context
SourceTypeModelYear.sourceTypeID = SourceTypeAge.sourceTypeID and
SourceTypeModelYear.modelYearID = yearID - SourceTypeAge.ageID

ACActivityFraction (zoneID, monthID, hourID, sourceTypeID, modelYearID)
= ACOnFraction * ACPenetrationFraction * functioningACFraction

CREC 4-c: Weight FullACAdjustment Factors by Operating Mode

Input Variables:

FullACAdjustment (polProcessID, sourceTypeID, opModeID) from
FullACAdjustment table
opModeFraction (sourceTypeID, hourDayID, linkID, polProcessID, opModeID)
from OpModeDistribution Table
linkID from master loop context

Output Variable:

weightedFullACAdjustment (in intermediate WeightedFullACAdjustment table)

Calculation:

```
For all records in join of FullACAdjustment and OpModeDistribution
using polProcessID, sourceTypeID, and opModeID
where linkID = value from master loop context and hourDayID is in the run
    specification

weightedFullACAdjustment (sourceTypeID, polProcessID, linkID, hourDayID )
 = sum (fullACAdjustment * opModeFraction)
```

CREC 4-d: Calculate AC Adjustment Factor

Input Variables:

ACActivityFraction (zoneID, monthID, hourID, sourceTypeID, modelYearID)
From step 4-b
weightedFullACAdjustment (sourceTypeID, polProcessID, linkID, hourDayID)
From WeightedFullACAdjustment table

Output Variable:

ACAdjustment (in intermediate ACAdjustment table)

Calculation:

```
For all records in join of ACActivityFraction and WeightedFullACAdjustment
    where
    WeightedFullACAdjustment.linkID=Link.linkID and
    Link.zoneID=ACActivityFractioin.zoneID and
    WeightedFullACAdjustment.hourDayID=HourDay.hourDayID and
    HourDay.hourID=ACActivityFraction.hourID and
    WeightedFullACAdjustment.sourceTypeID=ACActivityFraction.sourceTyp
    eID

ACAdjustment (zoneID, monthID, hourID, dayID, sourceTypeID,
    modelYearID, polProcessID)
 = 1 + ((weightedFullACAdjustment –1)*ACActivityFraction)
```

CREC-5: Weight emission rates by source bin

Input Variables:

meanBaseRate (from EmissionRateWithIM table from step 1-b)
sourceBinActivityFraction (from SourceBinDistribution table)

Output Variable:

meanBaseRate (in a new intermediate SBWeightedEmissionRate table)

Calculation:

For all records in join of EmissionRateWithIM, SourceBin, SourceBinDistribution, and SourceTypeModelYear where:
SourceBinDistribution.sourceBinID = SourceBin.sourceBinID and
SourceBinDistribution.sourceTypeModelYearID = SourceTypeModelYear.sourceTypeModelYearID and
EmissionRateWithIM.polProcessID=SourceBinDistribution.polProcessID and
EmissionRateWithIM.sourceBinID=SourceBinDistribution.sourceBinID and
EmissionRateWithIM.modelYearID=SourceTypeModelYear.modelYearID

meanBaseRate(zoneID, yearID, polProcessID, sourceTypeID, modelYearID, fuelTypeID, opModeID)
= sum (sourceBinActivityFraction * meanBaseRate)

Note that sourceBinID is not included in the summation grouping; all of its attributes except fuelTypeID are included instead, also note that there should be only one modelYearGroupID present in the source bin distributions for a modelYearID and polProcessID without having to specify this.

CREC-6: Weight emission rates by operating mode

Input Variables:

meanBaseRate (from SBWeightedEmissionRate table from step 5)
opModeFraction (from OpModeDistribution table)

Output Variable:

meanBaseRate (in a intermediate FullyWeightedEmissionRate table)

Calculation:

For all records in join of SBWeightedEmissionRate and OpModeDistribution where
SBWeightedEmissionRate.polProcessid=OpModeDistribution.polProcessID and
SBWeightedEmissionRate.opModeID=OpModeDistribution.opModeID and
SBWeightedEmissionRate.sourceTypeID=OpModeDistribution.sourceTypeID
OpModeDistribution.linkID=Link.linkID and
Link.zoneID= SBWeightedEmissionRate.zoneID
meanBaseRate(linkID, yearID, polProcessID, sourceTypeID, modelYearID, fuelTypeID, hourDayID)
= sum (opModeFraction * meanBaseRate)

CREC-7: Apply fuel, temperature, and AC adjustment factors to weighted emission rates

CREC 7-a: Combine Temperature and AC Adjustment Factors

Input Variables:

temperatureAdjustment (from METAdjustment table from step 3)
ACAdjustment (from ACAdjustment table from step 4)

Output Variable:

tempAndACAdjusment (in intermediate TempAndACAdjustment table)

Calculation:

```
For all records in join of METAdjustment and ACAdjustment
    Using zoneID, polProcessID, MonthID, and hourID

tempAndACAdjustment (zoneID, polProcessID, sourceTypeID, modelYearID,
    fuelTypeID, monthID, hourID, dayID) =
    temperatureAdjustment*ACAdjustment
```

CREC 7-b: Apply fuel adjustment to fully weighted emission rates

Input Variables:

meanBaseRate (from FullyWeightedEmissionRate table from step 6)
fuelAdjustment (from FuelSupplyAdjustment table from step 2)

Output Variable:

fuelAdjustedRate (in intermediate FuelAdjustedRate table)

Calculation:

```
    For linkID in master loop context
    From join of FullyAdjustedEmissionRate and FuelSupplyAdjustment where
FullyAdjustedEmissionRate.linkID = Link.linkID and
Link.countyID=FuelSupplyAdjustment.countyID and
FullyAdjustedEmissionRate.yearID=FuelSupplyAdjustment.yearID and
FullyAdjustedEmissionRate.polProcessID
            =FuelSupplyAdjustment.polProcessID and
FullyAdjustedEmissionRate.sourceTypeID=
            FuelSupplyAdjustment.sourceTypeID and
FullyAdjustedEmissionRate.modelYearID=
            FuelSupplyAdjustment.modelYearID and
FullyAdjustedEmissionRate.fuelTypeID=FuelSupplyAdjustment.fuelTypeID

    fuelAdjustedRate (linkID, yearID, polProcessID, sourceTypeID, modelYearID,
        fuelTypeID, monthID, hourDayID) = meanBaseRate * fuelAdjustment
```

CREC 7-c: Apply temperature –and-AC adjustment to fuel-adjusted emission rate

Input Variables:

tempAndACAdjustment (from TempAndACAdjustment table from step 7-a)
fuelAdjustedRate (from FuelAdjustedRate table from step 7-b)

Output Variable:

meanBaseRate (in intermediate WeightedAndAdjustedEmissionRate table

Calculation:

From join of TempAndACAdjustment and FuelAdjustedRate tables where
FuelAdjustedRate.linkID = Link.linkID and
Link.zoneID= TempAndACAdjustment.zoneID and
TempAndACAdjustment.hourID=HourDay.hourID and
TempAndACAdjustment.dayID=HourDay.dayID and
HourDay.hourDayID= FuelAdjustedRate.hourDayID and
TempAndACAdjustment.polProcessID= FuelAdjustedRate.polProcessID and
TempAndACAdjustment.sourceTypeID= FuelAdjustedRate.sourceTypeID and
TempAndACAdjustment.modelYearID= FuelAdjustedRate.modelYearID and
TempAndACAdjustment.monthID= FuelAdjustedRate.monthID and
TempAndACAdjustment.fuelTypeID= FuelAdjustedRate.fuelTypeID

meanBaseRate (linkID, yearID, polProcessID, sourceTypeID, modelYearID, fuelTypeID, hourID, dayID, monthID) = fuelAdjustedRate * tempAndACAdjustment

CREC-8: Calculate and Apply Humidity Correction Factor to NOx Emissions

Input Variables:

meanBaseRate (from WeightedAndAdjustedEmissionRate table from step 7)
humidityCorrectionCoeff (from FuelType table)
specificHumidity (from ZoneMonthHour table)

Output Variable:

meanBaseRate (in WeightedAndAdjustedEmissionRate table)

Calculation:

Using join of ZoneMonthHour, FuelType, and WeightedAndAdjustedEmissionRate where:

WeightedAndAdjustedEmissionRate.fuelTypeID = FuelType.fuelTypeID and
WeightedAndAdjustedEmissionRate.linkID=Link.linkID and
Link.zoneID=ZoneMonthHour.zoneID and
WeightedAndAdjustedEmissionRate.monthID = ZoneMonthHour.monthID and
WeightedAndAdjustedEmissionRate.hourID=ZoneMonthHour.hourID and
pollutantProcessID = 301

boundedSpecificHumidity = GREATEST(21.0, LEAST(specificHumidity,124.0))

```
K = 1.0 – ( (boundedSpecificHumidity – 75.0) * humidityCorrectionCoeff)
meanBaseRate = meanBaseRate * K
```

CREC-9: Multiply fully weighted and adjusted emission rates by source hour operating (SHO) activity to generate inventory

Input Variables:

meanBaseRate (from WeightedAndAdjustedEmissionRate table from step 8)
SHO (from SHO table)
yearID, stateID, countyID, zoneID, linkID from MasterLoop context
pollutantID, processID from PollutantProcessAssoc table
roadTypeID from Link table

Output Variable:

emissionQuant (in MOVESWorkerOutput table)

Calculation:

```
For all records in join of SHO, WeightedAndAdjustedEmissionRate,
HourDay,  Link, and PollutantProcessAssoc tables where:
    SHO.linkID = linkID value from MasterLoopContext and
    WeightedAndAdjustedEmissionRate.linkID = SHO.linkID and
    SHO.yearID = WeightedAndAdjustedEmissionRate.yearID and
    yearID – SHO.ageID
            = WeightedAndAdjustedEmissionRate.modelYearID and
    SHO.monthID = WeightedAndAdjustedEmissionRate.monthID and
    SHO.sourceTypeID = WeightedAndAdjustedEmissionRate.sourceTypeID
    and
    SHO.hourDayID = HourDay.hourDayID and
    HourDay.hourID = WeightedAndAdjustedEmissionRate.hourID and
    HourDay.dayID = WeightedAndAdjustedEmissionRate.dayID and
    WeightedAndAdjustedEmissionRate.polProcessID
           =PollutantProcessAssoc.polProcessID

roadTypeID = Link.roadTypeID
SCC = null
emissionQuant(stateID, countyID, zoneID, linkID, roadTypeID, yearID,
    monthID, dayID, hourID, pollutantID, processID, sourceTypeID,
    modelYearID, fuelTypeID, SCC)
= meanBaseRate * SHO
```

CREC-10: Conditionally convert results by sourceTypeID to results by SCC

This step is only performed when the run specification requires output by SCC

Input Variables:

SCCVTypeFraction (from SCCVTypeDistribution table)
SCCRoadTypeFraction(from SCCRoadTypeDistribution table)
SCC from SCC table
emissionQuant from MOVESWorkerOutput table from step 9)
sourceTypeID, modelYearID from SourceTypeModelYear

Output Variable:

emissionQuant (in MOVESWorkerOutput table)

Calculation:

From join of SCCVtypeDistribution, SCC, SourceTypeModelYear, and
SCCRoadTypeDistribution where:

zoneID = zoneID from master loop context
SourceTypeModelYear.sourceTypeModelYearID =
SCCVtypeDistribution.sourceTypeModelYearID and
SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
MOVESWorkerOutput.sourceTypeID = sourceTypeModelYear.sourceTypeID and
SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
MOVESWorkerOutput.modelYearID = sourceTypeModelYear.modelYearID and
MOVESWorkerOutput.fuelTypeID = SCCVtypeDistribution.fuelTypeID and
MOVESWorkerOutput.roadTypeID = SCCRoadTypeDistribution.roadTypeID

SCC= SCC.SCC
emissionQuant = sum of emissionQuant * SCCVtypeFraction *
SCCRoadTypeFraction over sourceTypeID and roadTypeID

sourceTypeID = null
roadTypeID = null
linkID = null

10.22. Criteria Pollutant Start EmissionCalculator (CSEC)

10.22.1. General Description

This EmissionCalculator signs up with the MOVES master looping mechanism at the Year level which, for the start process, means that a single execution need deals with one Zone and one Year.

The major steps of the calculation are as follows:

Preliminary Steps:

CSEC-1 Calculate emission rates which account for I/M programs

This step has two substeps:

The first substep calculates an intermediate IMAdjustment table from the contents of the IMCoverage table for a particular zone and year. It puts the information into a usable form. The form of the IMCoverage table, in particular the use of model year ranges as non-key fields and the presence of information only where I/M programs exist, was constrained by overall table size considerations. To be used in subsequent steps the information needs to be into a more usable form, which this substep accomplishes.

The second sub-step combines information from the EmissionRateByAge table and the IMAdjustment table produced in the first substep to produce an intermediate EmissionRatesWithIM table. The function here is to weight the meanBaseRate and meanBaseRateIM fields together using the IM fractions calculated in the previous substep. (This makes the “blended” rates dependent upon yearID and countyID, but we are executing for a single Year and County)

CSEC-2 Calculate fuel-supply-weighted fuel adjustment factors

This step has two substeps:

The first substep uses the FuelAdjustment and County tables to produce an intermediate CountyFuelAdjustment table specific to the county containing the zone being executed. It resolves the GPA and non-GPA fuel adjustment factors into a single factor based on the GPAFract value in the County table. It is still at the fuelFormulation level. (This step is repeated unnecessarily for each zone and year in the county, but is a minor one.)

The second substep uses CountyFuelAdjustment and the FuelSupply table to produce an intermediate FuelSupplyAdjustment table of fuel adjustment factors at the fuelTypeID level.

CSEC-3 Calculate temperature adjustment factors

This step uses the StartTempAdjustment table along with temperature information from the ZoneMonthHour table to produce an intermediate METStartAdjustment table. This step is repeated unnecessarily for the zone for each year.

The central core model calculations

CSEC-4 Apply Start Temperature Adjustment to Emission Rates.

This step adds the start temperature adjustment factors determined CSEC-3 to the EmissionRatesWithIM table to produce an intermediate EmissionRatesWithIMAndTemp table.

CSEC-5 Weight EmissionRates by Source Bin

This step applies the source bin distributions to the EmissionRatesWithIMAndTemp table from the previous step to produce an intermediate METSourceBinEmissionRates table.   FuelType distinctions from the source bin classification are preserved but other source bin discriminators used in the start process (currently engine technology and regulatory class) are aggregated out. The modelYearID distinction is present in the EmissionRatesWithIMAndTemp table and is preserved, effectively subsuming the role of modelYearGroupID as a source bin discriminator.

CSEC-6 Weight temperature-adjusted emission rates by operating mode

This step applies the operating mode distributions to the results of step 5 to produce an intermediate ActivityWeightedEmissionRate table.   This can be done since no further information is operating mode dependent. The ActivityWeightedEmissionRate table is at the level of the MOVESWorkerOutput by SourceType.

CSEC-7 Apply fuel adjustment factors

This step applies the fuel supply adjustment factors from step 2, stored in the FuelSupplyAdjustment table, to the results of step 6.

CSEC-8 Multiply by start activity

This step multiplies the result of step 7 by the activity from the Starts table.

CSEC-9 Convert results by source type into results by SCC

This step is performed only when the run specification requests output by SCC code and has two substeps.

The first substep produces an SCCDistribution table from the SCCSourceTypeDistribution and SCCRoadTypeDistribution tables. This is simplified by the fact that start emissions are associated only with the off network roadtype, so that only 12 of the 144 SCC codes are involved.

The second substep applies the SCCDistribution table to the results of step 9 to convert the output from being by sourceTypeID to being by SCC.

Implicit in these calculations is that the additional fields of stateID, countyID, and roadTypeID are added to the product table as needed by the MOVESOutput table format. The roadTypeID for start emissions is always 1.

10.22.2. Detailed Calculation Steps

CSEC-1: Calculate emission rates which account for I/M programs

CSEC 1-a: Complete I/M adjustment fraction information

Input Variables:

begModelYearID, endModelYearID, IMAdjustFract (IMCoverage table)
countyID, zoneID, yearID (from Master Loop Context).
ageID (AgeCategory table)
regClassID (RegulatoryClass table)

Output Variable:

IMAdjustFract (in intermediate IMAdjustment table)

Calculation:

```
For zoneID, yearID in Master Loop Context,
for start process (which is also the Master Loop Context process)
            of all pollutants in runSpec which this calculator calculates
for all ageID in AgeCategory table,
for all fuelTypeID in IMCoverage,
for all regClassID values in RegulatoryClass except regClassID=0


modelYearID = yearID-ageID
IMAdjustFract (zoneID, yearID, polProcessID, modelYearID, fuelTypeID,
    regClassID) =
= IMAdjustFract if IMCoverage record exists
            with begModelYearID <= modelYearID <= endModelYearID
= 0.0 otherwise
```

CSEC 1-b: Combine I/M and non I/M rates

CSEC 1-b-1

Input Variables:

ageID (AgeCategory table)
polProcessID (from SourceBin table)

modelYearGroupID (from tables of PollutantProcessModelYear and IMAdjustment)
fuelTypeID & regClassID (from IMAdjustment in step 1-a and SourceBin table)
modelYearID (from IMAdjustment in step 1-a and PollutantProcessModelYear table)

Output Variable:

zoneID, yearID, ageGroupID, polProcessID, modelYearID,
sourceBinID, IMAdjustFract (in intermediate table IMAdjustmentWithSourceBin)

Calculation:

```
INSERT INTO IMAdjustmentWithSourceBin
SELECT zoneID, yearID, ageGroupID, ima.polProcessID, ima.modelYearID,
sourceBinID, IMAdjustFract
FROM IMAdjustment ima
INNER JOIN AgeCategory ac ON (ac.ageID = yearID-ima.modelYearID)
INNER JOIN PollutantProcessModelYear ppmy ON (
ppmy.polProcessID=ima.polProcessID AND
ppmy.modelYearID=ima.modelYearID)
INNER JOIN SourceBin sb ON (
sb.fuelTypeID = ima.fuelTypeID AND
sb.regClassID = ima.regClassID AND
sb.modelYearGroupID = ppmy.modelYearGroupID);
```

CSEC 1-b-2

Input Variables:

meanBaseRate and meanBaseRateIM (EmissionRate ByAge table)
sourceBinID, polProcessID and ageGroupID (from tables of EmissionRateByAge and step 1-b-1 IMAdjustmentWithSourceBin)
modelYearGroupID (from PollutantProcessModelYear table)
IMAdjustFract (from step 1-b-1 IMAdjustmentWithSourceBin table)

Output Variable:

meanBaseRate (in intermediate table EmissionRateWithIM)

Calculation:

For all records in join of IMAdjustmentWithSourceBin Table to EmissionRateByAge table WHERE

IMAdjustmentWithSourceBin.polProcessID=EmissionRateByAge.polProcessID

IMAdjustmentWithSourceBin.ageGroupID = EmissionRateByAge.AgeGroupID AND
IMAdjustmentWithSourceBin.polProcessID = EmissionRateByAge.polProcessID AND
IMAdjustmentWithSourceBin.sourceBinID= EmissionRateByAge.sourceBinID

meanBaseRate (zoneID, yearID, polProcessID, modelYearID,

sourceBinID, opModeID) = GREATEST(meanBaseRateIM*IMAdjustFract +
    meanBaseRate*(1.0-IMAdjustFract),0.0)

Note that IMAdjustFract might be greater than 1.0.

CSEC-2: Calculate fuel-supply-weighted Fuel Adjustment Factors

CSEC 2-a: Combine GPA and non GPA fuel adjustment factors

Input Variables:

fuelAdjustment and fuelAdjustmentGPA (from FuelAdjustment table)
GPAFract (from County table)
countyID (from MasterLoop Context)

Output Variable:

fuelAdjustment (in intermediate CountyFuelAdjustment table)

Calculation:

```
For countyID from Master Loop Context,
For all records in FuelAdjustment
for start process (which is also the Master Loop Context process)
            of all pollutants in runSpec which this calculator calculates


fuelAdjustment (countyID, polProcessID, fuelMYGroupID, sourceTypeID,
    fuelFormulationID )
= fuelAdjustment + GPAFract*(fuelAdjustmentGPA-fuelAdjustment)
```

NOTE: An internal MOVES model component, the DefaultDataMaker, has set fuelAdjustment=1 for cases which are not populated in fuelAdjustment.

CSEC 2-b: Aggregate county fuel adjustments to fuel type

Input Variables:

fuelAdjustment (from CountyFuelAdjustment table from step 2-a)
marketShare (from FuelSupply table)
fuelSubTypeID (from FuelFormulation table)
fuelTypeID (from FuelSubtype table)
monthID (from MonthOfAnyYear table)
yearID (from MasterLoopContext)
fuelYearID (from Year table)
modelYearID (from PollutantProcessModelYear table)

Output Variable:

fuelAdjustment (in intermediate FuelSupplyAdjustment table)

Calculation:

```
    For all records in join of CountyFuelAdjustment and FuelSupply where
CountyFuelAdjustment.countyID=FuelSupply.countyID and
Year.yearID = yearID from MasterLoopContext and
Year.fuelYearID = FuelSupply.yearID and
FuelSupply.monthGroupID = MonthOfAnyYear.monthGroupID and
CountyFuelAdjustment.monthGroupID=FuelSupply.monthGroupID and
CountyFuelAdjustment.fuelFormulationID=FuelFormulation.fuelFormulationID and
FuelFormulation.fuelSubtypeID=FuelSubType.fuelSubtypeID  and
        PollutantProcessModelYear.polProcessID=
            CountyFuelAdjustment.polProcessID and
PollutantProcessModelYear.fuelMYGroupID=
            CountyFuelAdjustment.fuelMYGroupID

    fuelAdjustment (countyID, yearID, monthID, polProcessID, modelYearID,
                sourceTypeID, fuelTypeID )

    = sum (fuelAdjustment * marketShare) over all fuel formulations in each fuel
        type.
```

CSEC-3: Calculate temperature adjustment factors

Input Variables:

tempAdjustTermA, tempAdjustTermB (from StartTempAdjustment table)
temperature (from ZoneMonthHour table)
zoneID (from MasterLoop context)
modelYearID (from PollutantProcessModelYear table)

Output Variable:

temperatureAdjustment (in intermediate METStartAdjustment table)

Calculation:

```
    For zoneID in Master Loop Context,
    For start process (which is also the Master Loop Context process)
                of all pollutants in runSpec which this calculator calculates
    For all records in join of ZoneMonthHour , StartTemperatureAdjustment and
    PollutantProcessModelYear where:
PollutantProcessModelYear.polProcessID =
    StartTempAdjustment.polProcessID and
PollutantProcessModelYear.modelYearGroupID=
            StartTempAdjustment.modelYearGroupID

    temperatureAdjustment(zoneID, monthID, hourID, polProcessID, modelYearID,
        fuelTypeID, opModeID)
    = tempAdjustTermA * (temperature-75) + tempAdjustTermB (temperature-75)^2
    + tempAdjustTermC * (temperature-75)^3
```

CSEC-4: Apply temperature adjustment factors

Input Variables:

meanBaseRate (from EmissionRateWithIM table from step 1-b)
temperatureAdjustment (from METStartAdjustment table from step 3)

Output Variable:

meanBaseRate (in a new intermediate EmissionRateWithIMAndTemp table)

Calculation:

```
    For all records in join of METStartAdjustment and EmissionRateWithIM,
    (Using SourceBin table to relate METSTartAdjustment.fuelTypeID to
        EmissionRateWithIM.sourceBinID)

    EmissionRateWithIMAndTemp.meanBaseRate (zoneID, monthID, hourID,
        yearID, polProcessID, modelYearID, sourceBinID, opModeID) =
        EmissonRateWithIM.meanBaseRate + temperatureAdjustment
```

CSEC-5: Weight emission rates by source bin

Input Variables:

meanBaseRate (from EmissionRateWithIMAndTemp table from step 4)
sourceBinActivityFraction (from SourceBinDistribution table)

Output Variable:

meanBaseRate (in an intermediate METSourceBinEmissionRate table)

Calculation:

```
For all records in join of EmissionRateWithIMAndTemp, SourceBin,
      SourceBinDistribution, and
SourceTypeModelYear where:
SourceBinDistribution.sourceBinID = SourceBin.sourceBinID and
      SourceBinDistribution.sourceTypeModelYearID
            = SourceTypeModelYear.sourceTypeModelYearID
      EmissionRateWithIMAndTemp.polProcessID=
      SourceBinDistribution.polProcessID and
      EmissionRateWithIMAndTemp.sourceBinID=
      SourceBinDistribution.sourceBinID  and
SourceTypeModelYear.modelYearID =
      EmissionRateWithIMAndTemp.modelYearID
```

Note: This calculation can take advantage of the fact that there is only one modelYearGroupID present for a modelYearID.

```
meanBaseRate(zoneID, monthID, hourID, yearID, polProcessID, sourceTypeID,
      modelYearID, fuelTypeID, opModeID)
= sum over engTechID and regClassID (sourceBinActivityFraction *
      meanBaseRate)
```

CSEC-6: Weight temperature-adjusted emission rates by operating mode

Input Variables:

meanBaseRate (from METSourceBinEmissionRate table from step 5)
opModeFraction (from OperatingModeDistribution table)
linkID (from MasterLoopContext)
(Note: calculations only performed for off-network links)

Output Variable:

meanBaseRate (in an intermediate ActivityWeightedEmissionRate table)

Calculation:

```
For linkID from MasterLoop context.
For all records in join of METSourceBinEmissionRate, HourDay, and
      OperatingModeDistribution where:
OpModeDistribution.hourDayID = HourDay.hourDayID and
HourDay.hourID=METSourceBinEmissionRate.hourID and
OpModeDistribution.polProcessID =
      METSourceBinEmissionRate.polProcessID and
OpModeDistribution.sourceTypeID =
      METSourceBinEmissionRate.sourceTypeID and
```

OpModeDistribution.opModeID = METSourceBinEmissionRate.opModeID

meanBaseRate(zoneID, yearID, monthID, dayID, hourID, polProcessID,
    sourceTypeID, modelYearID, fuelTypeID)
= sum over opModeID (meanBaseRate * opModeFraction)

CSEC-7: Apply fuel adjustment factor

Input Variables:

meanBaseRate (from ActivityWeightedEmissionRate table from step 6)
fuelAdjustment (from FuelSupplyAdjustment table from step 2)

Output Variable:

meanBaseRate (in intermediate ActivityWeightedEmissionRate table)

Calculation:

For all records in join of ActivityWeightedEmissionRate and
    FuelSupplyAdjustment using: yearID, monthID, polProcessID,
    sourceTypeID, modelYearID, and fuelTypeID

meanBaseRate(zoneID, yearID, monthID, dayID, hourID, polProcessID,
    sourceTypeID, modelYearID, fuelTypeID)
= meanBaseRate * fuelAdjustment

CSEC-8: Multiply emission rates by start activity to generate inventory

Input Variables:

meanBaseRate (from ActivityWeightedEmissionRate table from step 7)
starts (from Starts table)
yearID, stateID, countyID, zoneID, linkID from MasterLoop context
pollutantID, processID from PollutantProcessAssoc table

Output Variable:

emissionQuant (in MOVESWorkerOutput table)

Calculation:

```
For all records in join of Starts, ActivityWeightedEmissionRate, HourDay, and
    PollutantProcessAssoc tables where:
    Starts.zoneID = zoneID value from MasterLoopContext
    Starts.yearID = yearID value from MasterLoopContext
    yearID – Starts.ageID
        = ActivityWeightedEmissionRate.modelYearID and
    Starts.monthID = ActivityWeightedEmissionRate.monthID and
    Starts.sourceTypeID = ActivityWeightedEmissionRate.sourceTypeID and
    Starts.hourDayID = HourDay.hourDayID and
    HourDay.hourID = ActivityWeightedEmissionRate.hourID and
    HourDay.dayID = ActivityWeightedemissionRate.dayID and
    ActivityWeightedEmissionRate.polProcessID
        =PollutantProcessAssoc.polProcessID

roadTypeID = 1
SCC = null
```

emissionQuant(stateID, countyID, zoneID, linkID, roadTypeID, yearID,
    monthID, dayID, hourID, pollutantID, processID, sourceTypeID,
    modelYearID, fuelTypeID, SCC)
= meanBaseRate * starts

CSEC-9: Conditionally convert results by sourceTypeID to results by SCC

This step is only performed when the run specification requires output by SCC

CSEC 9-a: Calculate combined SCC Distribution (by both SCCVtype and SCCRoadtype)

Input Variables:

SCCVtypeFraction (from SCCVtypeDistribution table)
SCC from SCC table

Output Variable:

SCCFraction (in intermediate SCCDistribution table)

Calculation:

From join of SCCVtypeDistribution, SCC, and SourceTypeModelYear where:
SourceTypeModelYear.sourceTypeModelYearID =
    SCCVtypeDistribution.sourceTypeModelYearID and
SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
SCC.SCCroadTypeID=1

SCCFraction(zoneID, sourceTypeID, modelYearID, fuelTypeID, SCC) =
    SCCVtypeFraction

CSEC 9-b: Apply SCCDistribution

Input Variables:

emissionQuant (from MOVESWorkerOutput table from step 8)
SCCFraction (from SCCDistribution from step 9-a)

Output Variable:

emissionQuant (in MOVESWorkerOutput table)

Calculation:

sourceTypeID=null
from join of MOVESWorkerOutput and SCCDistribution using:
    sourceTypeID, modelYearID, fuelTypeID
emissionQuant(stateID, countyID, zoneID, linkID, roadTypeID, yearID,
    monthID, dayID, hourID, pollutantID, processID, sourceTypeID,
    modelYearID, fuelTypeID, SCC)
= sum over sourceTypeID (emissionQuant*SCCFraction)

10.23. Basic Running PM EmissionCalculator

10.23.1. General Description

This calculator computes the emissions of OCarbon and ECarbon PM2.5 from the running exhaust process (polProcessIDs 11101 and 11201). It signs up with the MOVES master looping mechanism at the YEAR level. It retrieves its emission rates from EmissionRateByAge, and only considers the meanBaseRate, ignoring meanBaseRateIM. The only calculation it performs, beyond the core model considerations of retrieving total activity from SHO, applying source bin distributions and operating mode distributions, is to apply a multiplicative temperature adjustment factor retrieved from the TemperatureAdjustment table.

10.23.2. Detailed Calculation Steps

Step BRPMC-1: Weight Emission Rates by Operating Mode

Structurally this combines the OpModeDistribution and EmissionRateByAge tables. Relative to EmissionRateByAge this step removes opModeID but adds hourDayID and sourceTypeID. If the calculator executed above the LINK level, linkID would also have to be added.

Input Variables:

polProcessID, linkID from Master Loop Context
EmissionRateByAge table from MOVESExecution database
OpModeDistribution table from MOVESExecution database

Output Variables:

Intermediate OpModeWeightedEmissionRate table

Key fields: hourDayID, sourceTypeID, sourceBinID, ageGroupID
Data field: opModeWeightedMeanBaseRate

Calculation:

For polProcessID, linkID from Master Loop Context
For all hourDayID, sourceTypeID in Run Specification

opModeWeightedMeanBaseRate = SUM(opModeFraction * meanBaseRate)

Step BRPMC-2: Weight Emission Rates by Source Bin

This combines the results of the previous step with the SourceBinDistribution Table. In terms of table structure relative to the results of the previous step, this removes the engTechID and regClassID components of sourceBinID and fully expands ageGroupID and the modelYearGroupID component of sourceBinID into individual modelYearIDs. Because SourceBinDistributions are by individual model year, and the results of the previous step are by ageGroupID, yearID is added.

Input Variables:

OpModeWeightedEmissionRate table from previous step
SourceBinDistribution table from MOVESExecution database
SourceTypeModelYear table from MOVESExecution database
PollutantProcessModelYear table from MOVESExecution database
SourceBin table from MOVESExecution database
AgeCategory table from MOVESExecution database
yearID value from the master loop context

Output Variables:

Intermediate FullyWeightedEmissionRate table

Key fields: yearID, hourDayID, sourceTypeID, fuelTypeID, modelYearID
Data field: fullyWeightedMeanBaseRate

Calculation:

For yearID in the master loop context

modelYearID = yearID – ageID

fullyWeightedMeanBaseRate =
SUM(sourceBinActivityFraction * opModeWeightedMeanBaseRate)

Step BRPMC-3: Multiply Emission Rates by Activity

This combines the results of the previous step with the SHO table. In terms of table structure relative to the results of the previous step, this adds monthID.

Input Variables:

FullyWeightedEmissionRate table from previous step
SHO table from MOVESExecution database
monthID values from the run specification

Output Variables:

Intermediate UnadjustedEmissionResults table

Key fields: yearID, monthID, hourDayID, sourceTypeID, fuelTypeID, modelYearID

Data field: unadjustedEmissionQuant

Calculation:

```
modelYearID = calendar year - ageID
For all monthID values in the run specification
    unadjustedEmissionQuant = fullyWeightedMeanBaseRate * SHO
```

Step BRPMC-4: Apply Fuel Adjustment by weighing emission rates by fuel adjustment

Step BRPMC-4-a: Combine GPA and non GPA fuel adjustment factors

Input Variables:

countyID from the MasterLoopContext
FuelAdjustment table from MOVESExecution database

Output Variables:

Intermediate CountyFuelAdjustment table

Key fields: countyID, polProcessID, fuelMYGroupID, sourceTypeID, fuelFormulationID

Data field: fuelAdjustment

Calculation:

For the countyID in the Master Loop Context

New fuelAdjustment = fuelAdjustment+GPAFract*(fuelAdjustmentGPA-fuelAdjustment)

Step BRPMC-4-b: Aggregate county fuel adjustments to fuel type

Input Variables:

countyID and yearID from the MasterLoopContext
fuelFormulationID & marketShare from FuelSupply table from from MOVESExecution database
fuelTypeID from FuelSubType table from MOVESExecution database

monthID from MonthOfAnyYear table from MOVESExecution database
CountyFuelAdjustment table in previous step
modelYearID from PollutantProcessModelYear table from from MOVESExecution database

Output Variables:

Intermediate FuelSupplyWithFuelType table

Key fields: countyID, yearID, monthID, fuelFormulationID, fuelTypeID
Data field: marketShare

Execution Code :

```
INSERT INTO FuelSupplyWithFuelType
SELECT countyID, yearID, may.monthID, fs.fuelFormulationID, fst.fuelTypeID,
fs.marketShare
FROM FuelSupply fs
INNER JOIN FuelFormulation ff ON ff.fuelFormulationID = fs.fuelFormulationID
INNER JOIN FuelSubType fst ON fst.fuelSubTypeID = ff.fuelSubTypeID
INNER JOIN MonthOfAnyYear may ON fs.monthGroupID = may.monthGroupID
INNER JOIN Year y ON y.fuelYearID = fs.fuelYearID
WHERE y.yearID = ##context.year##;
```

Intermediate FuelSupplyAdjustment table

Key fields: countyID, yearID, monthID, polProcessID, modelYearID, sourceTypeID, fuelTypeID

Data field: fuelAdjustment

Calculation:

For the countyID (in the Master Loop Context), yearID (in the Master Loop Context), monthID, polProcessID, modelYearID, sourceTypeID, fuelTypeID

New fuelAdjustment = SUM(fuelAdjustment*marketShare)

Step BRPMC-4-c: Apply fuel adjustment to weighted emission rates

Input Variables:

UnadjustedEmissionResults table from previous step
FuelSupplyAdjustment table from from previous step

Output Variables:

Intermediate FuelAdjustedEmissionRate table

Key fields: yearID, monthID, hourDayID, sourceTypeID, fuelTypeID, modelYearID, polProcessID

Data field: unadjustedEmissionQuant

Calculation:

For the yearID (in the Master Loop Context), monthID, hourDayID, sourceTypeID, fuelTypeID, modelYearID, polProcessID

New unadjustedEmissionQuant = unadjustedEmissionQuant * fuelAdjustment

Step BRPMC-5: Apply Temperature Adjustment

This applies the temperature adjustment to the results of the previous step. The table resulting from this step could have the same structure as that produced by the previous step, but it seems desirable to decompose hourDayID into hourID and dayID since this is needed to join to the ZoneMonthHour table and is the form eventually needed for MOVEWorkerOutput.

Input Variables:

zoneID and polProcessID from the MasterLoopContext
UnadjustedEmissionResults table from previous step
ZoneMonthHour table from MOVESExecution database
TemperatureAdjustment table from MOVESExecution database
HourDay table from MOVESExecution database

Output Variables:

Intermediate AdjustedEmissionResults table

Key fields: yearID, monthID, dayID, hourID, sourceTypeID, fuelTypeID, modelYearID

Data field: emissionQuant

Calculation:

For the polProcessID and zoneID in the Master Loop Context

emissionQuant = unadjustedEmissionQuant * exp(tempAdjustTermA * (72-temperature))

Step BRPMC-6: Convert Results to Structure of MOVESWorkerOutput by sourceTypeID

Structurally we need to add stateID, countyID, zoneID, linkID, roadTypeID, SCC (as null value) and decompose polProcessID into pollutantID and processID.

Input Variables:

stateID, linkID from MasterLoop Context
Link table from MOVESExecution
PollutantProcessAssoc table from MOVESExecution
AdjustedEmissionResults table from previous step

Output Variables:

MOVESWorkerOutput Table

Key fields: yearID, monthID, dayID, hourID, stateID, countyID, zoneID, linkID, pollutantID, processID, ,sourceTypeID, fuelTypeID, modelYearID, roadTypeID, SCC

Data field: emissionQuant

Calculation:

MOVESWorkerOutput.emissionQuant =AdjustedEmissionResults.emissionQuant
countyID = Link.countyID
zoneID = Link.zoneID
roadTypeID = Link.roadTypeID
pollutantID = PollutantProcessAssoc.pollutantID
processID = PollutantProcessAssoc.processID
SCC = null

Step BRPMC-7: Conditionally Convert Results to Structure of MOVESWorkerOutput by SCC

This step is only performed when the run specification requires output by SCC. It is performed in several other emission calculators, e.g. step 10 of the CREC, and is repeated here:

Input Variables:

SCCVTypeFraction (from SCCVTypeDistribution table)
SCCRoadTypeFraction(from SCCRoadTypeDistribution table)
SCC from SCC table
emissionQuant from MOVESWorkerOutput table from previous step
sourceTypeID, modelYearID from SourceTypeModelYear

Output Variable:

emissionQuant (in restructured MOVESWorkerOutput table)

Calculation:

```
    From join of SCCVtypeDistribution, SCC, SourceTypeModelYear, and
        SCCRoadTypeDistribution where:
        zoneID = zoneID from master loop context
SourceTypeModelYear.sourceTypeModelYearID =
    SCCVtypeDistribution.sourceTypeModelYearID and
SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
MOVESWorkerOutput.sourceTypeID =
    sourceTypeModelYear.sourceTypeID and
MOVESWorkerOutput.modelYearID = sourceTypeModelYear.modelYearID
    and
MOVESWorkerOutput.fuelTypeID = SCCVtypeDistribution.fuelTypeID and
MOVESWorkerOutput.roadTypeID =
    SCCRoadTypeDistribution.roadTypeID


    SCC= SCC.SCC
    emissionQuant = sum of emissionQuant * SCCVtypeFraction *
        SCCRoadTypeFraction over sourceTypeID and roadTypeID

    sourceTypeID = null
    roadTypeID = null
    linkID = null
```

10.24. Sulfate PM EmissionCalculator (SEC)

10.24.1. General Description

This calculator will gives the model the capability to compute sulfate PM 2.5 emissions. The basic design of the calculator is to compute sulfate emissions as the product of a sulfate emission factor and total energy consumption of the corresponding MOVES emission process. The MOVES SulfatePMEmissionsCalculator is 'chained to' the Total Energy Consumption Calculator.

The pollutant-processes for which emission calculations are performed by this calculator are:

SulfatePM2.5 – running exhaust polProcessID = 11501
SulfatePM2.5 – start exhaust polProcessID = 11502
SulfatePM2.5 – extended idle exhaust polProcessID = 11590

Inputs to the calculator are:

- MOVESWorkerOutput.emissionQuant for total energy consumption of the corresponding emissions process (running exhaust, start exhaust, or extended idle exhaust).
- FuelFormulation.sulfurLevel. (Market-weighted average per FuelSupply.marketShare)
- FuelType.energyContent
- SulfateEmissionRate.meanBaseRate.

The meanBaseRate sulfate emission factors for gasoline vehicles are unitless and scaled to yield grams of SO4 particulate when multiplied by the product of the grams of gasoline fuel consumed * ppm sulfur of the fuel.

Because the calculations are based on the emission results for total energy consumption this calculator is 'chained' to the TotalEnergyConsumptionCalculator which executes at the YEAR level. Like all emission calculators its output is MOVESWorkerOutput.emissionQuant.

10.24.2 Detailed Calculation Steps

The calculation will be performed in two basic steps:

Step SEC-1

The first step calculates the marketshare-weighted-average sulfur level of the fuel supply at the place (county) and times (year-months) for which the master loop is executing.

Step SEC-2

The second step completes the calculation, the basic equation for which is simply:

emissionQuant(SulfatePM) = SulfateEmissionRate.meanBaseRate * averageSulfurLevel (from first step) * emissionQuant(EnergyConsumption) / FuelType.energyContent

A sample of this calculation, showing the engineering units involved, is:

(1.4807e-08 gSO4 / g fuel – ppm S) * (30 ppm gasoline S) * 1000 KJ/hr   / 10 KJ / gram fuel = 4.442e-05 g/SO4 / hr

10.24.3 PM 10 Emission Calculator

10.24.3.1 General Description

This calculator will gives the model the capability to compute PM 10 sulfate, elemental carbon, and organic carbon emissions. The basic design of the calculator is to compute PM 10 emissions as the product of a PM 2.5/sourcetype/fueltype-specific emission factor and the amount of PM 2.5 of the corresponding MOVES emission process.   The MOVES PM10EmissionCalculator is ‘chained to’ any PM 2.5 calculator.

The pollutant-processes for which emission calculations are performed by this calculator are:

SulfatePM10 – running exhaust polProcessID = 10501
SulfatePM10 – start exhaust polProcessID = 10502
SulfatePM10 – extended idle exhaust polProcessID = 10590

Organic Carbon PM10 – running exhaust polProcessID = 10101

Organic Carbon PM10 – start exhaust polProcessID = 10102

Organic Carbon PM10 – extended idle exhaust polProcessID = 10190

Organic Carbon PM10 – crankcase running exhaust polProcessID = 10115

Organic Carbon PM10 – crankcase start exhaust polProcessID = 10116

Organic Carbon PM10 – crankcase extended idle exhaust polProcessID = 10117

Elemental Carbon PM10 – running exhaust polProcessID = 10201

Elemental Carbon PM10 – start exhaust polProcessID = 10202

Elemental Carbon PM10 – extended idle exhaust polProcessID = 10290

Elemental Carbon PM10 – crankcase running exhaust polProcessID = 10215

Elemental Carbon PM10 – crankcase start exhaust polProcessID = 10216

Elemental Carbon PM10 – crankcase extended idle exhaust polProcessID = 10217

Inputs to the calculator are:

- MOVESWorkerOutput.emissionQuant for PM 2.5 of the corresponding emissions process (running exhaust, start exhaust, extended idle exhaust, or other).
- PM10EmissionRatio – a unitless factor for the amount of PM 10 created per unit of PM 2.5. This factor varies by the output pollutant, the source type, and the fuel type.

Like all emission calculators its output is MOVESWorkerOutput.emissionQuant.

10.24.3.2 Detailed Calculation Step

The calculation will be performed in one basic step:

emissionQuant(PM 10) = emissionQuant(PM 2.5) * PM10EmissionRatio(PM 10 pollutant and process, source type, fuel type)

Note that the emissionQuant values vary by all dimensions of the MOVESWorkerOutput table, including process, source type, and fuel type.

10.25. Basic Start PM EmissionCalculator

10.25.1. General Description

This calculator computes the emissions of OCarbon and ECarbon PM2.5 from the start exhaust process (polProcessIDs 11102 and 11202). It signs up with the MOVES master looping mechanism at the YEAR level. It retrieves its emission rates from EmissionRateByAge table, and only considers the meanBaseRate, ignoring meanBaseRateIM. The only calculation it performs, beyond the core model considerations of retrieving total activity from Starts, applying source bin distributions and operating mode distributions, is to apply a multiplicative temperature adjustment factor retrieved from the StartTempAdjustment table. This allows the temperature adjustment to vary by operating mode and model year group, which is needed in order to apply the effects of the Mobile Source Air Toxics (MSAT) rule effects on engine start emissions.

10.25.2. Detailed Calculation Steps

This calculator is very similar to the Basic Running PM Emission Calculator. However, the operating mode is carried forward until the temperature adjustments (which depend on operating mode) can be applied.

Step BSPMC-1: Weight Emission Rates by Operating Mode

Structurally this combines the OpModeDistribution and EmissionRateByAge tables. Relative to EmissionRateByAge adds hourDayID and sourceTypeID. If the calculator executed above the LINK level, linkID would also have to be added. The meanBaseRate is adjusted by the opModeFraction, but the operating modes are not yet summed, since the temperature adjustment (in BSPMC-4) has not yet been applied.

Input Variables:

polProcessID, linkID from Master Loop Context
EmissionRateByAge table from MOVESExecution database
OpModeDistribution table from MOVESExecution database

Output Variables:

Intermediate OpModeWeightedEmissionRate table

Key fields: hourDayID, sourceTypeID, sourceBinID, ageGroupID, opModeID
Data field: opModeWeightedMeanBaseRate

Calculation:

For polProcessID, linkID from Master Loop Context
For all hourDayID, sourceTypeID in Run Specification

opModeWeightedMeanBaseRate = (opModeFraction * meanBaseRate)

Step BSPMC-2: Weight Emission Rates by Source Bin

This combines the results of the previous step with the SourceBinDistribution Table. In terms of table structure relative to the results of the previous step, this removes the engTechID and regClassID components of sourceBinID and fully expands ageGroupID and the modelYearGroupID component of sourceBinID into individual modelYearIDs. Because SourceBinDistributions are by individual model year, and the results of the previous step are by ageGroupID, yearID is added.

Input Variables:

OpModeWeightedEmissionRate table from previous step
SourceBinDistribution table from MOVESExecution database
SourceTypeModelYear table from MOVESExecution database
PollutantProcessModelYear table from MOVESExecution database
SourceBin table from MOVESExecution database
AgeCategory table from MOVESExecution database
yearID value from the master loop context

Output Variables:

Intermediate FullyWeightedEmissionRate table

Key fields: yearID, hourDayID, sourceTypeID, fuelTypeID, modelYearID, opModeID

Data field: fullyWeightedMeanBaseRate

Calculation:

For yearID in the master loop context

modelYearID = yearID – ageID

fullyWeightedMeanBaseRate =
SUM(sourceBinActivityFraction * opModeWeightedMeanBaseRate)

Step BSPMC-3: Multiply Emission Rates by Activity

This combines the results of the previous step with the SHO table. In terms of table structure relative to the results of the previous step, this adds monthID.

Input Variables:

FullyWeightedEmissionRate table from previous step
SHO table from MOVESExecution database
monthID values from the run specification

Output Variables:

Intermediate UnadjustedEmissionResults table

Key fields: yearID, monthID, hourDayID, sourceTypeID, fuelTypeID, modelYearID, opModeID

Data field: unadjustedEmissionQuant

Calculation:

```
modelYearID = calendar year - ageID
For all monthID values in the run specification
    unadjustedEmissionQuant = fullyWeightedMeanBaseRate * SHO
```

Step BSPMC-4: Apply Temperature Adjustment

This applies the temperature adjustment to the results of the previous step. The table resulting from this step could have the same structure as that produced by the previous step, but it seems desirable to decompose hourDayID into hourID and dayID since this is needed to join to the ZoneMonthHour table and is the form eventually needed for MOVEWorkerOutput. The unadjustedEmissionQuant values are summed across operating modes, since this key is no longer needed after the temperature adjustments have been applied.

Input Variables:

zoneID and polProcessID from the MasterLoopContext
UnadjustedEmissionResults table from previous step
ZoneMonthHour table from MOVESExecution database
StartTempAdjustment table from MOVESExecution database
HourDay table from MOVESExecution database

Output Variables:

Intermediate AdjustedEmissionResults table

Key fields: yearID, monthID, dayID, hourID, sourceTypeID, fuelTypeID, modelYearID

Data field: emissionQuant

Calculation:

For the polProcessID and zoneID in the Master Loop Context

emissionQuant = sum(unadjustedEmissionQuant * tempAdustTermB*exp(tempAdjustTermA * (72-least(temperature,72))) + tempAdustTermC

Step BSPMC-5: Convert Results to Structure of MOVESWorkerOutput by sourceTypeID

Structurally we need to add stateID, countyID, zoneID, linkID, roadTypeID, SCC (as null value) and decompose polProcessID into pollutantID and processID.

Input Variables:

stateID, linkID from MasterLoop Context
Link table from MOVESExecution
PollutantProcessAssoc table from MOVESExecution
AdjustedEmissionResults table from previous step

Output Variables:

MOVESWorkerOutput Table

Key fields: yearID, monthID, dayID, hourID, stateID, countyID, zoneID, linkID, pollutantID, processID, ,sourceTypeID, fuelTypeID, modelYearID, roadTypeID, SCC

Data field: emissionQuant

Calculation:

MOVESWorkerOutput.emissionQuant =AdjustedEmissionResults.emissionQuant
countyID = Link.countyID
zoneID = Link.zoneID
roadTypeID = Link.roadTypeID
pollutantID = PollutantProcessAssoc.pollutantID
processID = PollutantProcessAssoc.processID
SCC = null

Step BSPMC-6: Conditionally Convert Results to Structure of MOVESWorkerOutput by SCC

This step is only performed when the run specification requires output by SCC. It is performed in several other emission calculators, e.g. step 10 of the CREC, and is repeated here:

Input Variables:

SCCVTypeFraction (from SCCVTypeDistribution table)
SCCRoadTypeFraction(from SCCRoadTypeDistribution table)
SCC from SCC table
emissionQuant from MOVESWorkerOutput table from previous step
sourceTypeID, modelYearID from SourceTypeModelYear

Output Variable:

emissionQuant (in restructured MOVESWorkerOutput table)

Calculation:

```
    From join of SCCVtypeDistribution, SCC, SourceTypeModelYear, and
        SCCRoadTypeDistribution where:
        zoneID = zoneID from master loop context
SourceTypeModelYear.sourceTypeModelYearID =
    SCCVtypeDistribution.sourceTypeModelYearID and
SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
MOVESWorkerOutput.sourceTypeID =
    sourceTypeModelYear.sourceTypeID and
MOVESWorkerOutput.modelYearID = sourceTypeModelYear.modelYearID
    and
MOVESWorkerOutput.fuelTypeID = SCCVtypeDistribution.fuelTypeID and
MOVESWorkerOutput.roadTypeID =
    SCCRoadTypeDistribution.roadTypeID


    SCC= SCC.SCC
    emissionQuant = sum of emissionQuant * SCCVtypeFraction *
        SCCRoadTypeFraction over sourceTypeID and roadTypeID

    sourceTypeID = null
    roadTypeID = null
    linkID = null
```

10.26. Basic Brake and Tire Wear Emission Calculators

10.26.1. General Description

These two almost identical calculators compute the tirewear and brakewear process emissions of OCarbon and ECarbon PM2.5 (polProcessIDs 11609 and 11710).   They sign up with the MOVES master looping mechanism at theYEAR level. They retrieve their emission rates from EmissionRate, and only consider the meanBaseRate, ignoring meanBaseRateIM. The only operations they perform are the core model calculations of retrieving total activity from SHO, applying source bin distributions and (possibly) operating mode distributions. The brake wear calculation does apply an operating mode distribution; the tire wear calculation does not.

10.26.2. Detailed Calculation Steps

Because this calculator is very much like Basic Running PM Emission Calculator, and is actually a bit simpler because its emission rates do not vary by age and it does not apply at temperature adjustment the details of that calculation are not repeated here.

10.27. CriteriaAndPMExtendedIdleEmissionCalculator

10.27.1. General Description

This component calculates the the extended idle process emissions of 5 criteria and PM pollutants, specifically those of:

Total Hydrocarbons (polProcessID = 190)
Carbon Monoxide (polProcessID = 290)
Oxides of Nitrogen (polProcessID = 390)
OCarbon PM Size 2.5 (polProcessID = 11190)
ECarbon PM Size 2.5 (polProcessID = 11290)

It signs up with the MOVES master looping mechanism at theYEAR level.

Extended idle emissions are associated with the single link in each zone representing off highway network locations.

This calculator retrieves its emission rates from EmissionRate, and only considers the meanBaseRate field, ignoring meanBaseRateIM. It applies a multiplicative temperature adjustment factor retrieved from the TemperatureAdjustment table, an ACAdjustment factor as done in several other EmissionCalculators, and a humidity correction factor to NOx emissions as done by the CriteriaRunningEmissionCalculator.

It does not apply an operating mode distribution.

10.27.2. Detailed Calculation Steps

Step CEIC-1: Calculate Temperature and NOx Humidity Adjustments

Input Variables:

tempAdjustTermA and tempAdjustTermB
    from the TemperatureAdjustment Table
temperature and specificHumidity from the ZoneMonthHour Table
humidityCorrectionCoeff from the FuelType Table

Output Variables:

An intermediate METAdjustment table:

Key fields: zoneID, monthID, hourID, polProcessID, fuelTypeID
Data fields: temperatureAdjustment, K (the NOx correction factor)

Calculations:

temperatureAdjustment = 1.0 + (tempAdjustTermA * (temperature-75.0)) + (tempAdjustTermB * (temperature-75) * (temperature-75))

K= 1.0 – ((boundedSpecificHumidity – 75.0) * humidityCorrectionCoeff)
Where
boundedSpecificHumidity = GREATEST(21.0,LEAST(specificHumidity,124.0))

Step CEIC-2: Calculate AC Adjustment Factor

This step has 3 substeps:

Step CEIC-2a: Calculate AC On Fraction

Input Variables:

heatIndex from ZoneMonthHour table
ACActivityTermA, B, and C from the MonthGroupHour Table
MonthOfAnyYear table used to associate month groups and months

Output Variables:

An intermediate ACOnFraction table:

Key Fields: zoneID, monthID, hourID
Date Field: ACOnFraction

Calculation:

ACOnFraction = ACActivityTermA + ACActivityTermB * heatIndex + ACActivityTermC * heatIndex * heatIndex

If ACOnFraction < 0.0 Then ACOnFraction = 0.0
If ACOnFraction > 1.0 Then ACOnFraction = 1.0

Step CEIC-2b: Calculate AC Activity Fraction

Input Variables:

ACOnFraction from previous sub-step
ACPenetrationFraction from SourceTypeModelYearTable
functioningACFraction from SourceTypeAge Table

Output Variables:

An intermediate ACActivityFraction table:

Key Fields: zoneID, monthID, hourID, sourceTypeID, modelYearID
Date Field: ACActivityFraction

Calculation:

ageID = calendar year – modelYearID
(needed to join to the SourceTypeAge table)

ACActivityFraction = ACOnFraction * ACPenetrationFraction * functioningACFraction

Step CEIC-2c: Calculate ACAdjustmentFraction

Input Variables:

ACActivityFraction from previous sub-step
fullACAdjustment from FullACAdjustment table

Output Variables:

An intermediate ACAdjustment table:

Key Fields: zoneID, monthID, hourID, sourceTypeID, modelYearID,
polProcessID
Date Field: ACAdjustment

Calculation:

Assume FullACAdjustment populated only for a single opModeID

ACAdjustment = 1.0 + ((fullACAdjustment-1.0) * ACActivityFraction)

Step CEIC-3: Calculate SourceBin-Weighted Emission Rates

This step weights emission rates, which vary by source bin, by the source bin distribution, retaining, however, the fuel type distinction because it is needed by subsequent steps and in the eventual output. The model year group distinction within source bin is subsumed by the broader model year distinction required in the output.

Input Variables:

meanBaseRate from the EmissionRate table
sourceBinActivityFraction from the SourceBinDistribution table

The SourceBin table is needed to decompose sourceBinID values into their constituent components.
The SourceTypeModelYear table is needed to decompose sourceTypeModelYearID values into their constituent sourceTypeID and modelYearID values.

Output Variables:

An intermediate SBWeightedEmissionRate Table

Key fields: polProcessID, sourceTypeID, modelYearID, fuelTypeID
Data field: meanBaseRate

Calculation:

In joining these tables it can be assumed that only one modelYearGroupID will be present in the source bin distributions for a polProcessID and modelYearID.

SBWeightedEmissionRate.meanBaseRate =
sum(sourceBinActivityFraction * EmissionRate.meanBaseRate)

Step CEIC-4: Apply Adjustment Factors to Emission Rates

Input Variables:

meanBaseRate from the SBWeightedEmissionRate table from the previous step
temperatureAdjustment and K from the METAdjustment table produced by step 1
ACAdjustment from the ACAdjustment table produced by step 2-c.

Output Variables:

An intermediate WeightedAndAdjustedEmissionRate table

Key fields: polProcessID, sourceTypeID, modelYearID, fuelTypeID, zoneID, monthID, hourID

Data field: meanBaseRate

Calculation:

K_ToUse = K if pollutantID=3 otherwise K_ToUse = 1.0

WeightedAndAdjustedEmissionRate.meanBaseRate = SBWeightedEmissionRate.meanBaseRate * temperatureAdjustment * ACAdjustment * K_ToUse

Step CEIC-5: Multiply Emission Rates by Activity

Input Variables:

meanBaseRate from WeightedAndAdjustedEmissionRate table from step 4.
extendedIdleHours from ExtendedIdleHours table.

HourDay table needed to decompose hourDayID values in extendedIdleHours into dayID and hourID values.

Output Variables:

emissionQuant in an intermediate AdjustedEmissionResults table

Key Fields: polProcessID, sourceTypeID, modelYearID, fuelTypeID, zoneID, monthID, hourID, dayID, yearID

Data Field: emissionQuant

Calculations:

ageID = calendar year – modelYearID
(needed to join to the extendedIdleHours table)

emissionQuant = meanBaseRate * extendedIdleHours

Step CEIC-6: Convert Results to Structure of MOVESWorkerOutput by sourceTypeID

Structurally we need to add stateID, countyID, linkID, roadTypeID , SCC (as null value) and decompose polProcessID into pollutantID and processID.

Input Variables:

stateID, countyID from MasterLoop Context
linkID from Link table
PollutantProcessAssoc table
AdjustedEmissionResults table from previous step

Output Variables:

MOVESWorkerOutput Table

Key fields: yearID, monthID, dayID, hourID, stateID, countyID, zoneID, linkID, pollutantID, processID ,sourceTypeID, fuelTypeID, modelYearID, roadTypeID, SCC

Data field: emissionQuant

Calculations:

MOVESWorkerOutput.emissionQuant =AdjustedEmissionResults.emissionQuant
stateID = stateID from MasterLoop context
countyID = countyID from MasterLoop context
roadTypeID = 1
linkID = value from Link table for zoneID where roadTypeID = 1
pollutantID = PollutantProcessAssoc.pollutantID
processID = PollutantProcessAssoc.processID
SCC = null

Step CEIC-7: Conditionally Convert Results to Structure of MOVESWorkerOutput by SCC

This step is only performed when the run specification requires output by SCC. It is performed in several other emission calculators; this description has been elaborated to explicitly consider SCCProcID.

Input Variables:

SCCVTypeFraction (from SCCVTypeDistribution table)
SCCRoadTypeFraction(from SCCRoadTypeDistribution table)
SCCProcID from EmissionProcess table
SCC from SCC table
emissionQuant from MOVESWorkerOutput table from previous step
sourceTypeID, modelYearID from SourceTypeModelYear

Output Variable:

emissionQuant (in restructured MOVESWorkerOutput table)

Calculation:

From join of SCCVtypeDistribution, SCC, EmissionProcess,
SourceTypeModelYear, and SCCRoadTypeDistribution where:
zoneID = zoneID from master loop context
SourceTypeModelYear.sourceTypeModelYearID =
SCCVtypeDistribution.sourceTypeModelYearID and
SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
SCC.SCCProcID = EmissionProcess.SCCProcID and
EmissionProcess.processID = processID from masterloop context

```
MOVESWorkerOutput.sourceTypeID = sourceTypeModelYear.sourceTypeID and
MOVESWorkerOutput.modelYearID = sourceTypeModelYear.modelYearID and
MOVESWorkerOutput.fuelTypeID = SCCVtypeDistribution.fuelTypeID and
MOVESWorkerOutput.roadTypeID = SCCRoadTypeDistribution.roadTypeID


    SCC= SCC.SCC
    emissionQuant = sum of emissionQuant * SCCVtypeFraction *
        SCCRoadTypeFraction over sourceTypeID and roadTypeID

    sourceTypeID = null
    roadTypeID = null
    linkID = null
```

10.28 Air Toxics Calculator

The MOVES model calculates emission rates and inventories for several air toxic pollutants and processes. The algorithm for the air toxics calculator can be found in the MOVES file AirToxicsCalculator.sql. The term air toxics and the particular compounds listed in MOVES are generally defined in the EPA Mobile Source Air Toxics (MSAT) rulemaking as ‘air toxics’. The MOVES list is not an exhaustive one of all vehicle related air toxic pollutants. Exclusion from the list does not imply that a particular compound (i.e., carbon monoxide) is non-toxic. The list of MOVES air toxics pollutants includes:

Benzene

Ethanol

MTBE

Naphthalene

1,3-Butadiene

Formaldehyde

Acetaldehyde

Acrolein

All of the air toxic pollutants have exhaust processes. However, only benzene, ethanol, MTBE and Naphthalene have evaporative processes. Ethanol and MTBE are not products of combustion so they may have zero emission rates if the fuel(s) used in the MOVES simulation are non-ethanol or non-MTBE fuels.

10.28.1. Air Toxics Calculator Overview

The MOVES model does not contain direct emission rates for any of the air toxic pollutants. All of them are computed in MOVES from ‘chained’ calculators using simple ratios. All pollutants, with the exception of Naphthalene, are ratios (and chained) to volatile hydrocarbon (VOC) emissions. Naphthalene is a ratio of Total PM10 emissions *(PM10 is in turn chained to PM2.5 emissions which is in turn chained to total energy emissions).*

The basic conceptual air toxics equations in MOVES are:

Air Toxic Emissions = Air Toxic Ratio * VOC Emissions Eq 10.28.1

Air Toxic Emissions = Air Toxic Ratio * PM10 Emissions Eq 10.28.2

All correction factors such as I/M, fuel effects, temperature, etc. are performed on the underlying VOC or Total PM10 emissions. Thus, the air toxic’s calculator does not apply any additional correct factor modules after Eq 10.28.1 or 10.28.2 is performed.

10.28.2. Air Toxics calculation

Because of size considerations, the air toxic ratios (see Eq 10.28.1) are stored in three separate MOVE database tables. These tables are ATRatioGas1, ATRatioGas2 and ATRatioNonGas. Tables ATRatioGas1 and ATRatioGas2 contain air toxic ratios for gasoline powered vehicles only. More specifically, MOVES table ATRatioGas1 contains air toxic ratios for pollutants Benzene, MTBE, 1,3-Butadiene, Formaldehyde and Acetaldehyde for the Running, Start and Extended Idle processes, and air toxics ratios for Benzene and MTBE for all of the evaporative processes. Table ATRatioGas2 contains air toxic ratios for pollutants Ethanol and Naphthalene for all exhaust and evaporative processes, and Acrolein air toxic ratios for all exhaust processes. Table ATRatioNonGas contains air toxic ratios for diesel and ethanol (E-85) powered vehicles for all pollutant / process combinations.

Input Variables:

From table MOVESWorkerOutput the following variables:

yearID, monthID, dayID, hourID, stateID, countyID, zoneID, linkID, sourceTypeID, .fuelTypeID, modelYearID, roadTypeID, SCC and emissionQuant

Because air toxics are chained pollutants, emissionQuant contains the VOC or Total PM10 emission quantity as it enters the air toxics calculator. The variable emissionQuant contains the air toxic pollutant quantity as it departs the air toxics calculator.

Other inputs are:

fuelsupply. marketShare
ATRatioGas1. ATRatio
ATRatioGas2. ATRatio
ATRatioNonGas. ATRatio

The value and table source for the variable ATRatio depends on which air toxic pollutant / process / fueltype is being calculated.

Output Variable:

MOVESWorkerOutput. emissionQuant

Calculation:

emissionQuant(air toxic compound) = AirToxicGas1.ATRatio
fuelSupply.marketShare * MOVESWorkerOutput. emissionQuant(VOC)

emissionQuant(air toxic compound) = AirToxicGas2.ATRatio *
fuelSupply.marketShare * MOVESWorkerOutput. emissionQuant(VOC)

emissionQuant(air toxic compound) = AirToxicNonGas.ATRatio *
fuelSupply.marketShare * MOVESWorkerOutput. emissionQuant(VOC)

for Naphthalene

emissionQuant(air toxic compound) = AirToxicGas2.ATRatio *
fuelSupply.marketShare * MOVESWorkerOutput. emissionQuant(PM10)

emissionQuant(air toxic compound) = AirToxicNonGas.ATRatio *
fuelSupply.marketShare * MOVESWorkerOutput. emissionQuant(PM10)

10.29. Permeation Calculator

10.29.1. General Description

This component signs up with the MOVES master looping mechanism at the MONTH level.

Its inputs include:

Source Bin Distributions
Operating Mode Distributions
The EmissionRateByAge table
The AverageTankTemperature table
The TemperatureAdjustment Table
The FuelAdjustment Table (and associated tables used to determine fuel market shares)

Its results are placed in the same MOVESOutput table format as other EmissionCalculators.

10.29.2. Detailed Calculation Steps

PC-1: Weight emission rates by source bin

Input Variables:

- meanBaseRate (from EmissionRateByAge table)
- sourceBinActivityFraction (from SourceBinDistribution table)

Output Variables:

- meanBaseRate (intermediate table SBWeightedPermeationRate)

Calculation:

meanBaseRate(zoneID, yearID, polProcessID, sourceTypeID, modelYearID, fuelTypeID)
= sum (sourceBinActivityFraction * meanBaseRate)

PC-2: Calculate weighted temperature adjustment

Inputs:

- averageTankTemperature (from averageTankTemperature)
- opModeFraction (from operatingModeDistribution)
- tempAdjustTermA (from temperatureAdjust)
- tempAdjustTermB (from temperatureAdjust)

Outputs:

- weightedTemperatureAdjust (sourceTypeID, monthID, hourDayID, linkID, tankTemperatureGroupID)

Preliminary Calculation: temperatureAdjustByOpmode (zoneID, monthID, hourDayID, hourDayID, tankTemperatureGroupID, opModeID)
$= \text{tempAdjustTermA} * e^{\text{tempAdjustTermB} * \text{averageTankTemperature (opModeID)}}$

Calculation: weightedTemperatureAdjust
= SUM(temperatureAdjustByOpMode*opModeFraction) over all modes

PC-3: Calculate weighted fuel adjustment

Input Variables:

- fuelAdjustment (from fuelAdjustment table)
- marketShare (from fuelSupply table)

Output Variables:

- weightedFuelAdjust(polProcessID, modelYearGroupID, sourceTypeID, fueltypeid)

Calculation: weightedFuelAdjust
= SUM(fuelAdjustment * marketShare) over all fuel formulations for the county, fuelyear, month and fueltypeid.

NOTE: An internal MOVES model component, the DefaultDataMaker, has set fuelAdjustment=1 for cases which are not populated in fuelAdjustment.

PC-4: Calculate fuel adjusted meanBaseRate

Input Variables:

- meanBaseRate (SBWeightedEmissionRate from PC-1)
- weightedFuelAdjust (PC-3)

Output Variables:

- FuelAdjustedMeanBaseRate (zoneID, yearID, polProcessID, sourceTypeID, modelYearID, fuelTypeID)

Calculation: FuelAdjustedMeanBaseRate
= meanBaseRate * weightedFuelAdjustment

PC-5: Calculate fuel adjusted emissionQuant

Input Variables:

- fuelAdjustedMeanBaseRate (PC-4)
- SourceHours (CMIT from TAG)

Output Variables:

- fuelAdjustedEmissionQuant (linkID, hourDayID, monthID, yearID, modelYearID, sourceTypeID, fueltypeID)

Calculation:

- fuelAdjustedEmissionQuant
= fuelAdjustedMeanBaseRate * sourceHours

PC-6: Calculate emissionQuant with temperature adjustment

Input Variables:

- fuelAdjustedEmissionQuant (PC-5)
- weightedTemperatureAdjustment (PC-2)

Output Variables:

- emissionQuant (MOVES output table fields)

Calculation:

- emissionQuant = fuelAdjustedEmissionQuant * weightedTemperatureAdjustment

PC-7 Convert to SCC

This step is only performed when the run specification requires output by SCC

Input Variables:

SCCVTypeFraction (from SCCVTypeDistribution table)
SCCRoadTypeFraction(from SCCRoadTypeDistribution table)
SCC from SCC table
emissionQuant from MOVESWorkerOutput table from step 6)
sourceTypeID, modelYearID from SourceTypeModelYear

Output Variable:

emissionQuant (in MOVESWorkerOutput table)

Calculation:

```
From join of SCCVtypeDistribution, SCC, SourceTypeModelYear, and
    SCCRoadTypeDistribution where:
            zoneID = zoneID from master loop context
            SourceTypeModelYear.sourceTypeModelYearID =
                SCCVtypeDistribution.sourceTypeModelYearID and
            SCC.SCCVtypeID = SCCVtypeDistribution.SCCVtypeID and
            SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
            MOVESWorkerOutput.sourceTypeID = sourceTypeModelYear.sourceTypeID and
                SCC.SCCroadTypeID=SCCRoadTypeDistribution.roadTypeID and
            MOVESWorkerOutput.modelYearID = sourceTypeModelYear.modelYearID and
            MOVESWorkerOutput.fuelTypeID = SCCVtypeDistribution.fuelTypeID and
            MOVESWorkerOutput.roadTypeID = SCCRoadTypeDistribution.roadTypeID


                SCC= SCC.SCC
                emissionQuant = sum of emissionQuant * SCCVtypeFraction *
                    SCCRoadTypeFraction over sourceTypeID and roadTypeID

                sourceTypeID = null
                roadTypeID = null
                linkID = null
```

10.30. Liquid Leaking (LL) Calculator

10.30.1. General Information

This calculator executes at the MONTH master looping level.

Input Tables

- OpModeDistribution
- SourceHours
- EmissionRateByAge
- SourceBinDistribution
- IMCoverage
- Miscellaneous MOVES Database category and association tables

Output Table: MOVESWorkerOutput (has structure of MOVESOutput Table)

10.29.2. Detailed Calculation Steps

LL – 1 Compute I/M Adjustment Fraction Information
(same as CREC 1-a except for different value(s) of pollutant-process)

Input Variables:

IMCoverage table
zoneID, yearID, polProcessID from masterloop context
AgeCategory table
RegulatoryClass Table
FuelType table

Input Variable:

IMAdjustment Table
Keys: zoneID, yearID, polProcessID, modelYearID, fuelTypeID, regClassID
Data: IMAdjustFract

Calculation:

For zoneID, yearID in masterloop context
For Vapor Venting and Liquid Leaking processes of all pollutants in runspec which this calculator calculates
For all ageID in AgeCategory
For all regClassID in RegulatoryClass except regClassID=0
For FueltypeID = 1 & 5 (right?)

modelYearID = yearID – ageID
IMAdjustFract.IMAdjustFract = IMCoverage.IMAdjustFract if record exists
With begModelYearID <= modelYearID <= endModelYearID
= 0.0 otherwise

LL-2: Calculate I/M-Adjusted MeanBaseRates

Input Variables:

- EmissionRateByAge table
- SourceBinDistribution table
- SourceBinTable
- IMAdjustment table (from step LL-1)
- AgeCategory
- PollutantProcessModelYear

Output Variables:

An intermediate WeightedMeanBaseRate table

- Keys: yearID, polProcessID, sourceTypeID, fuelTypeID, zoneID, monthID, hourDayID, modelYearID, opModeID
- Data: weightedMeanBaseRate

Calculation:

modelYearID = yearID – ageID
fuelTypeID = 1 (gasoline) or 5 (E85)

weightedMeanBaseRate = (meanBaseRateIM * sourceBinActivityFraction * IMAdjustFract) + (meanBaseRate * sourceBinActivityFraction * (1-IMAdjustFract))

summed over portions of sourceBinID not needed in the output, namely regClassID and engTechID.

LL-3: Calculate MOVESWorkerOutput by Source Type and FuelType

Input Variables:

- WeightedMeanBaseRate table from previous step
- SourceHours table
- OpModeDistribution table
- HourDay table
- County table
- Zone table
- PollutantProcessAssoc table
- Link table

Output Variables:

- MOVESWorkerOutput table, which has structure of MOVESOutput

Calculation:

```
emissionQuant
        = weightedMeanBaseRate * sourceHours * opModeFraction
hourID = hourDay.hourID
dayID = hourDay.dayID
stateID = county.stateID
countyID =  zone.countyID
pollutantID = PollutantProcessAssoc.pollutantID
processID = PollutantProcessAssoc.processID
roadTypeID = Link.roadTypeID
SCC = null
```

LL- 4 Conditionally Convert SourceType Output to SCC

This step is performed only when output by SCC is requested by the run specification and, if performed, is the same as that performed in several other EmissionCalculators, e.g. CREC step 10.

10.31. HC Speciation Calculator (HCSC) Calculator

10.31.1. General Information

This calculator gives the model the ability to calculate NMHC (Non-Methane Hydrocarbons, pollutant 79), NMOG (Non-Methane Organic Gases, pollutant 80), TOG (Total Organic Gases, pollutant 86), and VOC (Volatile Organic Compounds, pollutant 87). The basic design of the calculator is to compute the results as linear functions of chained pollutants and fuel-formulation-specific constants. The input pollutants vary according to the output and the type of fuel. Also, being dependent upon fuel formulations not just fuel types, market shares are factored into the results.

10.30.2. Detailed Calculation Steps

The HCSpeciation table provides formula parameters, with most formulas being of the form:

outputQuant = sum of (inputQuant * fuel formulation market share * (speciationConstant + oxySpeciation*volToWtPercentOxy* sum of oxygenates in the fuel formulation)) across all fuel formulations in a county

There is an added condition on the above: if a fuel has no oxygenates, it has no HC speciation output. This is not a mere algebraic outcome of the above equation which would still yield a non-zero number given the speciationConstant term even with zero oxygenates. All HC speciation equations that require oxygenates are subject to this conditional logic.

All HC speciation calculations have common table connectivity, differing only in their formula for calculating new emission quantities and criteria for input pollutants and fuels. To perform HC speciation, connect the following tables and columns:

- MOVESWorkerOutput.fuelTypeID with FuelSupply.fuelTypeID
- MOVESWorkerOutput.countyID with FuelSupply.countyID
- MOVESWorkerOutput.yearID with FuelSupply.yearID

- MOVESWorkerOutput.monthID with FuelSupply.monthID
- MOVESWorkerOutput.modelYearID with PollutantProcessModelYear.modelYearID
- MOVESWorkerOutput.processID with PollutantProcessAssoc.processID
- FuelSupply.fuelFormulationID with FuelFormulation.fuelFormulationID and with HCSpeciation.fuelFormulationID
- HCSpeciation.fuelMYGroupID with PollutantProcessModelYear.fuelMYGroupID
- HCSpeciation.polProcessID with PollutantProcessModelYear.polProcessID and PollutantProcessAssoc.polProcessID

The output of the calculations has the same dimensional values as each input MOVESWorkerOutput table row except that the output pollutantID should be taken from PollutantProcessAssoc.pollutantID. All outputs are summed across the contributing fuel formulations and weighted according to market share.

For non-E85, non-E70 fuels:

NMHC = (THC – Methane) * marketShare

Thus, for an input record of THC, an equal amount of NMHC should be generated. For Methane input, the output is NMHC = –Methane. Worker-side and master-side aggregation logic will sum the values completing the formula.

For E85 or E70 fuels:

NMHC=THC * (speciationConstant + oxySpeciation * volToWtPercentOxy * ETOHVolume) * marketShare

For non-E85, non-E70 fuels:

NMOG = NMHC * (speciationConstant + oxySpeciation * volToWtPercentOxy * (MTBEVolume + ETBEVolume + TAMEVolume + ETOHVolume)) * marketShare

For E85 or E70 fuels:

NMOG=THC * (speciationConstant + oxySpeciation * volToWtPercentOxy * ETOHVolume) * marketShare

For non-E85, non-E70 fuels:

VOC = NMHC * (speciationConstant + oxySpeciation * volToWtPercentOxy * (MTBEVolume + ETBEVolume + TAMEVolume + ETOHVolume)) * marketShare

For E85 or E70 fuels:

VOC=THC * (speciationConstant + oxySpeciation * volToWtPercentOxy * ETOHVolume) * marketShare

For non-E85, non-E70 fuels:

TOG = (NMOG + Methane) * marketShare

Thus, for an input record of NMOG, an equal amount of TOG should be generated. For Methane input, an equal amount of TOG should be generated. Worker-side and master-side aggregation logic will sum the values completing the formula.

For E85 or E70 fuels:

TOG=THC * (speciationConstant + oxySpeciation * volToWtPercentOxy * ETOHVolume) * marketShare

10.32. Tank Vapor Venting (TVV) Calculator

10.32.1. General Information

This calculator executes at the MONTH master looping level.

Input Tables

- TankVaporGenCoeffs
    - Keys: ethanolLevelID (0 or 10), altitude ("H" or "L")
    - Data: tvgTermA, tvgTermB, tvgTermC
- CumTVVCoeffs (Analogous to EmissionRateByAge, knowing that there is only one opModeID and that the particular sourceBinID components needed are regClassID and modelYearGroupID)
    - Keys: polProcessID, regClassID, modelYearGroupID, ageGroupID
    - Data: tvvTermA, tvvTermB,tvvTermC
      tvvTermACV,tvvTermBCV, tvvTermCCV
      tvvTermAIM,tvvTermBIM,tvvTermCIM
      tvvTermAIMCV,tvvTermBIMCV,tvvTermCIMCV
- ResidualVaporRatio (a hard-coded Java array rather than an actual table)
    - Key: "hours past hour of max cold soak tank temperature"
    - Data: residualVaporRatio
- AverageTankGasoline (Produced by Tank Fuel Generator)
    - Keys: zoneID, fuelYearID, monthGroupID, fuelTypeID
    - Data: ETOHVolume, RVP
- ColdSoakTankTemperature (from TTG-1)
    - Keys: zoneID, monthID, hourID
    - Data: coldSoakTankTemperature
- ColdSoakInitialHourFraction (from TTG-7)
    - Keys: sourceTypeID, zoneID, monthID, hourDayID, initialhourDayID
    - Data: coldSoakInitialHourFraction

- OpModeDistribution
- SourceHours
- EmissionRateByAge
- SourceBinDistribution
- IMCoverage
- Miscellaneous MOVES Database category and association tables

Output Table: MOVESWorkerOutput (has structure of MOVESOutput Table)

Calculation Overview:

The complete calculation at Macroscale requires up to ten steps. Steps 2 thru 7 apply only to the cold soaking operating mode and are not required for Mesoscale Lookup calculations.

TVV-1 is a preliminary step. It calculates the I/M adjustment factors which are used later in the calculation.

TVV-2 is also a preliminary step. It calculates the hour of the day when the peak cold soak tank temperature value occurs.

TVV-3 calculates the amount of tank vapor generated, which depends upon gasoline and E85 ethanol contents.

TVV-4 calculates an ethanol-weighted values of the tank vapors generated from the values in step 3.

TVV-5 calculates the tank vapor vented using the values in step 4. The values calculated here are cumulative for the day and are distinguished by the hour that the cold soaking began, as well as the hour of the day when they occur.

TVV-6 calculates the total cumulative tank vapor vented for each hour of the day from the values in step 5, so that they are no longer distinguished by the hour of the day when the cold soaking began.

TVV-7 calculates hourly (not cumulative) tank vapor vented.

TVV-8 Involves all three operating modes and applies the I/M adjustment factors calculated in preliminary step 1.

TVV-9 multiplies by activity and applies the operating mode distribution.

TVV-10 converts results to SCC if required.

10.32.2. Detailed Calculation Steps

Note: Steps TVV-2 through TVV-7 need not be performed for mesoscale lookup runs.

TVV – 1 Compute I/M Adjustment Fraction Information (same as CREC 1-a except for different value(s) of pollutant-process)

Input Variables:

IMCoverage table
zoneID, yearID, polProcessID from masterloop context
AgeCategory table
RegulatoryClass Table
FuelType table

Output Variable:

IMAdjustment Table
Keys: zoneID, yearID, polProcessID, modelYearID, fuelTypeID, regClassID
Data: IMAdjustFract

Calculation:

```
For zoneID, yearID in masterloop context
For Vapor Venting and Liquid Leaking processes of all pollutants in runspec
which this calculator calculates
For all ageID in AgeCategory
For all regClassID in RegulatoryClass except regClassID=0

modelYearID = yearID – ageID
IMAdjustFract.IMAdjustFract = IMCoverage.IMAdjustFract if record exists
        With begModelYearID <= modelYearID <= endModelYearID
        = 0.0 otherwise
```

TVV – 2 Determine Hour of Peak Cold Soak Tank Temperature

Input Variable:

ColdSoakTankTemperature table

Output Variables:

intermediate PeakHourOfColdSoak table
Keys: zoneID, monthID
Data: peakHourID

Calculation:

peakHourID is the first (smallest) hourID having the highest coldSoakTankTemperature for the zoneID and monthID

TVV – 3 Calculate TankVaporGenerated (TVG) by Ethanol Level

Input Variables:

- ColdSoakInitialHourFraction table
- PeakHourOfColdSoak table from previous step
- ColdSoakTankTemperature table
- TankVaporGenerationCoeffs table
- AverageTankGasoline table
- HourDay table
- County table

Output Variables:

An intermediate TankVaporGenerated table
Keys: hourDayID, initialhourDayID, ethanolLevelID, monthID, zoneID, sourceTypeID, fuelYearID, fueltypeID
Data: tankVaporGenerated

Calculation:

Calculation is limited to:

monthID, zoneID, sourceTypeIDs and fueltypeids in the masterloopcontext and the run specification
combinations of hourDayID and initialhourDayID present in ColdSoakInitialHourFraction for which initialhourDayID <> hourDayID
hourDayID values having an hourID value <= PeakHourOfColdSoak.peakHourID

tankVaporGenerated = 0.0 if t1>=t2
otherwise
tankVaporGenerated = $(a\ e^{b\ (RVP)}\ (e^{ct2} - e^{ct1})) * k$

Where:
t2 = ColdSoakTankTemperature of hourID associated with hourDayID
t1 = ColdSoakTankTemperature of hourID associated with initialhourDayID
k = 50% fill adjustment constant to be determined
a, b, and c are terms from TankVaporGenCoeffs table
RVP is from AverageTankGasoline table by fuelTypeID

The altitude of the County to which the zoneID belongs should be used when selecting the a,b, and c coefficients from the TankVaporGenCoeffs table.

TVV-4 Calculate ethanol-weighted TVG

Input Variables:

- TankVaporGenerated table from previous step
- EtOH Volume value from AverageTankGasoline table

Output Variables:

- An intermediate EthanolWeightedTVG table
  - Key fields: hourDayID, initialhourDayID, monthID, zoneID, sourceTypeID, fuelYearID, fueltypeID
  - Data: ethanolWeightedTVG

Calculation:

ethanolWeightedTVG
$= \text{tankVaporGenerated}_{E10} \text{ (ETOHVolume / 10) + tankVaporGenerated}_{E0} \text{ (1 – (ETOHVolume/ 10))}$

TVV-5: Calculate Cumulative Tank Vapor Vented (TVV)

Input Variables:

- EthanolWeightedTVG table from previous step
- CumTVVCoeffs table
- PollutantProcessModelYear
- AgeCategory

Output Variables:

An intermediate CumulativeTVV table:

Key fields: regClassID, ageID, polProcessID, hourDayID, initialhourDayID, monthID, zoneID, sourceTypeID, fuelTypeID

Data fields: tankVaporVented, tankVaporVentedIM

Calculation:

For all ageID:

$\text{tankVaporVented} = \text{tvvTermA} + \text{tvvTermB*ethanolWeightedTVG} + \text{tvvTermC*ethanolWeightedTVG}^2$

$\text{tankVaporVentedIM} = \text{tvvTermAIM} + \text{tvvTermBIM*ethanolWeightedTVG} + \text{tvvTermCIM*ethanolWeightedTVG}^2$

The structure of the output table specified here implies that fuelYearID, ageGroupID, and modelYearGroupID be converted to the common basis of individual ageID to facilitate proper table joining and minimize the size of the resulting table (which nevertheless may be performance-constraining). The PollutantProcessModelYear, and

AgeCategory tables can be used to accomplish this, along with the relationship ageID = yearID – modelYearID.

TVV-6: Calculate Weighted CumulativeTVV Across Initial/Current pair

Input Variables:

- CumulativeTVV Table from previous step
- ColdSoakInitialHourFraction table

Output Variables:

- An intermediate WeightedCumulativeTVV Table

Key fields: regClassID, ageID, polProcessID, hourDayID, monthID, zoneID, sourceTypeID, fuelTypeID

Data fields: weightedTVV, weightedTVVIM

Calculation:

For each combination of regClassID, ageID, polProcessID, hourDayID, monthID, zoneID, and sourceTypeID, fuelTypeID

weightedTVV = [tankVaporVented * coldSoakInitialHourFraction] summed over all initialHourIDs

weightedTVVIM = [tankVaporVentedIM * coldSoakInitialHourFraction] summed over all initialHourIDs

TVV-7: Calculate HourlyTVV Emissions by RegulatoryClass and Vehicle Age

Input Variables:

- WeightedCumulativeTVV Table (from previous step)
    - LEFT JOINED to itself associating each hourID with otherwise same record for hourID-1, if present.
- ResidualVaporRatio array; Values are:
    - 0.02 for 1 hour past max cold soak temperature
    - 0.01 for 2 hours past max cold soak temperature
    - 0.004 for 3 hours past max cold soak temperature
    - 0.0005 for 4 hours past max cold soak temperature
    - 0.0 for 5 or more hours past max cold soak temperature
- PeakHourOfColdSoak table from step TVV-2
- HourOfAnyDay

Output Variable:

an intermediate HourlyTVV table

- o key fields: regClassID, ageID, polProcessID, hourDayID, monthID, zoneID, sourceTypeID, fuelTypeID
- o data fields: hourlyTVV, hourlyTVVIM

Calculation:

For records in WeightedCumulativeTVVTable (which have hourIDs prior or equal to the hour of the peak cold soak tank temperature because of the limited domain of the calculation in step 3)

hourlyTVV = weightedTVV for the current hourID – weightedTVV for the previous hourID, (if no record is present for the previous hourID then the weighted TVV for the previous hourID is 0.0)

hourlyTVVIM = weightedTVVIM for the current hourID – weightedTVVIM for the previous hourID, (if no record is present for the previous hourID then the weighted TVVIM for the previous hourID is 0.0)

For all hourDayIDs in HourDay which have hourDayIDs greater than PeakHourOfColdSoak.peakHourID:

hourlyTVV for each such hourID = weightedTVV for peakHourID * the residualVaporRatio for the number of hours the hourID is past the peakHourID (hourID-peakHourID).

hourlyTVVIM for each such hourID = weightedTVVIM for peakHourID * the residualVaporRatio for the number of hours the hourID is past the peakHourID (hourID-peakHourID).

TVV-8: Calculate I/M-Adjusted MeanBaseRates

Previous steps have been done only for the cold soaking operating mode. This step begins calculating for the other two operating modes,”operating” and “hot soaking”, for which only the basic core model calculations are needed, applies the source bin distributions, and accounts for the effect of IM.

Input Variables:

:

- HourlyTVV table (from previous step), used for cold soak
- EmissionRateByAge table, used for hot soak and operating
- SourceBinDistribution table
- SourceBinTable
- IMAdjustment table (from step TVV-1)
- AgeCategory
- PollutantProcessModelYear

Output Variables:

:

An intermediate WeightedMeanBaseRate table

- Keys: yearID, polProcessID, sourceTypeID, fuelTypeID, zoneID, monthID, hourDayID, modelYearID, opModeID
- Data: weightedMeanBaseRate

Calculation:

modelYearID = yearID – ageID
fuelTypeID = 1 (gasoline) or 5 (E85)

For cold soak mode (opModeID=151):

weightedMeanBaseRate = (hourlyTVVIM * sourceBinActivityFraction * IMAdjustFract) + (hourlyTVV * sourceBinActivityFraction * (1-IMAdjustFract))

summed over portions of sourceBinID not needed in the output, namely regClassID and engTechID

For operating and hot soaking modes (opModeIDs 300 and 150):

weightedMeanBaseRate = (meanBaseRateIM * sourceBinActivityFraction * IMAdjustFract) + (meanBaseRate * sourceBinActivityFraction * (1-IMAdjustFract))

summed over portions of sourceBinID not needed in the output, namely regClassID and engTechID.

TVV-9: Calculate MOVESWorkerOutput by Source Type

Input Variables:

- WeightedMeanBaseRate table from previous step
- SourceHours table
- OpModeDistribution table
- HourDay table
- County table
- Zone table
- PollutantProcessAssoc table
- Link table

Output Variables:

- MOVESWorkerOutput table, which has structure of MOVESOutput

Calculation:

emissionQuant

```
            = weightedMeanBaseRate * sourceHours * opModeFraction
        hourID = hourDay.hourID
        dayID = hourDay.dayID
        stateID = county.stateID
        countyID =  zone.countyID
        pollutantID = PollutantProcessAssoc.pollutantID
        processID = PollutantProcessAssoc.processID
        roadTypeID = Link.roadTypeID
        SCC = null
```

TVV-10 Conditionally Convert SourceType Output to SCC

This step is performed only when output by SCC is requested by the run specification and, if performed, is the same as that performed in several other EmissionCalculators, e.g. CREC step 10.

10.33. Result Data Aggregation and Engineering Units Conversion

This function aggregates the results produced by MOVES EmissionCalculators to the level of detail called for in the run specification and converts these results to the engineering units it specifies. Aggregation is performed to the extent possible by the MOVES Worker program and completed by the MOVES Master program. Conversion to engineering units is the final operation performed and is done by the MOVES Master program.

10.33.1. How Aggregation Levels are Specified

The level of aggregation is specified on the MOVES GUI Output EmissionsDetail Screen. These specifications are made in terms of what distinctions are desired in the output. Output rows are always distinguished by time periods. The level of this distinction may be hour, single day, period of week, month or year. (Choices may be more limited, however, if time period “preaggregation” of the database was performed.) Output rows are always distinguished by location. The level of this distinction may be Nation, State, County, Roadtype or Link. (Choices may be more limited, however, if geographic “preaggregation” of the database was performed. Link is not available at Macroscale and is required for Mesoscale Lookup calculations.) Output is always distinguished by pollutant.

When reporting for entire months or years, DRAFT MOVES2009 scales the results up by the number of weeks in each month, i.e. by the number of days in the month divided by seven. The reader may wish to refer to section 9.2 of this document where MOVES time periods are discussed.

Output may be distinguished by SourceUseType (the recommended option), Source Code Category (SCC) or neither. (But not both since the two highway vehicle classification schemes are exclusive.)

Output may optionally be distinguished by:

Model Year
Fuel Type

Emission Process

Roadtype

In general, any combination of these distinctions may be specified. Output by SCC implies that roadtype and fueltype will be distinguished and the MOVES GUI enforces this. If “Roadtype” is selected as the Location level then “Roadtype” is automatically distinguished in the output (but the reverse is not necessarily true).

10.33.2.Aggregation Algorithm - Logical Level Specification

The raw output of MOVES is distinguished by year, month, day, hour, state, county, zone, link, road type, pollutant, process, source use type, fuel type, model year, and SCC. Taken together these fields may be considered an alternate key for the MOVESOutput table, except that, once aggregations are performed, they may assume the null value.

At macroscale, zones are redundant with counties, so whenever a countyID is present, its corresponding zoneID is also present, and when counties are aggregated out, then so are zones. Also at macroscale, links represent a combination of a county and a roadtype, so when county and roadtype are both known, so is link. Otherwise linkID is null whenever either county or roadtype is null.

The following aggregations may be carried out as implied by the RunSpec. Unless stated otherwise, any combination of these aggregations may be called for. This description is at a logical level; physical implementation differs. For example aggregation to the state level is not actually performed by aggregating roadtypes to counties and then aggregating counties to states.

a. Output may be aggregated to be by source type only, by SCC only or neither. The GUI does not allow output to be requested by both SCC and source type.

b. Hours are combined if the output time period is 24-hour day, portion of week, month or year. A warning message is generated if this aggregation is performed and all hours are not selected in the RunSpec.

c. Days are converted to months if the output time period is month or year. The number of days in each month is obtained from the MonthOfAnyYear table, adding 1 day to February for leap years. 1/7 of these days are considered to be Mondays, 1/7

Tuesdays, etc. without regard to calendar year . A warning is generated if this aggregation is performed and all days are not selected in the RunSpec. The user may choose to proceed but should be aware in this case that results produced for the "month" or "year" will only include emission results for the kinds of days contained in the run specification.

d. Months are totaled to years if the output time period is year. A warning is generated if this aggregation is performed and all months are not selected. The user may choose to proceed but should be aware in this case that results produced for the "year" will not represent a 12 month period and will only include emissions for the months contained in the run specification.

e. Road types are combined into county totals whenever "roadtype" has not been selected in the Output Emissions detail screen. Note that selection of the SCC level of detail forces road type to be selected and this aggregation not to be performed, as does selection of the roadtype level of geographic output detail. A warning message is generated if this aggregation is performed and all road types are not included in the runspec.

f. State totals are combined into a single national (or total user modeling domain) result if the national level of geographic detail is selected. This aggregation is performed even if the road type level of detail has been selected on the "Output Emission Detail" panel. It is the user's responsibility to insure that all desired states are included in domain totals.

g. Fuel types are combined within source use types if this level of detail is not selected in the Output Emission Detail panel. It is the user's responsibility to insure that all desired fuel types are included.

h. Emission processes are combined if this level of detail is not selected in the Output Emission Detail panel. It is the user's responsibility to insure that all desired processes are included.

10.33.3. Engineering Unit Conversion

Engineering units are indicated in the run specification for mass, energy, time and distance. In the MOVES GUI these are specified on the “Outputs” panel within the “General Output Screen”. Supported are:

time units (seconds, hours, days, weeks, months and years)
mass units (kilograms, grams, pounds, and U.S. tons)
energy units (Joules, kiloJoules, and million BTU)
distance units (miles and kilometers)

The engineering units used are reported in the MOVESRun table.

Distance units are only required if the run specification calls for travel distance to be reported.

MOVES always reports mass pollutant emission results in terms of mass per time unit and always reports energy consumptions results in terms of energy per time unit. If the time unit equals the output time period specified on the “Output Emissions Detail” screen, then the quantities reported amount to an inventory for that time period.

10.34. Post-Processor for Mesoscale Lookup

An “integrated post-processor” runs automatically when Mesoscale Lookup is selected, after aggregation to the reporting level and conversion to engineering units have been completed. This post-processor uses the regular MOVES output tables to produce an additional “MOVESLookupOutput” database table with the following fields:

a. MOVESLookupOutputRowID
b. MOVESRunID
c. iterationID
d. yearID
e. monthID (May be aggregated out)
f. dayID   (May be aggregated out)
g. hourID (May be aggregated out)
h. stateID
i. countyID
j. zoneID
k. sourceTypeID (May be aggregated out)
l. fuelTypeID (May be aggregated out)
m. modelYearID (May be aggregated out)
n. roadTypeID
o. pollutantID
p. processID (May be aggregated out)
q. averageSpeedBinID (as determined through joins with the LinkAverageSpeed Table)
r. temperature (as determined through joins with the ZoneMonthHour table)
s. humidity (as determined through joins with the ZoneMonthHour table)
t. emissionRate (MOVESMesoscaleOutput.emissionQuant divided by MOVESMesoscalActivityOutput.activity (for all activityTypeID=1 records). “Zero” miles for any link leads to divide-by-zero errors. In such cases there are no emissions reported.

10.35. Post-Processing Script Execution

The “Post Processing” menu in the MOVES GUI includes a selection to “Run MySQL Script on Output Database”. When this item is selected MOVES searches the “OutputProcessingScripts”subdirectory in its “...database” directory, and presents a list of file names found there which have a file name extension of “.sql”. Users may wish to add scripts to those distributed with the model.

When one of these files is selected, MOVES displays to the user any leading block of comment lines it may contain in a popup window. If the user wants to proceed, MOVES executes the file as a MySQL script on the MOVES output database specified by the currently active run specification (or gives an appropriate error message if no output database is specified). Scripts in the OutputProcessingScripts folder should operate only on the currently active MySQL database, should not include the MySQL USE command, should use as input only the tables contained in the MOVES Output database schema, (along possibly with MOVESDefault) and should store any files and tables they produce in this same database. The MOVES program, however, does not enforce these conventions.

Scripts distributed with the model are:

Script	**Function**
DecodeMovesOutput.sql	Decodes (i.e. adds textual descriptions of) most of the key fields in MOVESOutput and MOVESActivityOutput tables. The SummaryReporter, described in the next section, does this in a more general fashion and so this is now best viewed as an example of how these scripts can be written.
TabbedOutput.sql	Produces tab-delimitted output file versions of the MOVESRun, MOVESActivityOutput, and

	MOVESOutput tables. These are suitable for importing into EXCEL or other software. The same thing can no be accomplished by using the SummaryReporter, described in the next section.

10.36. Summary Reporter

Background:

The output directly available from the MOVES core model is in the form of a MySQL database as documented in Chapter 12. The Summary Reporter provides a "post-processing" function which makes it easy to produce various reports summarizing this information. Several of the organizations which commented formally on DRAFT MOVES2009 indicated that a more convenient reporting capability was needed.

Functional Specification:

Inspired by the NONROAD Model's reporting utility, the Summary Reporter produces reports consisting of:

1. Header information such as:

    Report Title
    Date and time the report produced.
    MOVES Output Database Name
    MOVES Output Run Number
    MOVES Output Run Date and Time
    RunSpec used
    Date and time of the runspec file used.
    "Description" field from the Runspec used
    Engineering units used for Mass, Energy, Time, and Distance
    Emission Process (either a particular emission process or "all")

2. A tabular report body where the data columns correspond to pollutants and activity basis results (currently "distance") and the rows correspond to an ordered set of MOVES output classifications selected from the following options:

    Year
    Month

Day

Hour

State

County

Zone

Source Type (mutually exclusive with SCC)

SCC (mutually exclusive with Source Type)

Fuel Type

Model Year

Road Type (not allowed if SCC selected)

MOVESRunID

Assuming the report is printed in landscape, there is sufficient space to report four or five pollutants and activities, e.g. HC, CO, NOx and distance, classified by three categories. Classification values are reported in their integer form. Specification of more columns than will fit on a single page is allowed. The supporting GUI includes an option to estimate how wide the report will be. Wide reports are fine as a table or .csv file, but wrap when displayed on the screen or printed.   The fixed-column style report tables produced by this utility could be further processed with a reporting utility to deal more gracefully with wide reports, but MOVES does not currently provide this feature.

3. The current version of Summary Reporter reports numeric category codes in the table constituting the report body, and includes an appended lookup table which can be used to decode them.

The Summary Reporter includes a simple Graphical User Interface which works roughly as follows:

1. The user loads the run specification which produced the output. This serves to point to the output database and helps the GUI insure that the specified report is consistent with the run.

2. The user selects from a popup list the run, or set of runs to be reported. Only the most recent runs are available.

3. The Summary Reporter, based on the run specification and the output run data, constructs a dialog window (JDialog) to get user selections for:

   a. The report title. This defaults to “Summary Report”.
   b. A base name for the report files. This defaults to “SummaryReport”. (Report header, body, and decoding tables are produced in separate files, plus several forms of output (table, .prn, and tab-separated .txt) can be produced) .
   c. The emission process to be reported or “all”.
   d. A list of pollutants and activity bases (currently “distance”) from which the user selects the report data columns desired.
   e. Selections from the list of output classifications listed above. Only options consistent with the run specification are offered.
   f. The forms of output desired.

4. Several forms of output may be selected:

   a. MySQL table output is always produced.

   b. Option for screen display (Screen reports can be printed after being displayed on the screen; they are then closed.)

   c. Option for tab-separated text form.

Report output is produced in the directory containing the output database.

Additional Information:

The Summary Reporter is covered from a GUI perspective in the *DRAFT MOVES2009 User Guide*.

10.37. GREET Model Interface

The GREET Model Interface is not operational in the version of DRAFT MOVES2009. We are working with the authors of GREET to restore this function in future versions of MOVES. This section describes how the interace functions in DRAFT MOVES2009.

MOVES is designed to interface with GREET, as detailed in the following section. Further detail on the GREET interface itself is contained in a separate document entitled “User Manual and Technical Issues of GREET for MOVES Integration”, prepared by Argonne National Laboratory.

10.37.1 DRAFT MOVES2009 GREET Interface Functionality:

1. If requested in the RunSpec, DRAFT MOVES2009 calculates well-to-pump inventories for total energy consumption, petroleum-based energy consumption, fossil fuel-based energy consumption, N2O, and CH4, using the emission factors in the GREETWellToPump table.

2. DRAFT MOVES2009 offers a “preprocessing” menu option to “Update Well-To-Pump” factors. When this menu option is selected DRAFT MOVES2009 produces, in XML format, a table of information needed by GREET. This contains a list of calendar years, and a list of MOVES fuel sub-types implied by the RunSpec.

3. DRAFT MOVES2009 then executes the GREET GUI described below.

4. When control is returned, DRAFT MOVES2009 receives a tab-separated variable, well-to-pump emission factor result table from the GREET GUI and incorporates these factors into the GREETWellToPump table in a database suitable for use as in input database by subsequent DRAFT MOVES2009 model runs.

5. DRAFT MOVES2009 interacts with the GREET GUI, which in turn invokes the GREET spreadsheet model. DRAFT MOVES2009 does not interact directly with the GREET spreadsheet model, only with the GREET GUI.

10.37.2. GREET and GREET GUI Functionality

The design of the interface between DRAFT MOVES2009 and GREET is based upon the following functional characteristics of GREET.

1. GREET estimates Well-to-Pump and Vehicle Manufacture/Disposal emissions of Total Energy Consumption, Petroleum-based Energy Consumption, Fossil Fuel-based Energy Consumption, N2O, and CH4 appropriate to calendar years 1990 thru 2050. (Internally to GREET, estimates are actually produced by interpolating between estimates applicable to five year periods and estimates beyond 2020 are based on many of the same parameter values as those for 2020.)
2. When the two models are used together, the United States (or the user modeling domain) is modeled as a single geographic region.
3. Where GREET has multiple Pathway Options for a fuel, it is generally possible for the user to supply input parameters for each pathway (with associated percentages which sum to unity), and these Pathway Options and percentages may vary by calendar year.
4. Only GREET parameters accessible from a version of the GREET GUI may be altered. To alter more deeply embedded GREET parameters the user would have to manually modify the underlying GREET spreadsheet before running MOVES.
5. GREET estimates well-to-pump emissions relative to consumption of the following fuels (in DRAFT MOVES2009 these are referred to as fuel subtypes):

   Conventional Gasoline

   Reformulated Gasoline

   Gasohol (E10)

   Conventional Highway Diesel Fuel

   (GREET has the logic built in to use its high sulfur version of this fuel prior to 2006, its low sulfur version after 2006 and a blend for 2006).

   Biodiesel

   FT Diesel

   CNG

   LPG

   Ethanol

   Methanol

   Gaseous H2

   Liquid H2

Electricity

6. The GREET GUI, while not itself written in Java, can be run from a Java program, i.e. DRAFT MOVES2009. It accepts command line parameters which specify:

   - an XML input file name
   - an output file name (for well-to-pump factors and vehicle manufacture/disposal factors)
   - an error file name (used to return any GREET error messages to MOVES).

7. Execution of the GREET GUI normally results in execution of the version of the GREET spreadsheet, but may return to MOVES without executing it if the user desires. Upon return to DRAFT MOVES2009, status information is available as to whether the GREET spreadsheet was run or not and whether execution was successful or an error occurred. Error messages are returned to MOVES in a separate file.

8. The GREET GUI accepts an XML-formatted list of DRAFT MOVES2009 Calendar YearIDs, determines what five-year GREET periods are needed to estimate emissions for all years listed, and executes the GREET spreadsheet for each such period, and interpolates between these results as needed to produce results for the calendar years listed. GREET also accepts an XML-formatted list of fuel types (MOVES fuelSubtypes). GREET results are limited to these fuels.

9. The GREET GUI consolidates the results of these GREET spreadsheet runs into a single tab-delimited table of well-to-pump emission factors and (eventually) a single tab-delimited table of vehicle manufacture/disposal emission factors for use by DRAFT MOVES2009. The table of well-to-pump emission factors is described in the next section.

10.37.3. GREET Well-to-Pump Emission Factor Result Table:

This table is a tab-separated variable ASCII text file and contains the following fields:

a. pollutantID (an Integer from the set of values used in DRAFT MOVES2009)

b. fuelSubtypeID (an Integer from the set of values used in DRAFT MOVES2009)

c. yearID (an Integer identifying the calendar year to which the factor applies)

d. emission rate (a floating point number, expressed in the appropriate units)

Fields a, b, and c, together form the primary key of this table. Field d is its only non-key field. The fields pollutantID, fuelSubtypeID and yearID are as defined in the MOVESDefault database schema.

10.38 Future Emission Rate Creator (FERC)

The Future Emission Rate Creator (FERC) allows MOVES users to alter the energy and emission rates for advanced technology vehicles for model years 2001 and later, and for all energy and emission rates (conventional and advanced technology) for 2011 and later. The FERC works with user-supplied ratios which express the relative benefit of advanced technologies versus conventional technology, by operating mode, to generate advanced technology rates. The FERC is an "external control strategy", meaning it requires work outside of MOVES with MySQL databases and is executed from the "Pre-Processing" menu of the MOVES GUI, rather than being executed as part of the model run itself.

Input data: The FERC requires two input tables, one to create "short term" future rates and another to create "long term" future rates. These are stored in ASCII text, comma separated variable (CSV) format. To generate an alternative set of future emission rates the user must alter these tables directly (outside of MOVES) before running the FERC. The user names these tables as desired and selects them from the MOVES GUI. One set of example input tables is supplied with the demonstration version of DRAFT MOVES2009 .

The two tables contain identical fields, except that the opModeID field is present only in the "short term" table. The first record in each file contains column names to facilitate human readability and is ignored by the calculations. The FERC input table fields are shown in Table 10-9.

Table 10-9: FERC Input Table Fields

Field	Description
polProcessID	Pollutant / Process ID 501 = CH4 running 502 = CH4 start 601 = N2O running 602 = N2O start 9101 = total energy running 9102 = total energy start 9190 = total energy start
opModeID (present in short	Operating Mode ID

term table only)	0-36 = running VSP/speed modes (total energy only) 200 = start mode 300 = running mode (CH4 and N2O)
targetFuelTypeID	Type of fuel: 1 = Gasoline 2 = Diesel 3 = CNG 4 = LPG 5 = E85 6 = M85 7 = Gaseous Hydrogen 8 = Liquid Hydrogen 9 = Electricity
targetEngTechID	Engine Technology ID 1 = Conventional Internal Combustion (CIC) 2 = Advanced Internal Combustion (AIC) 11 = Moderate Hybrid CIC 12 = Full Hybrid CIC 20 = Hybrid AIC (used only for Hydrogen IC hybrid) 21 = Moderate Hybrid AIC 22 = Full Hybrid AIC 30 = Electric 40 = Fuel Cell 50 = Hybrid Fuel Cell
targetModelYearGroupID	Model Year Group ID 20012010 = 2001 thru 2010 20112020 = 2011 thru 2020 20212050 = 2021 thru 2050
fuelTypeID	Base fuel type ID (same as targetFuelTypeID)
engTechID	Base engine technology ID (same as targetEngTech ID)
modelYearGroupID	Base model year group ID (same as targetModelYearGroupID)
fuelEngAdjust	Fuel / Engine Adjustment, i.e. the relative benefit of the target fuel / engine technology versus the base fuel/engine technology. A value of 1 = no benefit, a value of 0.5 = 50% improvement
dataSourceID	5001 thru 5004 = FERC short term 6000 = FERC long term

These fields fall into three functional groups:

1. fields describing the emission rates produced by the adjustment
    a. polProcessID
    b. opModeID
    c. targetFuelTypeID
    d. targetEngTechID
    e. targetModelYearGroupID
    f. dataSourceID

2. fields describing the base technology emission rates to which the adjustment is applied
    a. polProcessID
    b. opModeID
    c. fuelTypeID
    d. engTechID
    e. modelYearGroupID

3. the adjustment itself
    a. fuelEngAdjust

Of these fields, the user would generally only need to alter values of fuelEngAdjust, which are the ratios which define the benefit of the target technology relative to the base technology. Note that the values of polProcessID and opModeID contained in the base records are carried into the future emission rates produced, as are the values of regClassID, engSizeID and weightClassID implicit in the base record sourceBinIDs.

The short term table contains the information necessary to produce alternative fuel and advanced vehicle technology rates for model years 2001 through 2010 (which are treated as one model year group). The long term table contains the information necessary to produce conventional and advanced vehicle technology rates for model year groups 2011 and later, split into two model year groups: 2011 - 2020 and 2021 - 2050. The main difference between the tables is that the short term table uses as base rates the gasoline and diesel rates for conventional technology in the 2001 through 2010 model year group, while the long term table uses the 2001 – 2010 rates as a base the 2001-2010 rates for each technology (i.e. those generated by the short term table). The long term table enables the user to model technology evolution over time, within each technology.

Output produced:

The FERC produces a MySQL database suitable for use as a MOVES user input database. This database contains two tables: EmissionRate and SourceBin. The name of this database is specified by the user.

Calculations performed:

Short term future emission rates are created by applying fuelEngAdjust as a multiplicative adjustment factor to the meanBaseRate of all non-motorcycle EmissionRate table records in the MOVESDefault database with dataSourceIDs less than 5000. For each such record with matching values of polProcessID, opModeID, fuelTypeID, engTechID, and modelYearGroupID a short term future emission rate record is generated having the targetFuelTypeID, targetEngTechID, targetModelYearGroupID and new dataSourceID (while retaining the prior values of polProcessID, opModeID, regClassID, engSizeID, and weightClassID).

Long term future emission rates are created by applying fuelEngAdjust as a multiplicative adjustment factor to the meanBaseRate of all EmissionRate table records in the MOVESDefault database with dataSourceIDs less than 5000 (including those for Motorcycles), along with the short term future emission rates produced by the first step. For each such record with matching values of polProcessID, fuelTypeID, engTechID, and modelYearGroupID a long term future emission rate record is created having the targetFuelTypeID, targetEngTechID, targetModelYearGroupID, and new dataSourceID (while retaining the prior values of polprocessID, opmodeID, regClassID, engSizeID, and weightClassID).

For this calculation to operate correctly the MOVESDefault.SourceBin table must contain a record for each sourceBinID present in the EmissionRate table having a dataSourceID less than 5000. The MOVESDefault database distributed with DRAFT MOVES2009 satisfies this condition.

10.39 I/M Coverage Table Editor

The IMCoverage Table editor provides a Graphical User Interface (GUI) to easily display, edit, and print reports of the IMCoverage table contents. It uses the loaded run specification to determine which records are of interest. As regards its input, it constructs and works to the extent possible only with the IMCoverage Table contents that would be used if that run specification were executed. As regards its output, rather than actually changing the IMCoverage table in MOVESDefault, it produces an IMCoverage table in a User Input Database. Because it operates as a user graphical user interface, further documentation can be found in the *DRAFT MOVES2009 User Guide*.

10.40 Estimating the Uncertainty of MOVES Results

Uncertainty in the model results is introduced by the model theory, the mathematical formulation of the model, and the data used to calibrate the model. The first two sources of uncertainty are not amenable to quantification, so the uncertainty estimates in MOVES focus on the data used to calibrate the model. However, even with this narrower focus, the challenges are significant. MOVES uses data for dozens of variables covering most aspects of mobile source emission generation: the vehicle fleet, vehicle activity patterns, emission rates, fuel properties, geographic location, fuel properties, meteorology, and the presence or absence of vehicle emission control programs. The uncertainty of emission rates is relatively easy to quantify, using standard data analysis techniques, but much of the fleet, activity, fuel, and meteorology data sources have limited information regarding their degree of uncertainty. The primary challenge in including uncertainty estimation in MOVES is therefore to quantify uncertainties for all of the input data used in the model.

DRAFT MOVES2009 includes a basic mechanism which can be used to estimate a portion of the uncertainty of its calculated emissions results, which results from the uncertainty in some particular model inputs, using a Monte Carlo method. A level was added to the model's master looping framework to execute multiple "iterations" of the same run specification on versions of the MOVESExecution database where these input values have been "pseudo-randomly" sampled (as independent samples) from their assumed probability distributions. As currently implemented these probability distributions are assumed to have the form of the normal or Gaussian distribution. Only certain inputs are considered by the program to be random variants, namely those which function as emission rates and emission rate adjustment factors. Specifically these are:

The meanBaseRate field in the EmissionRate, EmissionRateByAge, and SulfateEmissionRate tables.

The tank vapor venting (TVV) terms in the CumTVVCoeffs Table.

The temperature adjustment terms in the TemperatureAdjustment and StartTempAdjustment tables.

The “full AC adjustment factor” in the FullACAdjustment table.

The fuel adjustment factor in the FuelAdjustment table.

Because all records in the MOVESExecution database containing these values are sampled before every iteration and all components of the model obtain their input data from this database, every parameter has a single value for a given iteration that is identical everywhere it is used.

Calculation and Storage of the Input Data Random Samples:

When the Monte Carlo uncertainty estimation feature is invoked the user specifies: 1) the number of iterations to be run (which must be at least two), 2) whether the results of each individual iteration are to be reported or only the last one, 3) whether the input data samples used for each iteration are to be preserved for later reference in the output. Uncertainty estimation may not be invoked in conjunction with either geographic or time period data preaggregation.

When uncertainty estimation is being performed, prior to every iteration each data value in the MOVESExecution database which is considered to be uncertain is replaced with an independent “pseudo-random” sample from the normal or Gaussian distribution whose mean value is the original database point value and whose standard deviation is calculated from this mean value and an associated CV field value. (The coefficient of variation, or CV, is the standard deviation divided by mean.) The MOVESDefault database includes a Coefficient of Variation (CV) field for the fields listed above which the model uses at the start of each iteration to generate a sample value for the field. (The database also includes additional CV fields which are not currently used.)

If the CV field contains a zero or NULL value then the standard deviation is considered to be 0.0.

Specifically, if Z is a pseudo-random sample from the normal distribution with mean 0.0 and unit standard deviation, then:

sampledMeanValue[i] =
originalMeanValue * (1.0 + CV * Z[i])

In the unlikely, but not impossible, event that the sampledMeanValue[i]<0, MOVES sets the sampledMeanValue[i]=0. (In future this may not be done for every value subject to uncertainty.)

MOVES uses the java.util.Random class to generate the Z values.

None of the “random variates” currently implemented involves components of a distribution which sum to unity, but when this mechanism is expanded to include such elements, each term of the distribution will first be generated individually. Then the resulting set of variates will be normalized so that the total sums to one. This procedure will result in the terms of the distribution being negatively correlated, as they should be: if one term is especially large, others must be correspondingly small, and vice versa.

After versions of these tables to be used for a particular iteration have been produced, and if requested by the run specification, their records are copied into corresponding tables in the MOVES output database. These tables added to the output database have the same structure as the original MOVESExecution database tables (e.g. EmissionRate, EmissionRateByAge, etc.) with the addition of fields identifying model run and iteration (MOVESRunID and iterationID).

Calculation and Output of Uncertainty Statistics:

If uncertainty results are being calculated, and if the run specification calls for the results of each iteration to be reported, then two statistics are calculated and reported in each MOVESOutput and MOVESActivityOutput table record.

The mean values of emissionQuant and (if produced) distance, based on the iterations performed so far are reported in the emissionQuantMean and distanceMean fields.

The standard deviations of emissionQuant and (if produced) distance, based on the iterations performed so far are reported in the emissionQuantSigma and distanceSigma fields.

In order to calculate these statistics, the count (which equals the iterationNo), sum, and sum of the squares of emissionQuant and distance are accumulated in a running sum and saved between runs. The full results of each iteration do not need to be saved to calculate these statistics (although the user has the option to save the results for diagnostic and further analytical purposes). These statistics are calculated and saved at the level of detail, and in the engineering units, of the output table. Listed below are the formulas for emissionQuantMean and emissionQuantSigma. Similar formulas would apply to any other variable. In these formulas, the x_is are the results of each iteration, and n is the number of iterations.

$$emissionQuantMean = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$Var = \frac{\sum_{i=1}^{n} x_i^2 - n * (emissionQuantMean)^2}{n-1}$$

$Sigma = \sqrt{Var}$ (the positive square root)

If uncertainty results are being calculated, and if the run specification does not call for the results of each iteration to be reported, then only the records for the last iteration are reported.

Confidence intervals and other statistics can be generated, outside the model, from these results. Dynamic sensitivity can also be calculated from this model output by regressing model inputs against model outputs. Inputs with the highest (in absolute value) regression coefficients will be those for which the greatest improvement will result from decreasing their uncertainty.

10.41. Retrofit Strategy

The MOVES Retrofit Strategy implements a control strategy for on-road vehicle retrofit modeling. It allows the user to select a portion of the overall fleet, and reduce its emission rates by a selected percentage. This section of the SDRM contains a definition sub-section that describes the major retrofit variables, and discusses the retrofit conditional rules. A subsequent section describes the retrofit algorithm.

10.41.1 Definition of Variables

Initial Calendar Year of Retrofit Implementation - The Initial Calendar Year of the Retrofit Implementation is the first calendar year that a retrofit program is administered. A valid Initial Calendar Year input must be equal to or less than the Final Calendar Year of Retrofit Implementation. Initial Retrofit Calendar Year entries that are greater than the MOVES evaluation calendar year are to be ignored. All months within a calendar year are affected equally by the retrofit.

Final Calendar Year of Retrofit Implementation - The Final Calendar Year of the Retrofit Implementation is the last calendar year that a retrofit program is administered. Final Calendar Year input must be equal to or greater than Initial Calendar Year of Retrofit Implementation.

Initial Model Year that will be Retrofit - The Initial Model Year that will be retrofit is the first model year of coverage for a particular vehicle class / pollutant combination. Valid entries for initial model year must meet the following requirement:

Initial Model Year >= Initial Calendar Year - 30

Also, the Initial Model Year cannot be greater than the Final Model Year that will be Retrofit.

Final Model Year that will be Retrofit - The Final Model Year that will be retrofit is the last model year of coverage for a particular vehicle class / pollutant combination. No retrofit shall be performed on Final Model Year inputs which are greater than the Evaluation Calendar Year. Also, the Final Model Year input cannot be less than the Initial Model Year that will be Retrofit.

Percentage of the Fleet Retrofit per Year - The Percentage of the Fleet Retrofit per Year represents the **percentage of VMT** of a particular fleet of a particular vehicle class, retrofit calendar year group, model year group and pollutant combination that is to be rebuilt in a given calendar year. Only values greater than zero and less than or equal to 100.0% are valid. MOVES also checks to insure that the product of the number of calendar years of retrofit coverage (Final Calendar Year of Retrofit Implementation - Initial Calendar Year of Retrofit Implementation) times the Percentage of the Fleet Retrofit per Year does not exceed 100%. For example, a retrofit simulation is flagged as invalid if it had a retrofit program start in calendar year 2005, a program end in calendar year 2008, and a yearly Fleet Retrofit Percentage of 50 percent (3 times 50% > 100%).

Percentage Effectiveness of the Retrofit - The Percentage Effectiveness of the Retrofit is the percent emission reduction achieved by a retrofit. It is computed from a non retrofit emission baseline. The file input structure allows the user to enter a retrofit effectiveness value for a particular vehicle class, retrofit calendar year group, model year group and pollutant combination. All values up to 100% are valid. A negative value is permitted because it implies an emission increase as a result of retrofit which sometimes occurs. A value greater than 100% is not permitted because it implies negative emissions will be generated.

10.41.2 Retrofit Algorithm

The general form of the retrofit algorithm is shown below. Retrofit Emis is the emission rate after a retrofit. Base Emis is the emission rate without retrofit.

Retrofit Emis = [(%Retrofit1*(1-Retrofit Effectiveness1)) + (%Retrofit2*(1-Retrofit Effectiveness2)) … + … (%Retrofit (i) *(1-Retrofit Effectiveness(i))) + (1- %Retrofit1 -%Retrofit2 … - … %Retrofit (i))] * Base Emis

Where

%Retrofit1 + %Retrofit2 … + %Retrofit(i) <= 100%

i < 29 in %Retrofit(i)

11. DRAFT MOVES2009 Input and Default Databases

The principal input data for a MOVES model run is normally obtained from the MySQL database named MOVESDefault. A version of this database is included in each distribution of MOVES. The user may change this database if desired. This is not normally done directly, however, unless records need to be deleted, or a fundamental change to the scope of the model is involved, because MOVES has a more convenient mechanism for the user to supply additional or alternative input data. The MOVES GUI program and MOVES run specifications allow the user to specify one or more MySQL databases whose table records are to be added to or to replace data table records in MOVESDefault during model execution. The simplicity and generality of this scheme is that these MOVES input databases have exactly the same table structure as MOVESDefault (or any portion of it), and may be used to replace all input data items. A limitation of this scheme, however, is that the user is responsible for the accuracy, completeness and consistency of the database that results from this process. This is not always a simple endeavor. EPA envisions that “data importer” programs will be written to help prepare MOVES input databases and support the principal specialized input use cases as they arise. It is also envisioned that organizations outside EPA will produce data importers for MOVES. DRAFT MOVES2009 does not include any data importers, strictly speaking, but its future emission rate creator (FERC) is like a data importer in that it creates EmissionRate table records from externally supplied information. The I/M Table Coverage Editor in DRAFT MOVES2009 is also somewhat like a data importer in that it can be used to create IMCoverage table records, in this case the records are produced from user input to a GUI.

Since the MOVESDefault database and MOVES user input databases have the same table structure, this structure will be referred to in the remainder of this section as simply the “MOVES Database.” The MOVES Database is a relational database which means, among other things, that it is made up entirely of tables, and that every record within a given table has the same set of data items or “fields.” The MOVES database has a naming convention that table names begin with a capital letter and field names begin with a small letter (unless their name starts with an acronym).

The MOVES database structure is highly “normalized” which means that data is contained in many separate tables, several of which usually need to be joined together to satisfy an information requirement.

11.1. Use of Data Types

The overall approach to the use of MySQL column types in the MOVES database is to use integer type columns (2-byte integers where possible, 4-byte integers otherwise) for all key identifying fields (stateID, countyID, hourID, sourceTypeID, etc), and to use normal-precision floating point columns for numeric information. Since MySQL has no boolean (logical true/false) column type, the MOVES database uses columns of character type with length 1 for data items that have a yes/no or true/false nature. Such columns are populated with “Y” to represent “yes” or “true,” and “N” to represent “no” or “false.”

11.2. Functional Types of Tables:

While the MOVES Database consists entirely of tables, these tables serve several different purposes. Some tables function merely to establish value lists for some of the fundamental entities in the database. Examples of such “category value list” tables include State, Year, and DayOfAnyWeek.

A few tables represent “Associations” between database entities. These usually have “Assoc” as the last part of their name. An example of an association type table is PollutantProcessAssoc, which contains information as to which pollutants are emitted by which emission processes. Since this is a many-to-many relationship (i.e. several pollutants are generally produced by each emission process and a pollutant can be produced by several emission processes), an “Association type” table is used to store the valid combinations.

The most common kind of tables in the MOVES database store the substantive subject matter information, and in this document are termed “information” tables. The EmissionRate table, for example, stores emission rates for some of the emission processes in MOVES.

Some “information” tables store data “distributions,” i.e. sets of fractions which add to unity. The MOVES database has a naming convention that such tables have the word “Fraction” or “Distribution” as the last part of their name, and the field containing such fractions has the word “Fraction” as the latter part of its name. An example of this is the “HourVMTFraction” table whose “hourVMTFraction” field stores information as to what fraction of certain VMT occurs during each of the hours of the day. Another example is the “RoadTypeDistribution” table whose “roadTypeVMTFraction” field stores information as to what fraction of certain VMT occurs on each RoadType.

A kind of “information” table which merits special consideration consists of those which are written by MOVES Generators and which MOVES EmissionCalculators, (which can be considered to implement the MOVES Core Model), use as their principal inputs. These are called “core model input tables” or CMITs. The reason CMITs are important to the MOVES user is that they are alternative points for data entry to the model. Generally, the user has a choice as to whether to supply input data to a Generator and have the Generator populate its CMIT tables during model execution, or to place input data directly into the CMIT, (in which case the Generators are programmed not to modify it). A combination of the two approaches may also be used. Most CMITs are information type tables, but there is one notable exception to this, namely the SourceBin table, which can be considered a category or an association type table.

These various kinds of tables are not always completely distinct. The SourceUseType table for example functions as a “category” type table in that it defines the set of sourceTypeIDs available for use in the database, but it also functions as an “information” table, since it contains several subject matter information fields, e.g. sourceMass.

11.3. Database Tables and Their Use

This section briefly describes the purpose of each MOVES database table, and classifies each table in terms of the categories discussed in the previous section. For “Information Tables” it also lists any application-level components of the model which write the table’s records, and which use the table’s data elements. This section can therefore be used to trace the flow of data in the central portion of the model at the table level. The discussion of tables in this document is relatively brief, and is only intended to give a general idea of how each table is used and its most significant characteristics.

The following abbreviations are used in the table to identify the kinds of tables:

CVL (Category Value List)
ASSOC (Association Table)
CMIT (Core Model Input Table)
DIST (Distribution Table)
INF (Information Table which is not also a CMIT or a Distribution Table)

The following abbreviations are used to identify the components of MOVES which read and write the tables: (All tables are written into the MOVESExecution database by the InputDataManager, which along with data preaggregation, is not shown here. Usage by the GUI or other framework components is likewise not shown.)

LTLP (Lookup Table Link Producer)
TAG (Total Activity Generator, includes version for Mesoscale Lookup)

OMDG (Operating Mode Distribution Generator, for running and braking, includes version for Mesoscale Lookup)

SBDG (Source Bin Distribution Generator)

METG (Meterology Generator)

StartOMDG (Start process operating mode distribution generator)

TTG (Tank Temperature Generator)

EvapOMDG (Operating mode distribution generator for the evaporative emission processes)

TFG (Tank Fuel Generator)

AVFT (Alternative Vehicle Fuels and Technologies Strategy)

EC (EmissionCalculators for the Exhaust, TireWear and Brake Wear emission processes which are not "chained" to other calculators, includes Distance Calculator.)

ChainEC (Emission calculator which are "chained" to other calculators)

EvapEC (EmissionCalculators for the Evaporative emission processes)

Table 11.1 MOVES Input Database Tables – Use by Software Components

Table Name	**Type**	**Function /Description**	**Writers**	**Readers** **(as DIST, CMIT or INF)**
AgeCategory	CVL	Defines the valid age categories of SourceUseTypes. ageID=0 represents new vehicles; ageID=30 represents vehicles 30 years old or older. modelyearID=calendarYearID-ageID		
AgeGroup	CVL	Defines the valid age groups, a single set of which is used for all pollutant-processes.		
AverageTankGasoline	CMIT	The gasoline in the fuel tanks of gasoline-fueled vehicles. Allows modeling of mixtures of fuel formulations from the fuel supply.	TFG	EvapEC
AverageTankTemperature	CMIT	Temperature of the fuel in the tanks of vehicles.	TTG	EvapEC
AvgSpeedBin	CVL INF	Defines the average speed bins.		LTLP TAG OMDG
AvgSpeedDistribution	DIST	Distribution of time spent in		TAG

		average speed bins, by source type, roadtype, hour and day.		OMDG
ColdSoakInitialHourFraction	CMIT DIST	The fraction of cold soaking in each hour-day that began in each hour, by source type, zone and month.	TTG	EvapEC
ColdSoakTankTemperature	CMIT	Temperature of the fuel in tanks of cold-soaking vehicles, by zone, month, and hour	TTG	TTG EvapEC
County	CVL INF	Establishes the set of counties. Counties belong to states, but their IDs, based on FIPS state and county codes, are globally unique.		METG EC EvapEC
CountyYear	ASSOC	Associates years with counties		
CumTVVCoeffs	INF	Terms used to calculate tank vapor vented from tank vapor generated. Somewhat analogous to emission rates by age for TVV process. Stored by regClass, modelYearGroup, and ageGroup.		EvapEC
DataSource	INF	Metadata for emission rate records.		
DayOfAnyWeek	CVL	Establishes set of dayIDs which identify a <u>kind</u> of day of the week.		
DayVMTFraction	DIST	Distributes VMT for a source type in a month on a roadtype to the kinds of days of the week.		TAG
DriveSchedule	CVL INF	Defines the set of driving schedules or patterns. Each schedule has an average speed information item.		OMDG
DriveScheduleAssoc	ASSOC INF	Associates a driving schedule with a combination of source type and road type. Also indicates whether schedule is a ramp schedule.		OMDG
DriveScheduleSecond	INF	Records contain the speed for one second of a driving schedule. Schedules start at second = 0. Time gaps are allowed. Such gaps divide schedule into “snippets”.		OMDG
EmissionProcess	CVL	Establish the set of emission processes.		
EmissionRate	INF	Contains emission rates for most pollutant-processes which do not depend upon vehicle age and which are not calculated from other pollutant-processes. Rates depend upon polprocessID, opModeID, and		EC

		sourceBinID		
EmissionRateByAge	INF	Contains emission rates for most pollutant-processes which depend upon vehicle age and which are not calculated from other pollutant-processes. Rates depend upon polprocessID, opModeID, sourceBinID, and ageID		EC EvapEC
EngineSize	CVL	Establishes the set of engSizeID values.		
EngineTech	CVL	Establishes the set of engTechID values.		
ExtendedIdleHours	CMIT	Stores total activity for extended idling process. Geographic unit is the zone.	TAG	EC
FuelAdjustment	INF	Stores multiplicative emission result adjustment factors to account for fuel effects by polProcessID, sourceTypeID, fuel model year groups, and fuel formulation.		EC EvapEC
FuelEngFraction	DIST	Distributes sourceType - modelYear combinations to fuelType - engine technology combinations.	AVFT	AVFT SBDG
FuelEngTechAssoc	ASSOC	Established the combinations of fuelType and engTech that are valid for each sourceType.		AVFT (GUI portion)
FuelFormulation	CVL INF	Establishes the set of fuel formulations. Stores their fuel characteristics.		TFG EC ChainEC
FuelModelYearGroup	CVL	Establishes the model year groupings relevant to vehicle fuel effects upon emissions		
FuelSubType	CVL INF	Establishes the set of fuel subtypes. Stores information about them.		EC ChainEC
FuelSupply	DIST	Contains the marketshare of fuel formulations for each county for each month group and fuel year. When present, marketshares sum to unity for each fuel type. When data not present model assumes 100% marketshare of the fuel type's default formulation.		TFG EC ChainEC EvapEC
FuelSupplyYear	CVL	List of valid fuel supply years.		
FuelType	CVL INF	List of valid fuel types.		EC ChainEC EvapEC
FullACAdjustment	INF	Stores "fullACAdjustment" factors by source type, pollutant-process, and operating mode. Missing values mean no		EC

		adjustment.		
GREETManfAndDisposal	INF	Placeholder table for emission processes not yet implemented.		(not used)
GREETWellToPump	INF	Contains emission rates for the Well-to-Pump process by year, pollutant, and fuel subtype.	(May be written by GREET Interface)	ChainEC
Grid	CVL	Placeholder table to reprent grid cells.		(not used)
GridZoneAssoc	ASSOC	Placeholder table to associate grid cells with zones.		(not used)
HourDay	CVL ASSOC	Exists only because of database design considerations. Associates all hours of the day with all kinds of days of the week.		
HourOfAnyDay	CVL	Lists valid hourIDs. hourID=1 represents the hour beginning at midnight.		
HourVMTFraction	DIST	Allocates the VMT for a sourceTypeID-roadTypeID-dayID combination to the hours of the day		TAG
HPMSVtype	CVL	List of valid vehicle types as classified by the Highway Performance Management System		
HPMSVtypeYear	INF	Stores VMT data for base years and VMT growth factors for all years		TAG
IMCoverage	INF	Information about the existence and effectiveness of vehicle I/M programs. Stored by county, year, pollutant-process, fuel type, regulatory class, and range of modelyears. Missing data means no I/M.	(may be modified by IMCoverage table editor)	EC EvapEC
Link	CVL	Establishes the linkID values and relates Links to Counties, Zones, and Roadtypes. Its informational data items are not currently used.	LTLP	
LinkAverageSpeed	INF	Stores average speed information for a Link. Used for Mesoscale Lookup	LTLP	TAG (Mesoscale lookup) OMDG (Mesoscale lookup)
LinkHourVMTFraction	INF	Not yet used, a placeholder table, intended for smaller scale modeling		(not used)
ModelYear	CVL	Simply lists the valid modelYearIDs		
ModelYearGroup	CVL	Lists the modelYearGroupIds. Establishes a short form for		

		embedding these in sourceBinIDs		
MonthGroupHour	INF	Stores terms used to calculate AC activity		EC
MonthGroupOfAnyYear	CVL	Establishes month groups, which currently are individual months. Would allow fuel-related info to be stored more coarsely.		
MonthOfAnyYear	CVL	Establishes the set of monthID values. Relates months to month groups.		
MonthVMTFraction	INF	Allocates annual VMT to months. Allocation is by sourceTypeID.		TAG
OperatingMode	CVL INF	Establishes the opModeIDs.		OMDG StartOMDG
OpModeDistribution	CMIT DIST	Stores operating mode distributions for several operating mode distribution generators	OMDG StartOMDG EvapOMDG	EC EvapEC
OpModePolProcAssoc	ASSOC	Associates operating modes with pollutant-processes.		
Pollutant	CVL INF	Lists the pollutants calculated by the model, along with information item(s) pertaining to them.		ChainedEC
PollutantDisplayGroup	CVL INF	Used by the MOVES GUI in constructing the pollutant-processes screen.		
PollutantProcessAssoc	ASSOC	Establishes the combinations of pollutant and process which the model calculates		
PollutantProcessModelYear	ASSOC	Associates pollutant-processes with modelYearGroupIDs and fuelMYGroupIDs by listing what model year group and fuel model year group each pollutant-process-model year belongs to.		
RegClassFraction	DIST	Distributes elements of FuelEngFraction distributions by Regulatory Class	AVFT	SBDG
RegulatoryClass	CVL	Establishes the set of RegClassIDs.		
RoadType	CVL INF	Establishes the set of roadTypeIDs. Indicates fraction of SHO on each driven on ramps.		OMDG
RoadTypeDistribution	DIST	Distributes VMT, by sourceType, to roadTypes		TAG
SampleVehicleDay	CVL INF	Establishes the set of vehicles sampled for each kind of day. Indicates the source type of each sample vehicle.		TAG TTG

SampleVehicleTrip	INF	Contains data about each trip made by the sample vehicles.		TAG TTG StartOMDG
SCC	CVL	List of valid highway SCCs. Decomposes this composite code into its component parts.		
SCCProcess	CVL	List of process codes embedded in SCCs		
SCCRoadtype	CVL	List of road classifications embedded in SCCs		
SCCRoadTypeDistribution	DIST	Distributes activity on each roadtype in a zone to the SCCRoadtypes		EC EvapEC
SCCVtype	CVL	List of vehicle classifications embedded in SCCs		
SCCVtypeDistribution	DIST	Distribution activity within a sourceTypeID, modelYearID, and fuelTypeID to SCCVtypeIDs		EC EvapEC
SHO	CMIT	Stores source hours operating total activity and distance traveled. Geographically these are stored by Link.		TAG
SHP	CMIT	Stores source hours parked.	TAG	????
SizeWeightFraction	DIST	Distributions elements of FuelEngFraction distributions by engine size and vehicle weight.	AVFT	SBDG
SoakActivityFraction	CMIT DIST	Distributes hours of parking into hot soaking and cold soaking	TTG	EvapOMDG
SourceBin	CMIT CVL	Lists the sourceBins that used in SourceBinDistributions. Forms an association among the six source bin discriminating fields, indicating which combinations are used. The fact that source bins can be created dynamically is one of the most complicated aspects of the model.	SBDG	EC EvapEC
SourceBinDistribution	CMIT DIST	Stores source bin distributions, which allocate total activity to source bins.	SBDG	EC EvapEC
SourceHours	CMIT	Store source hours (all hours in which the source type exists, equal to SHO plus SHP) total activity. Geographicly these are stored by Link.	TAG	EvapOMDG EvapEC
SourceTypeAge	INF	Stores information items which depend only upon sourceTypeID and ageID		TAG EC
SourceTypeAgeDistribution	DIST	Allocates source type population in each year to vehicle ageIDs		TAG

SourceTypeHour	INF	Stores activity information pertinent to a sourceTypeID during each hourDayID, currently idleSHOFactor.		TAG
SourceTypeModelYear	CVL INF	Establishes combined IDs for combinations of sourceTypeID and modelYearID. Stores information that depends only on these two factors, currently ACPenetrationFraction.		EC
SourceTypeModelYearGroup	INF ASSOC	Associates TankTemperatureGroups with combinations of sourceTypeID and modelYearGroupID		TTG EvapEC
SourceTypePolProcess	ASSOC INF	Establishes which combinations of sourceTypeID and polProcessID require source bin distributions. For such combinatations indicates which source bin discriminators are to be considered in the source bin distributions.		SBDG
SourceTypeYear	INF	Store vehicle population for base year, sales growth and migration rate information for other years		TAG
SourceUseType	CVL INF	Establishes the set of sourceTypeID values. Relates sourceTypeIDs to HPMS vehicle types. Stores items depending only on sourceTypeID		OMDG
Starts	CMIT	Stores total number of starts activity. Geographically this is stored by Zone.	TAG	EC
StartsPerVehicle	CMIT	Stores number of starts per vehicle for each sourceType and hour on each kind of day	TAG	TAG
StartTempAdjustment	INF	Stores factors used to calculate temperature adjustments for certain pollutant-processes		EC
State	CVL	Defines the set of stateID values used in the database		
SulfateEmissionRate	INF	Stores “rates” used to calculate sulfate particulate emissions based on energy consumption		ChainedEC
TankTemperatureGroup	CVL	Defines the set of tank temperature groups		
TankTemperatureRise	INF	Stores coefficients for the equation used to calculate tank temperature rise.		TTG
TankVaporGenCoeffs	INF	Store coefficients for the equation used to calculate fuel tank vapor generation		EvapEC
TemperatureAdjustment	INF	Stores factors used to calculate		EC

		temperature adjustments for certain pollutant-processes		EvapEC
WeightClass	CVL	Establishes the set of weightID values.		
Year	CVL INF	Establishes the set of calendar years for which emissions may be estimated. Indicates which are base years. Maps years to a smaller set of “fuel years”.		TAG EC ChainedEC EvapEC
Zone	CVL DIST	Establishes the set of zoneIDs. Relates zones to counties. Stores information depending only on zoneID, currently several activity allocation factors.		TAG
ZoneMonthHour	INF	Stores information depending only upon zoneID, monthID, and hourID, currently meteorogy-related items.	METG	METG TTG TFG EC
ZoneRoadType	DIST	Stores information depending only upon zoneID and roadTypeID, currently SHO allocation factors		TAG

11.4. Where to Find More Detailed MOVES Database Documentation

More detailed documentation of the MOVES database is contained in a readme directory within the actual database directory. Assuming that MySQL is installed at the standard location on your hard drive and that MOVES has been installed with the MOVES installation package, this would be C:\mysql\data\MOVESDByyyymmdd\readme where “yyyymmdd is the version date. Within this directory a Microsoft WORD document named “MOVESDB.doc” contains definitions for the tables and fields in the database for which the tablename is not fully explanatory. Also located in this directory are a set of Entity-Relationship diagrams illustrating the database. These take the form of “.pdf” files. The graphical conventions which apply to these diagrams, one of which also appears in chapter 12 of this document, include:

Each rectangle represents a table.

The primary key fields of the table are shown above the line horizontal line inside these rectangles.

Fields designated “FK” are foreign key fields; i.e., they help identify related records in other tables.

Lines connecting the rectangles represent relationships between records in the tables.

A short perpendicular line at the end of relationship indicates that it is possible that one record from that table participates in the relationship.

A small “o” at the end of a relationship line indicates that it is possible that no records from that table participate in the relationship.

A small “v” (sometimes refered to as “crow’s feet”) at the end of a relationship line indicates that it is possible that many records from that table participate in the relationship.

12. DRAFT MOVES2009 Output Databases

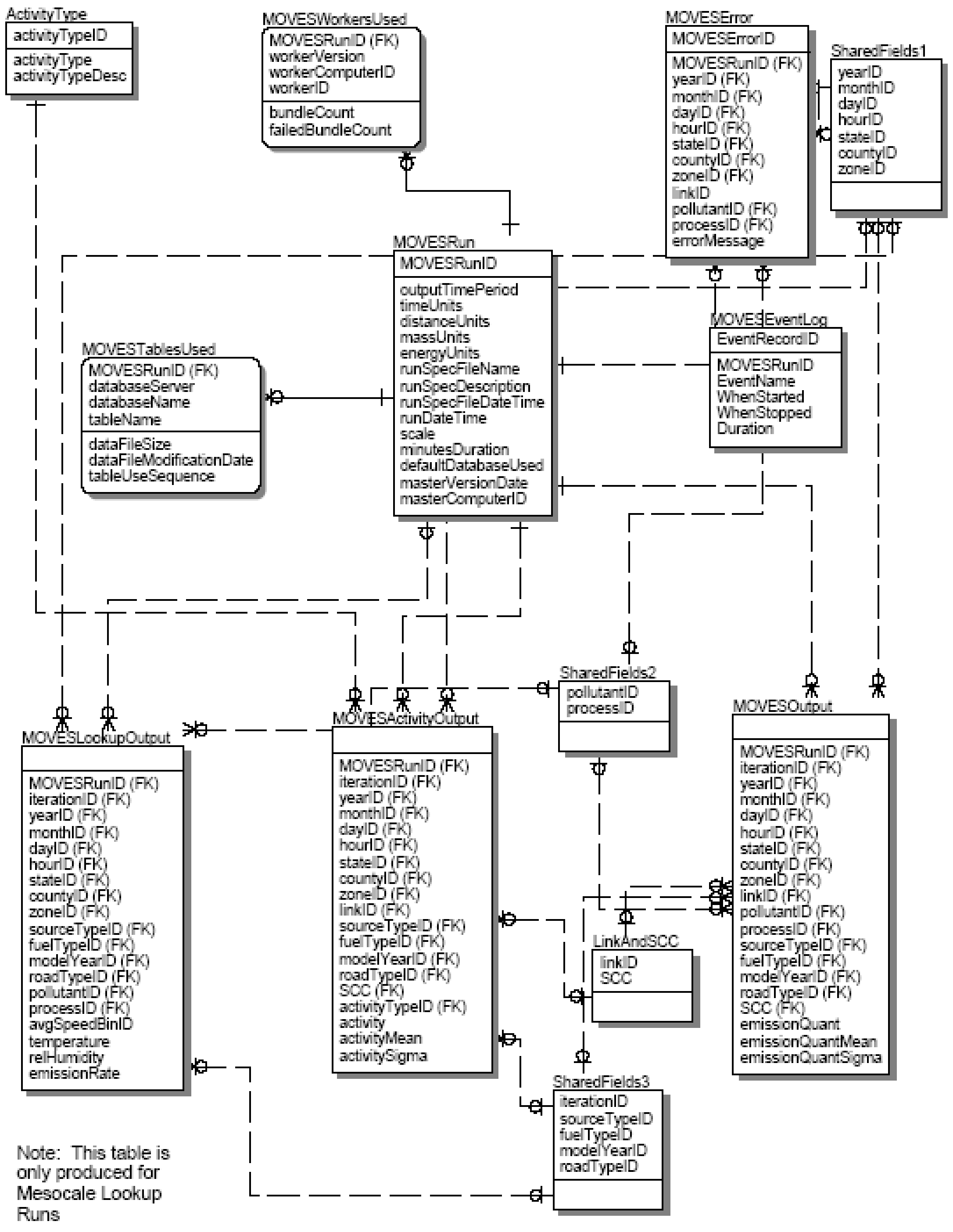
The MOVES GUI/Master program reports the results of each simulation run in the MySQL database named in its run specification. The MOVES GUI may also be used to create an “empty” MOVES Output Database.

There are several software options for working with these databases once they have been created:

> In many situations the most convenient option is to use the “Summary Reporter” component described in Chapter 10 Section 34 of this document and in greater detail in the *DRAFT MOVES2009 User Guide*.   This component, accessible via the “Post-Processing Menu in the MOVES GUI, can be used to aggregate the output various ways, producing screen displays (which can be printed) and tab-separated variable ASCII files which can be imported into other software.
>
> For operations that cannot be done with the Summary Reporter, the most natural method is to use MySQL itself, either via its command line client, which is installed as part of the MySQL installation, or via the MySQL Query Browser which is a graphical client program for MySQL.   The command line client is invoked from a DOS window by entering the “mysql” command. The MySQL Query Browser is included in the DRAFT MOVES2009 Program Suite Distribution and requires a separate installation. A set of MySQL commands, produced by either client program, can be stored and executed again as a MySQL “script”.   Several MySQL scripts are distributed with DRAFT MOVES2009 and are accessible via the “Post-Processor Script Execution” feature documented in Chapter 10.33. Users may add their own scripts to this menu in the MOVES GUI.
>
> MySQL output databases can also be used via Microsoft FoxPro or MicroSoft ACCESS via an ODBC connection. Even Microsoft EXCEL can be used in this fashion if the database is small enough. Instructions as to how to establish these ODBC connections can be found in Appendix B of the *DRAFT MOVES2009 User Guide*.

The results of multiple runs may be stored in the same database and are identified by MOVESRunID. This database consists of six tables as shown in Figure 12-1.

Figure 12-1. Tables in Output Database



12.1. MOVESRun Table

The MOVESRun Table contains information that pertains to the model run as a whole.

The MOVESRunID field contains a number identifying the model run. A single record is written into this table for each run. The first run whose results are stored in this database is assigned run number 1, and the number is incremented for any subsequent runs.

The outputTimePeriod field indicates the time period (Hour, Single Day, Portion of the Week, Month, or Year) for this run. This is a MOVES GUI selection.

The four Units fields indicate the engineering units used to express time, distance, mass, and energy results for this run. The GUI determines the time units automatically based on other selections. The last three are GUI selections.

The runSpecFileName field contains the file name (not including path) of the RunSpecification upon which this run was based. Of course the contents of the runspec may have been changed since the run was performed.

The runSpecDescription field contains the description portion of the run specification.

The runspecFileDateTime field indicates the time and date stamp the run specification file had when the run was executed. If this date-time differs between two runs using the same run specification file name, this may be an indication that some change was made to the run specification between the two runs.

The runDateTime field contains the date and time the run began.

The scale field indicates the modeling scale applicable to the run.

The minutesDuration field indicates the time required to complete the run, expressed in minutes.

The defaultDatabaseUsed field indicates the name of the default database used for the run.

The masterVersionDate field contains the version date of the master program used for the run. This is the same date as appears in a popup window when the program is started or the “Help-About MOVES” menu item is selected.

The masterComputerID field contains the contents of a field in the MOVES Configuration file which is intended to indicate the computer used for the run. When MOVES is first installed this field is not meaningful. To change it a text editor can be used to edit this item in the MOVESConfiguration.txt file.

12.2. MOVESError Table

This table contains a record for each error message generated during runs for which this is the output database. Internally MOVES has several levels of error messages. Currently “warning-level” and “error level” messages are written to this file, and “informational level” messages are not.

The MOVESErrorID field identifies the error message record. It serves as a key field for this table, but its value is not meaningful to the user.

As in the other MOVES Output Database tables, the MOVESRunID field identifies the model run during which the error occurred.

The errorMessage field contains the text of the error or warning level message.

The other fields in this table were intended to identify the portion of the model run that experienced the error, but are not used in DRAFT MOVES2009.

12.3. The MOVESActivityOutput, MOVESOutput, MOVESMesoscalActivityOutput and MOVESMesoscaleOutput Tables

These four tables contain the substantive results of each model run. All four tables are described in this section since they have many fields in common. The MOVESActivityOutput table is used to report activity-related information which is not specific to an emission process or pollutant . The MOVESOutputTable is used to report the pollutant emission results, including energy consumption. The MOVESMesoscaleActivityOutput and MOVESMesoscaleOutput tables match the structure of the MOVESActivityOutput and MOVESOutput tables respectively but are only populated for runs using the Mesoscale Lookup scale. Such runs require special indexing that would be a burden on normal MOVES runs and are separated for performance purposes. The fields of these four tables function as follows:

The MOVESRunID field identifies the model run that produced each output record.

The iterationID field supports estimation of the uncertainty of model results via Monte Carlo statistical approach described in Chapter 10. If uncertainty estimation is not being performed it has a value of 1. If uncertainty estimation is being performed it identifies the model iteration.

The yearID field identifies the calendar year to which the output record pertains. The Year table in the MOVESDefault database defines its legal values. The distributed version of MOVESDefault is intended to support calendar year 1990 and years 1999 thru 2050.

The monthID field identifies the month of the year (if any) to which the output record pertains. Its legal values are defined by the MonthOfAnyYear table in the MOVESDefault database and are 1 thru 12 in the distributed version. A null or zero value indicates that the record pertains to all months of the year that were included in the run specification.

The dayID field identifies the day or portion of the week to which the output record pertains. Its legal values are defined by the DayOfAnyWeek table in the MOVESDefault database. A null or zero value indicates that the record pertains to all portions of the week that were included in the run specification.

The hourID field identifies the hour of the day to which the output record pertains. Its legal values are defined by the HourOfAnyDay table in the MOVESDefault database and are 1 thru 24 in the distributed version (where hour number 1 begins at midnight). A null or zero value indicates that the record pertains to all hours of the day that were included in the run specification.

The stateID field identifies the state to which the output record pertains. Its legal values are defined by the State table in the MOVESDefault database and are based on the FIPS state codes in the distributed version. A null or zero value indicates that the record pertains to all states in the modeling domain (normally the nation) that were included in the run specification.

The countyID field identifies the county to which the output record pertains. Its legal values are defined by the County table in the MOVESDefault database and are based on the FIPS state and FIPS county codes in the distributed version. Note that these county identifications are unique across the entire database since they include the state identification. A null or zero value indicates that the record pertains to all counties in the state, (or, if stateID is also zero or null, the entire modeling domain) that were included in the run specification. When the state level data preaggregation computational shortcuts is taken, as described in Chapter 10.4, values of countyID based only on the FIPS states codes are used to represent entire states.

The zoneID field is based on the countyID in the default database distributed with DRAFT MOVES2009 and provides no additional information in this case. Databases can be constructed, however, wherein each county may have multiple zones.

The linkID field identifies the link (if any) to which the output record pertains. Its legal values are defined by the Link table in the MOVESDefault database and are based on the FIPS state and FIPS county codes and road type classifications in the distributed version. A null or zero value indicates that the record pertains to all links in the county or zone, that were included in the run specification. (In the default database distributed with DRAFT MOVES2009 this corresponds to all road types in the county that were included in the run specification.) This field has a special meaning in Mesoscale Lookup runs as describe in section 10.5.

The sourceTypeID field numerically identifies the source use type (if any) to which the output record pertains. Its legal values are defined by the SourceUseType table in the MOVESDefault database. In the distributed default version these are:

11	Motorcycle
21	Passenger Car
31	Passenger Truck
32	Light Commercial Truck
41	Intercity Bus
42	Transit Bus

43	School Bus
51	Refuse Truck
52	Single Unit Short-haul Truck
53	Single Unit Long-haul Truck
54	Motor Home
61	Combination Short-haul Truck
62	Combination Long-haul Truck

A null value indicates that the output record does not pertain to a particular SourceUseType. (Either the user has selected to report by Source Classification Code (SCC) instead, or not to distinguish the results by any vehicle classification.)

The SCC field identifies the Source Classification Code (if any) to which the output record pertains. Its legal values are defined by the SCC table in the MOVESDefault database. A null value indicates that the output record does not pertain to a particular SCC. (Either the user has selected to report by Source Use type instead, which is recommended, or not to distinguish the results by any vehicle classification.)

The fuelTypeID field numerically identifies the top-level fuel type (if any) to which the output record pertains. A null value indicates that the record pertains to all fuel types. Whether results are to be distinguished by fuel type is a GUI selection and is included in the run specification. The legal values of fuelTypeID are defined by the FuelType table in the MOVESDefault database. In the distributed default version these are:

1	Gasoline
2	Diesel Fuel
3	Compressed Natural Gas (CNG)
4	Liquid Propane Gas (LPG)
5	Ethanol (E85 or E95)
6	Methanol (M85 or M95)
7	Gaseous Hydrogen
8	Liquid Hydrogen
9	Electricity

The modelYearID field identifies the model year (if any) to which the output record pertains. A null value indicates that the record pertains to all model years. Whether results are to be distinguished by model year is a GUI selection and is included in the run specification.

The roadTypeID field identifies the road type (if any) to which the output record pertains. The legal values of this field are defined by the RoadType table in the MOVESDefault database. In the distributed version of DRAFT MOVES2009 there are 4 roadtypes intended to classify actual highways, plus a fifth RoadType representing locations in the zone which are not on the highway network.

1	Off-Network

2	Rural Roadways with Restricted Vehicle Access
3	Rural Roadways with Unrestricted Vehicle Access
4	Urban Roadways with Restricted Vehicle Access
5	Urban Roadways with Unrestricted Vehicle Access

MOVES associates start, extended idle, and well-to-pump emissions with RoadType 1 and running emissions with the four actual roadway classifications. A null value of roadTypeID indicates that the results pertain to all roadway types. Whether to distinguish results by roadTypeID is a GUI selection that is included in the run specification. Producing results by SCC implies that roadtypes are not distinguished which is a change in DRAFT MOVES2009. EPA considers roadtypes 2 thru 5 listed above to represent a set of HPMS roadway classifications, but MOVES itself does not make this assumption.

The pollutantID field numerically identifies the pollutant to which the output record pertains. It is present only in the MOVESOutput table. Results are always distinguished by pollutant. The Pollutant table in the MOVESDefault database defines the legal values of pollutantID. In the distributed default DRAFT MOVES2009 database for the demonstration version they are as follows:

1	Total Gaseous Hydrocarbons
2	Carbon Monoxide (CO)
3	Oxides of Nitrogen (NOx)
5	Methane (CH4)
6	Nitrous Oxide (N20)
90	Atmospheric Carbon Dioxide (CO2)
91	Total Energy Consumption
92	Petroleum Energy Consumption
93	Fossil Energy Consumption
98	CO2 Equivalent
105	Primary PM10 – Sulfate Particulate Matter
111	Primary PM2.5 – Organic Carbon Particulate Matter
112	Primary PM2.5 – Elemental Carbon Particulate Matter
115	Primary PM2.5 - Sulfate Particulate Matter
116	Primary PM2.5 – Brake Wear Particulate Matter
117	Primary PM2.5 – Tire Wear Particulate Matter

The processID field numerically identifies the emission process (if any) to which the output record pertains. It is present only in the MOVESOutputTable. The EmissionProcess table in the MOVESDefault database defines the legal values for processID. In the distributed version these are:

1	Running Exhaust
2	Start Exhaust
90	Extended Idle Exhaust
99	Well-to-Pump

A null value in this field indicates that the result record pertains to all emission processes. Whether to distinguish results by emission process is a GUI selection that is contained in the run specification.

The emissionQuant field is present only in the MOVESOutput and MOVES MESOscaleOutput tables contains the quantity of emissions of the given pollutant as qualified by all the other identifying fields. Engineering units for mass type pollutants may be in terms of kilograms, grams, pounds, or U.S. tons as selected in the GUI, contained in the run specification and output in the MOVESRun table. Engineering units for energy consumption results may be in terms of Joules or millions of BTU's as selected in the GUI, contained in the run specification, and output in the MOVERun table.

The emissionQuantMean field is present only in the MOVESOutput and MOVESMesoscaleOutput tables. It is only used if the uncertainty estimation feature is invoked in the run specification. In "normal" runs it contains a zero value. When uncertainty estimation is being performed it contains the mean value of emissionQuant in iterations up to and including the one represented by this record.

The emissionQuantSigma field is present only in the MOVESOutput and MOVESMesoscaleOutput tables. It is only used if the uncertainty estimation feature is invoked in the run specification. In "normal" runs it contains a zero value. When uncertainty estimation is being performed it contains the variance of the emissionQuant in iterations up to and including the one represented by this record.

The activity field is present only in the MOVESActivityOutput and MOVESMesoscaleActivityOutput tables and contains the distance traveled as qualified by all the other identifying fields. Its engineering units may be miles or kilometers as selected in the GUI, contained in the run specification,and output in the MOVESRun table.

The activityTypeID field is present only in the MOVESActivityOutput and MOVESMesoscaleActivityOutput tables. It references the ActivityType table which defines activityTypeID=1 as distance.

The activityMean field is present only in the MOVESActivityOutput and MOVESMesoscaleActivityOutput tables. It is not used in the demonstration version of Draft MOVES2009.

The activitySigma field is present only in the MOVESActivityOutput and MOVESMesoscaleActivityOutput tables. It is not used in the demonstration version of Draft MOVES2009.

Appendix A. Table of Acronyms

AB	aktiebolog (Swedish)
AC	air conditioning
API	application program interface
AVFT	alternative vehicle fuels and technologies
ASCII	American Standard Code for Information Interchange
BTU	British thermal unit
CH4	methane
CD	compact disc
CMIT	core model input table
CNG	compressed natural gas
CSEC	criteria start emission calculator
CSV	comma-separated variable
DOT	Department of Transportation
ECC	energy consumption calculator
EPA	Environmental Protection Agency
F	Fahrenheit
FERC	future emission rate calculator
FK	foreign key
FIPS	federal information processing standard
GB	gigabyte
GPL	general public license
GHz	gigahertz
GUI	graphical user interface
GNU	GNU’s not UNIX (recursive)
GREET	Greenhouse gases, Regulated Emissions, and Energy uses in Transportation
H2	hydrogen
HC	hydrocarbons
HDT	heavy duty truck
HPMS	Highway Performance Management System
IC	internal combustion
ID	identification
IM	inspection/maintenance
kW	kiloWatt
LDT	light duty truck
LDV	light duty vehicle
LPG	liquified propane gas
LTLP	lookup table link producer
MAR	mileage accumulation rate
MB	megabyte
MOVES	MOtor Vehicle Emissions Simulator
DRAFT MOVES2009	The Highway Vehicle Implementation of MOVES
N2O	nitrous oxide
NMIM	National Mobile Inventory Model

OBD	on board diagnostics
ODBC	open database connectivity
OMDG	operating mode distribution generator
OTAQ	Office of Transportation and Air Quality
PTW	pump-to-wheel
RAM	random access memory
RTC	runs to completion
SBDG	source bin distribution generator
SCC	source classification code
SDK	software development kit
SDRM	Software Design and Reference Manual
SHO	source hours operating
SHP	source hours parked
SQL	structured query language
SUV	sport utility vehicle
TAG	total activity generator
US	United States
VMT	vehicle miles traveled
VOC	volatile organic hydrocarbon
VSP	vehicle specific power
WTP	well-to-pump
XML	extended markup language

Appendix B. MOVES Error/Warning Messages

Messages

Adding execution locations for county, [*], failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Adding execution locations for link, [*], failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Adding execution locations for state, [*], failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Adding execution locations for the nation failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Adding execution locations for zone, [*], failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

All data is not yet ready to be exported. [*]

The selected internal control strategy is not ready to export its data. Check the strategy's GUI to resolve any errors before exporting.

An EmissionCalculator encountered an SQL exception while exporting data using:

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

An SQL exception occurred while adding execution locations.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

An SQL exception occurred while building execution indexes.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

An SQL exception occurred while building mesoscale lookup links.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

An SQL exception occurred while building the runspec filter sets.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

An SQL exception occurred while building the runspec filter tables.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Copy table, [*], from source database failed.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not build TTGeMinutes table.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not calculate average speeds
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not calculate Engine Power Distribution.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not calculate operating mode distributions
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not check driveScheduleSecondLink
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from

MOVES.

Could not create hot soak and operating temperature tables.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not create unique vehicle IDs.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not delete Project Total Activity data.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not delete Total Activity data from previous run.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine average tank temperature.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine brackets for Average Speed Bins.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine cold soak fractions.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine cold soak tank temperature.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine Evaporative Emissions Operating Mode Distribution.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine final Operating Mode Distribution.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine fraction of drive schedules in each speed bin.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine fractions of Operating Modes per Drive Schedule

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine hot soak temperatures.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine operating mode for sample vehicle trips.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine operating mode fraction.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine Operating Mode ID distribution.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine soak activity fraction.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine soak times for sample vehicle trips.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine the distribution of drive schedules for non ramp drive cycle.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not determine the distribution of drive schedules for ramp drive cycle.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not do link operating mode setup

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not do TFG sql.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not flag marker trips

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Could not load InternalControlStrategy text

The RunSpec file is corrupt, likely due to a typo.

Could not load InternalControlStrategy XML.

The RunSpec file is corrupt, likely due to a typo.

Could not load runspec XML.

The RunSpec file is corrupt, likely due to a typo.

Could not remove Source Bin Distribution data from previous run.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

DROP TABLE failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Error removing generated data from the execution database.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Exception occurred on 'randomizeTableData' [*] using

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Failed to get monthID from MonthOfAnyYear with:

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Failed to update MOVESRun.minutesDuration

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Get list of years failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

IMGUI countyName==null, countyID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI fuelName==null, fuelTypeID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI inspectFreqName==null, inspectFreq=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI null text for display line

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI pollutantName==null, polProcessID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI processName==null, polProcessID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI regClassName==null, regClassID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI testStandardsName==null, testStandardsID=

The I/M display could not find required human-readable details for an I/M program in the database.

IMGUI testTypeName==null, testTypeID=

The I/M display could not find required human-readable details for an I/M program in the database.

InternalControlStrategy failed to load

The RunSpec file is corrupt, likely due to a typo.

Invalid class name entry

The RunSpec file is corrupt, likely due to a typo.

Invalid DatabaseSelection

The RunSpec file is corrupt, likely due to a typo.

Invalid Generic County.

The RunSpec file is corrupt, likely due to a typo.

Invalid GeographicOutputDetail

The RunSpec file is corrupt, likely due to a typo.

Invalid GeographicSelection.

The RunSpec file is corrupt, likely due to a typo.

Invalid HydrocarbonUnitSystem

The RunSpec file is corrupt, likely due to a typo.

Invalid InputDatabase

The RunSpec file is corrupt, likely due to a typo.

Invalid InternalControlStrategy

The RunSpec file is corrupt, likely due to a typo.

Invalid InternalControlStrategy XML file.

The RunSpec file is corrupt, likely due to a typo.

Invalid ModelDomain

The RunSpec file is corrupt, likely due to a typo.

Invalid ModelScale

The RunSpec file is corrupt, likely due to a typo.

Invalid OffRoadVehicleSCC
The RunSpec file is corrupt, likely due to a typo.

Invalid OffRoadVehicleSelection
The RunSpec file is corrupt, likely due to a typo.

Invalid OnRoadVehicleSelection
The RunSpec file is corrupt, likely due to a typo.

Invalid OutputDatabase
The RunSpec file is corrupt, likely due to a typo.

Invalid OutputEmissionsBreakdownSelection
The RunSpec file is corrupt, likely due to a typo.

Invalid OutputFactors
The RunSpec file is corrupt, likely due to a typo.

Invalid OutputTimeStep
The RunSpec file is corrupt, likely due to a typo.

Invalid PMSize
The RunSpec file is corrupt, likely due to a typo.

Invalid PollutantProcessAssociation
The RunSpec file is corrupt, likely due to a typo.

Invalid RoadType
The RunSpec file is corrupt, likely due to a typo.

Invalid RunSpec XML file.
The RunSpec file is corrupt, likely due to a typo.

Invalid ScaleInputDatabase
The RunSpec file is corrupt, likely due to a typo.

Invalid timing on request for instantiated object [name]
This internal error can occur if user-modified Java code calls MOVESInstantiator.didInstantiate() out of the proper sequence.

Invalid UncertaintyParameter
The RunSpec file is corrupt, likely due to a typo.

Loading a list of runs for the MOVES Error Log failed.
An error occurred while working with a database. The database could

have become unavailable or could have been modified externally from MOVES.

loading counties failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

loading lookup table failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

loading pollutantProcessMap failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

loading reverse lookup table failed

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Missing parameter i=

The RunSpec file is corrupt, likely due to a typo.

Only one object per RunSpec is allowed for this type

For some internal control strategies, the AVFT for instance, it only makes sense to have at most one of them per RunSpec.

Replace into table, [*], from source database failed.

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Road type ID, [*], with name, '[*]', was not found in the default database.

The RunSpec file is corrupt, likely due to a typo.

Run specification must contain pollutant-process, county, year, and vehicle-fuel type selections for an IMCoverage report to be produced.

MOVES could not generate the list of I/M coverage records that apply to the current RunSpec because the RunSpec lacks selections on one or more required screens.

Since only one object per RunSpec is allowed for this type, importing a new one will overwrite this object. Do you really want to do this?

For some internal control strategies, the AVFT for instance, it only makes sense to have at most one of them per RunSpec.

SQL error in FER Adjustment File Loading
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

SQL error in heat index.calculation
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Table model unable to select
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Table name is NULL for the query :
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

The selected file does not contain this type of object, it has not been loaded.
MOVES could not find the requested internal control strategy object within the file you selected.

TTG could not access SoakActivityFractions
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to apply new FuelEngFraction
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to calculate FuelEngFraction
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to calculate model year range needed
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to calculate RegClassFraction

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to calculate SCCVTypeDistribution

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to calculate SizeWeightFraction

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to check for county domain data

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to check missing emission rates:

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to count operating modes for polProcessID = [*].

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to create InternalControlStrategy

While setting up a requested internal control strategy, an error occurred. This is likely a case of a RunSpec that requests an internal control strategy that is not compiled into this machine's MOVES.

Unable to create MOVESLookupOutput table

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to create temporary summary tables

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete EE Operating Mode Distribution data

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete from database

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete Operating Mode Distribution data from a previous run

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete Start Operating Mode Distribution data from a previous run

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete tank fuel data from a previous run

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to delete tank temperature data from a previous run

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to do AVFT diagnostics

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to fill lookup array

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to fill lookup arrays

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to finalize MOVESEventLog

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to find user inputs for ColdSoakInitialHourFraction
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to get database info for Pollutant Process Association where polProcessID =
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to get database key for Pollutant Process Association where Process ID = [*] and Pollutant ID = [*].
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to get SHO count in ECC
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to get single opModeID for polProcessID = [*].
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to learn Create Table statements
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to load ATRatio entries
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to load columns in StateCountyMapGUI using
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to load HC entries

An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to load tables in StateCountyMapGUI using
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to lookup retrofit abbreviation
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to open hot soak cursor
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to open SVTH cursor
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to perform Source Bin Distribution for pollutant/process,[*]/
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to process exported file contents
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to process work file contents
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to query PollutantProcessAssoc
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to save CMITs
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from

MOVES.

Unable to save XML for InternalControlStrategy
The RunSpec file is corrupt, likely due to a typo.

Unable to save XML.
The RunSpec could not be saved. This often happens if the file is already open in a text editor.

Unable to store table tracking details
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to update database
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to update MOVESWorkersUsed
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to write error message to output database.
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to write start time to MOVESEventLog
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Unable to write stop time to MOVESEventLog
An error occurred while working with a database. The database could have become unavailable or could have been modified externally from MOVES.

Warning: [SAX XML parser exception]
The RunSpec file is corrupt, likely due to a typo.