MOVES2014a Software Design Reference Manual

EPA United States Environmental Protection Agency

MOVES2014a Software Design Reference Manual

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

United States Environmental Protection Agency

EPA-420-B-15-096
November 2015

Introduction

Introduction

MOVES is the U.S. Environmental Protection Agency's (EPA) Motor Vehicle Emission Simulator. The purpose of MOVES is to provide an accurate estimate of emissions from cars, trucks and non-highway mobile sources under a wide range of user-defined conditions. This document provides information on MOVES software structure and algorithms.

At its most basic, MOVES is a collection of Java code, MySQL databases, and SQL scripts. At a higher level, it is a collection of interacting models that use curated data, user-supplied data, and each other's outputs to create inventory and emission rate values. This manual presents the high-level information required to understand the structure of the MOVES Java code and the flow of information through MOVES components.

While MOVES2014 can estimate air pollution emissions from nonroad mobile equipment, this manual focuses on the portions of the model devoted to highway vehicles. The chapters describing MOVES components, hardware and software requirements, functional structure, licenses and installation also apply to the nonroad portion of the model. However, the nonroad calculations have their own calculator and share the meteorology generator with highway vehicles. No other generators are used for nonroad.

Readers interested in the details of MOVES calculations may also want to review the "MOVES2014a Module Reference," a detailed HTML guide to each of the MOVES modules and MOVES default database tables.

A Note on Capitalization and Spacing

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). For example, 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 (vehicle miles traveled) or AC (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.

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.

Chapter 1

MOVES Software Components

Introduction

MOVES is written in Java and the MySQL relational database management system, products of Oracle Corporation. It also calls executable programs written in Fortran and the Go programming language. MySQL databases are the location of the principal user inputs, outputs and most of the model's internal working storage. The model includes a default input database covering 3228 counties of the United States which supports model runs for calendar years 1990 and 1999 - 2050.

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, introduced briefly in this chapter. Many of these components are described in more detail in later sections of this document.

MOVES Graphical User Interface (GUI) and Master Program

This is a Java program which manages the overall execution of a model run. The 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 Master Program without interacting with the MOVES GUI. The Master Program uses information in the Run Spec to create the execution database, assign calculations to MOVES workers and compile results into the MOVES output database.

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

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. These databases typically contain region-specific fuels, I/M programs, vehicle populations and activity.

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.

Chapter 2

MOVES Hardware and Software Requirements

Introduction

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. MOVES operates on versions of Windows XP, Vista, Windows 7, 8, and 10, but does not operate on earlier versions of Windows.

Computers used to MOVES should have at least 3GB of RAM (additional memory is recommended). Execution run time performance is a constraint with MOVES, so high speed multi-core processors are highly recommended. For disk storage, the MOVES model takes up approximately 600MB after installation. Additional space is required during runtime as MOVES Worker and Output databases and temporary files are also often voluminous, so at least 5GB of disk space should be available on all machines used to run MOVES. Extensive users of MOVES will want to use the highest performance microcomputer systems available to them. At least 50 GB of disk storage is recommended for machines performing large runs.

Note that while solid state drives (SSD) offer higher data transfer speeds that typical hard drive technology, they should be used with caution. Because of the intense volume of data writing and deleting done by MOVES for temporary files, using MOVES where MySQL and MOVES access data from an SSD may prematurely age the SSD and lead to possible failure. EPA recommends that modelers not use SSD in combination with MOVES, unless they are fully understand the extra wear that MOVES may cause to the SSD.

Please note that administrative privileges are required to install the MOVES model and prerequisite software; however, these privileges are not required to run the model or analyze the outputs.

Prerequisite Software: Java and MySQL

MOVES is primarily written in Java, and as such requires the presence of the Java Runtime Environment (JRE) version 7 or 8. In addition, MOVES uses MySQL to store data and perform calculations. Typical installation configurations utilize a MySQL Server operating on the same machine as MOVES. In this configuration, any computer running the MOVES GUI, MOVES Master, or MOVES Worker also must run a MySQL Server. Both MySQL Server 5.5 and 5.6 are supported.

Due to these requirements, the MOVES installer will check whether or not supported versions of Java and MySQL are present on your computer. If either are not found, the MOVES installer will direct you to the provider's website to download and install the missing program before continuing. The MOVES2014a installer does not include installation packages for these required software.

Though not required to run MOVES, MySQL Workbench and MySQL Connector/ODBC are recommended tools for analyzing MOVES output. More information can be found in the installation chapter of this document.

Since both Java and MySQL are available for a variety of other operating systems, such as Linux, Unix, and iOs among others, it may be possible to port MOVES to such environments. However, EPA does not test or support such configurations.

Worker External Calculator

Calculators within MOVES are expressed using SQL statements. These statements are created and executed by Java classes using a combination of conditional statements and files containing SQL templates. Java code in the master reads the SQL templates and generates text files containing table data and a list of SQL commands to be performed by a worker. A worker populates its database from the text files and executes the SQL commands. The worker extracts table data from output tables and bundles it for return to the master.

Beginning with MOVES2014a, calculators can also include Go-based logic executed on the worker computers. “Go” is an open source programming language developed by Google, Inc. and many contributors from the open source community. It has a streamlined textual style similar to Python and superb support for multithreaded programming. This feature allows a Go-based calculator to do several computations simultaneously, even generating output before completely finished reading its input file.

Go-based calculators contain Java code on the master and a SQL template file on the master, just as a traditional calculator. The SQL generated for a Go-based calculator includes special statements that cause the worker Java code to execute the external calculator program.

All Go-based calculators are compiled into a single executable file. The “ant go” command compiles the code in the \calc\go\src\ directory within the main MOVES installation directory. When used on a 64-bit operating system, the “ant go” command will create three executable files in the \calc\go\ directory: externalcalculatorgo.exe, externalcalculatorgo32.exe, and externalcalculatorgo64.exe. The externalcalculatorgo.exe program is always the default to the operating system, making it 64-bit when compiled on a 64-bit operating system, 32-bit otherwise. Externalcalculatorgo32.exe is a 32-bit program that will run on both 32 and 64-bit systems. Externalcalculatorgo64.exe will only operate on 64-bit systems. By modern standards, 32-bit programs are tightly constrained on their memory usage. When possible, use a 64-bit operating system and the externalcalculatorgo64.exe program as this will increase performance by reducing disk accesses. The "ant go" command compiles under Windows 32-bit, Windows 64-bit, and Linux.

The WorkerConfiguration.txt file controls which external calculator program is used by MOVES. The calculatorApplicationPath entry gives the full directory path and file name to be used, such as:

calculatorApplicationPath =
C:\Users\Public\EPA\MOVES\MOVES2014a\calc\go\externalcalculatorgo32.exe

To support the “ant go” command, MOVES accepts a 32-bit Go compiler installed to the \go32\ directory within the main MOVES installation directory. This can be created by performing a normal 32-bit Go compiler installation then copying its folders and files to MOVES’ directory.

The Go-based external calculator is called by the worker Java code in much the same way the Fortran-based Nonroad program is called by the worker Java code. The worker code first executes SQL that creates a text file of the MOVESWorkerOutput table. Next, the external calculator is run, which in turn reads the file and creates a new output file. Afterwards, the worker Java code executes SQL that loads the new output file back into the MOVESWorkerOutput table.

In MOVES02014a the Go-based external calculator computes:

- Nonroad fuelSubtype splits

- Onroad HC speciation (VOC, etc)

- Nonroad HC speciation

- Nonroad air toxics

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 to which the MOVES master and Worker have access. 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, 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 5 GB of disk space should be available.

Configuring MOVES

All MOVES application components may be installed on a single computer system. MOVES may also be configured so that several computers work together to execute model simulation runs. This can significantly improve the execution time 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 during a MOVES run, all that is necessary is for the computers to have access to the SharedWork directory. When using only a single 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 be satisfied by the computers 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.

Technical details on compiling MOVES are available in an Appendix to this document.

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 at no charge 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 Oracle Corporation 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/gpl.html.

Chapter 3

Installation Overview

Introduction

The MOVES installer is available for download from the EPA website, and is compatible with Windows XP/Vista/7/8/10. It contains the code and default database required to run the model but does not contain Java or MySQL, which are also required for MOVES to run. During MOVES installation, the installer will check whether or not Java and MySQL are present on your computer. If either are not found, the MOVES installer will direct you to the provider's website to download and install the missing program before continuing.

Administrative privileges are required to install the MOVES model and prerequisite software; however, these privileges are not required to run the model or analyze the outputs. If you do not have these privileges, you need to obtain them for the duration of the installation process, or enlist the help of someone who does. Contact your IT department for more information.

Please note that MOVES releases starting with the May 2015 Update to the MOVES2014 October Release (published on 5/5/2015) used new installation software. The original MOVES2014 installer is documented in the December 2014 version of the SDRM.

Installing Java

MOVES requires the Java Runtime Environment (JRE) version 7 or 8 to run. If the installer cannot find a suitable JRE on your computer, it will prompt you to download and install it from the Java website. In the unlikely case that a suitable JRE is present but the MOVES installer cannot find it, you may enter the path to the JRE directory on the "Could Not Find Java" page of the MOVES installer. The installer will not continue unless a valid JRE is identified. Frequently asked questions and additional help are available by clicking the "Java Help" button on all Java-related screens of the MOVES installer.

Multiple versions of the JRE can reside on a computer simultaneously and will not affect MOVES. However, if an update removes or renames the JRE installation that MOVES uses, MOVES will need to be reinstalled or reconfigured to work. To reconfigure, edit setenv.bat and smallenv.bat in the MOVES main installation directory so that the line starting with "set JRE_HOME=" contains the correct path to an acceptable JRE installation.

If you wish to recompile MOVES, you will also need the Java Development Kit (JDK) version 7 or 8. Technical details on compiling MOVES are available in an Appendix to this document. Note that the JDK is not required for simply running MOVES.

Installing MySQL

MOVES also requires MySQL Server 5.5 or 5.6. If the installer cannot find a suitable version of MySQL on your computer, it will prompt you to download and install it from the MySQL website. In the unlikely case that a suitable MySQL Server is present but the MOVES installer cannot find it, you may enter the path to the MySQL executable on the "Could Not Find MySQL" page of the MOVES installer. The installer will not continue until a valid MySQL Server is identified. Frequently asked questions and additional help are available by clicking the "MySQL Help" button on all MySQL-related screens of the MOVES installer.

In addition, the MySQL Server must be configured properly for the MOVES installation to be successful. Briefly, TCP/IP connectivity must be enabled, though you do not need to open firewall port for network access. In addition, MySQL Server must be installed as a Windows Service run as Standard System Account. If the root password for the server is set to something other than `root`, `moves`, `password`, or is left blank, the MOVES installer will prompt you to enter the root password during installation. This password is not saved or recorded by the installer, but it is required to continue. The installer uses root access to create a MOVES-specific MySQL account which is used when running MOVES.

Multiple versions of MySQL Server can reside on a computer simultaneously and will not affect MOVES. However, if the port number for the version of MySQL that MOVES uses changes, MOVES will need to be reinstalled or reconfigured to work. To reconfigure, edit MySQL.txt in the MOVES main installation directory to contain the new port number. If MySQL.txt does not exist, create it using a text editor and save the new port number in it. Note that this file should not have anything else in it other than the port number.

Users are recommended to also install MySQL Workbench. MySQL Workbench is a product of Oracle Corporation and replaces the earlier MySQL Query Browser used with MOVES2010. This product may be used to analyze MOVES output using MySQL commands or to export the data to Microsoft Excel. You may select to install it along with MySQL Server through the MySQL Installer program.

If users prefer to analyze MOVES output with other software, such as Microsoft Access, SAS, R, Python, etc., you should install the MySQL Connector/ODBC, which is also available through the MySQL Installer program.

Installing MOVES

After checking for Java and MySQL and prompting the user to download and install them if needed, the MOVES installer will prompt the user for where MOVES should be installed. It will also ask the user if it should create a Start menu entry for MOVES and/or desktop icons. After the user clicks “Install”, the installer will copy the required files for MOVES to run[1], install the default database, configure MOVES, and perform a short run to test the installation. If errors occur during any of these steps, the installer will notify the user. More information on resolving these issues may be found the Installation Troubleshooting Guide, available from any screen of the MOVES installer.

Uninstalling MOVES

MOVES may be uninstalled from either the MOVES Start menu shortcut or from Add/Remove Programs in Window’s Control Panel. Before taking any actions, the uninstaller will ask if you wish to remove the default database and the MOVES user in addition to

[1] In addition to installing the compiled Java code, MySQL scripts, and required configuration files, the installer also includes the Java source code, other Java extensions (such as JUnit and JFCUnit, which facilitate software testing), libraries (such as Apache’s XMLBeans, which provides access to Microsoft Excel formatted files), and Apache Ant (the tool used to compile the MOVES source code).

removing the MOVES code. Please note that if there are other versions of MOVES present on your computer, they will break if you select to delete the MOVES user.

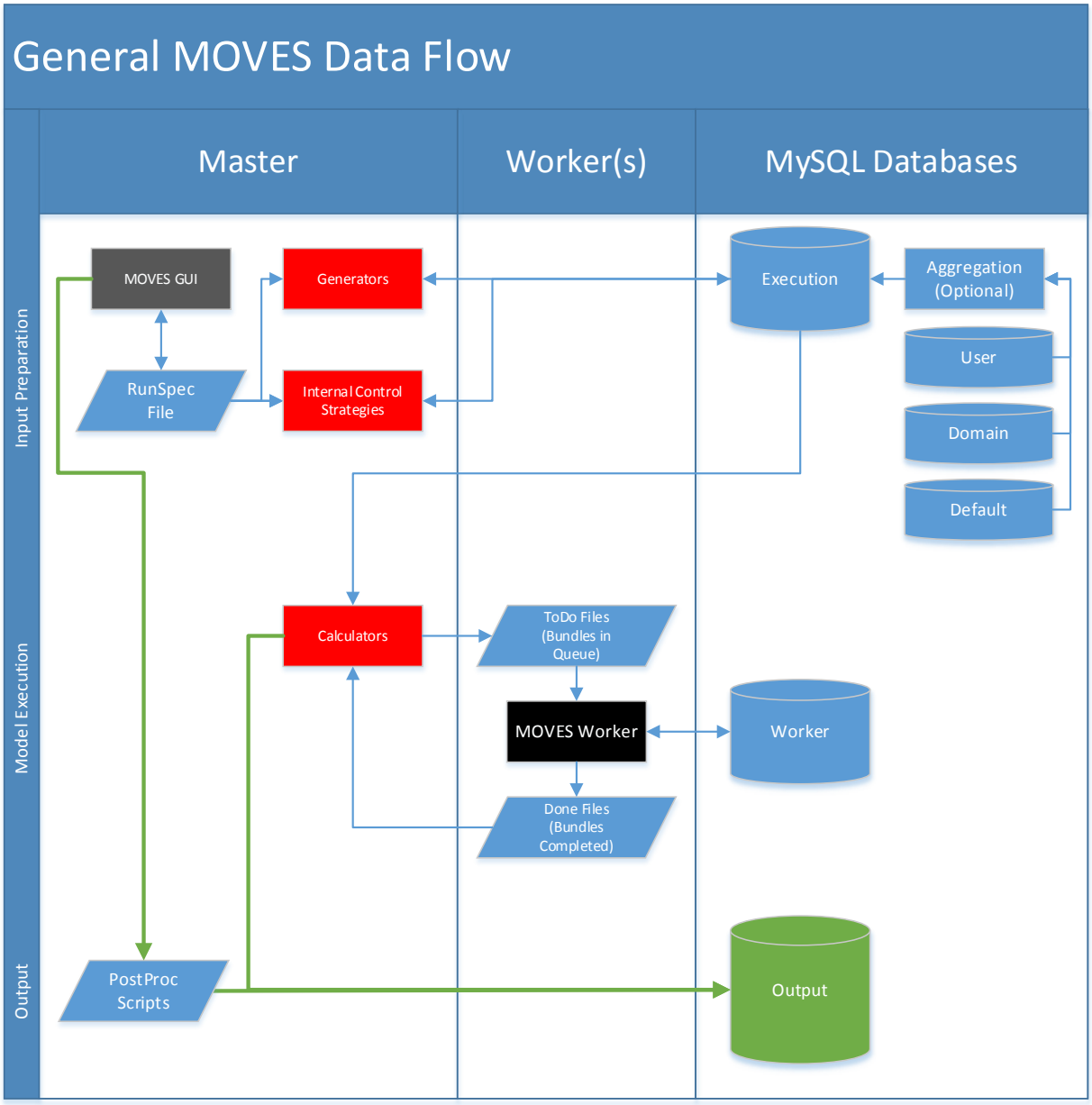
The MOVES uninstaller does not uninstall Java or MySQL. These components must be uninstalled separately using their own uninstallers, also accessible from Add/Remove Programs.

Chapter 4

MOVES Processing and Data Flow

Introduction

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



Graphical Conventions Used

- Cylinders represent databases.
- Slanted rectangles represent files or directories.
- Square boxes represent modules or processes.
- Solid line arrows represent data flow.

The modules highlighted in red are discussed in more detail in the sections that follow.

Typical Processing Steps

Before beginning a model run, any desired User Input Databases (shown at the right of the diagram) must be prepared. The diagram does not attempt to show this initial step, which is required if you need to modify 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 GUI can also be used to create an empty User Input Database to which you can add the table records you need.

A RunSpec is loaded or produced by using the MOVES GUI (MOVESWindow).

You initiate execution of the actual model run via the Action menu item in the GUI.

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 filtered out in this process.

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

MOVES Worker programs 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 RunSpec.

The Post-Processing menu in the MOVESWindow allows for additional, optional processing steps to be performed on the output database. The available post-processing scripts are described in the MOVES User's Guide.

MOVES Logical Level Data Flow

Users may access and update XML files containing saved Run Specifications (RunSpecs) using the MOVES GUI or an XML editor. Once it's complete, a RunSpec can be executed either from the GUI or with a simple MOVES Command Line Interface. Both the GUI and the command line allow you to import data from XLS, XLSX, TXT, and CSV files into MySQL databases for later use.

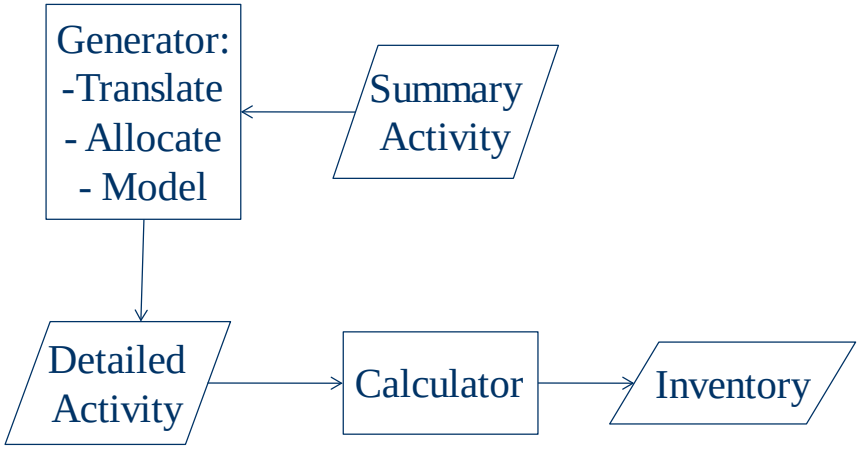
The Command Line Interface passes a selected RunSpec directly to the MOVES Application Program Interface (API), whereas the MOVES GUI Program allows you to select, edit, and save run specifications, and/or operate any importers before passing control to the MOVES API.

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 and domain 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 RunSpec.

MOVES Generators

The Master Loop sequences the operation of the generators, internal control strategies, and calculators. MOVES generators take data in summary or aggregate form and transform it to more detailed data to be used by MOVES calculators. Among others, MOVES 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. 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 (CMITs) and have an important role in the design of MOVES. Modelers can use their own external activity models and import data for these CMITs, causing MOVES to skip affected generators.

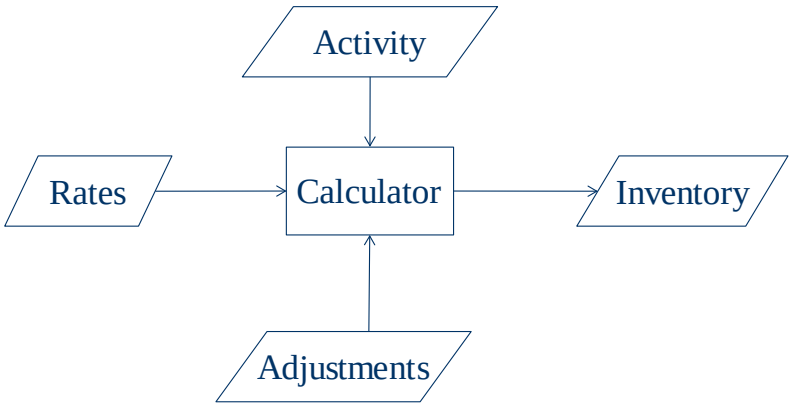


Internal Control Strategies

Internal Control Strategies are used to modify the inputs to generators and calculators, allowing what-if scenarios to be used without external models. The Rate-of-Progress Calculation strategy supports users modeling vehicle emissions for Reasonable Further Progress SIP requirements. If you select Compute Rate-of-Progress 'No Clean Air Act Amendments' Emissions, the MOVES model will assign 1993 model year emission rates to all post-1993 vehicles.

Emissions Calculators

Emission Calculators are the most central 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). Their main inputs are the emission rate, activity and adjustment tables within the MOVESExecution database.



Emission Calculators create data and instructions to be executed by Worker machines. Data and instructions are placed together into Bundle files that are compressed to save space and network transmission time. Worker computers process these bundled instructions and data then place their results back into the shared file folder. The result files created by the workers are then retrieved and loaded into the output database named in the RunSpec.

Chaining Calculators

Calculators can also be connected to one another, a process MOVES calls chaining. This is especially useful for pollutants calculated as ratios to other pollutants.

Chained calculators do not create their own work bundles. Rather, they add their calculation code and data to the bundles of another calculator.

After the Generators, Internal Control Strategies and Emission Calculators have executed, several more processing steps are required. Result Aggregation and Engineering Units Conversion functions are performed on the MOVESOutput database. If the Run Spec requires Emission Rates, an integrated Post-Processor for Emission Rate Lookup is then executed to produce an additional emission rate table in the output database.

Post-Processing

Following the model run, the MOVES GUI can be used to invoke additional post-processing functions to operate on MOVESOutput databases. Post-Processing Script Execution runs a selected MySQL script against the MOVESOutput database specified by the RunSpec. Several post-processing scripts are provided with the model and you may add to these if further customization of MOVES output is desired.

The rest of this section provides more detail on each processing step.

Graphical User Interface (GUI)/Run Specification Editor

The MOVES Graphical User Interface (GUI) is used to produce and modify MOVES RunSpecs.

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 pre-aggregation, and, when performing a county scale 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.

The MasterLooping mechanism uses a pool of control threads to bundle EmissonCalculator 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 programs. This thread also leads to the performance of the final result aggregation and unit conversion functions. If a rates run is being performed, an integrated post-processor is also invoked to create an additional table of emission rates in the output database.

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 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 used in addition to the default database to construct the MOVESExecution database. Only the values that have been successfully migrated to the MOVESExecution database are used by MOVES when performing calculations. The MOVESTableUsed table in the output database lists all of the sources for data used in the construction of the MOVESExecution database for each run.

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 used in MOVESExecution.

The InputDataManager adds 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 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.

Database Pre-Aggregation

To improve execution run time performance, when geographic selections are made at the state or national level, MOVES pre-aggregates 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 Geographic Bounds GUI screen and stored in MOVES RunSpecs. No database pre-aggregation 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 pre-aggregation is not allowed for rates calculations.

Options are also available to improve execution run time performance by pre-aggregating 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 pre-aggregation are not requested by the RunSpec.

Sequence of the Database Pre-Aggregation Operations

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

- 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 National Scale, there is a single link for each road type in the RunSpec.
- 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 National Scale, there is a single link in each such state for each road type in the RunSpec.
- 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 pre-aggregation is not allowed if evaporative emissions are being estimated.
- 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 pre-aggregation would remove this distinction.
- 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.
- Following any of the pre-aggregation operations performed, the set of ExecutionLocations used by the MasterLoop is recalculated based on the aggregated database.
- If the Day, Month, or Year aggregations have been performed, all information derived from the RunSpec 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.

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 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.

Algorithms Used to Perform the NATION and STATE Pre-Aggregations

The following table 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 exactly, 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.

Therefore, 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, 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.

Geographic Pre-Aggregation Algorithms

MOVES Database Table	GeographicSelectionType = NATION	GeographicSelectionType = STATE
State	Single Record, stateID=0, stateName=Nation, stateAbbr=US. Does not include Virgin Islands or Puerto Rico.	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,

MOVES Database Table	GeographicSelectionType = NATION	GeographicSelectionType = STATE
		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 from user input) Aggregate to national level roadType linkIDs, weighting by activity.	(contents 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 regions to a single new region. 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 regions in a state to a single new region. Since the fuel supply is in terms of fuel formulations, there may be many fuel formulations in the aggregate.
FuelUsageFraction	Activity-weighted aggregation of all old counties to single new county.	Activity-weighted aggregation of all old counties in a state to single new county.
IMCoverage	The Nation is considered to have IMCoverage with unknown (NULL) inspection frequencies, activity-weighted average complianceFactor values, and model year ranges which range from the earliest to the latest present at any location, for a given year, polProcess,fueltype and sourceType. This is an approximation.	Each I/M program present in a STATE is preserved with its complianceFactor values scaled by activity. Individual model year are computed separately for a given year, polProcess,fueltype and sourceType. Regclass is no longer used.
SHO	(contents from user input)	(contents from user input) Combine all links within state having same

MOVES Database Table	GeographicSelectionType = NATION	GeographicSelectionType = STATE
	Combine all links having same roadtype. SHO and distance are simple summations.	roadtype. SHO and distance are simple summations.
SourceHours	(contents from user input) Combine all zones. Starts field is a simple summation.	(contents from user input) Combine all zones within state. Starts field is a simple summation.
Starts	(contents from user input) Combine all zones. Starts field is a simple summation.	(contents from user input) Combine all zones within state. Starts field is a simple summation.
Extended IdleHours	(contents from user input) Combine all zones. extendedIdleHours field is a simple summation.	(contents 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 from user input) Combine all zones, averageTankTemperature values are activity-weighted.	(contents from user input) Combine all zones in state, averageTankTemperature values are activity-weighted.
SoakActivity Fraction	(contents from user input) Combine all zones. soakActivityFraction values are activity-weighted.	(contents from user input) Combine all zones in state. soakActivityFraction values are activity-weighted.
ColdSoakTank Temperature	(contents from user input) Combine all zones. coldSoakTankTemperature values are activity-weighted.	(contents from user input) Combine all zones in state. coldSoakTankTemperature values are activity-weighted.
ColdSoakInitialHourFraction	(contents from user input) Combine all zones. coldSoakInitialHourFraction values are activity-weighted.	(contents from user input) Combine all zones in state. coldSoakInitialHourFraction values are activity-weighted.
AverageTank Gasoline	(contents from user input) Combine all zones, ETOHVolume and RVP values are activity-weighted.	(contents from user input) Combine all zones in state, ETOHVolume and RVP values are activity-weighted.

Algorithms Used to Perform the Time Period Pre-Aggregations

The following table 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 pre-aggregation 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 exactly, 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. Therefore, 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: 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: 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: 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: based on the DAYVMTFraction table, with MonthID removed by using weights from MonthVMTFraction. Its key fields are sourceTypeID, roadTypeID, and dayID.
- DayWeighting1Normalized: based on DayWeighting1 such that the day fractions sum to 1.0 for each sourceTypeID and roadTypeID combination.
- DayWeighting2: based on DayWeighting1, with roadTypeID removed by using information from RoadTypeDistribution. Its key fields are sourceTypeID and dayID.

- DayWeighting2Normalized: based on DayWeighting2 such that the day fractions sum to 1.0 for each sourceTypeID.
- DayWeighting3: based on DayWeighting2 and uses the distribution for sourceTypeID = 21 for all sourceTypes.
- DayWeighting3Normalized: based on DayWeighting2Normalized 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.

Description of Time Period Pre-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	

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)
HPMSVTypeDay		Activity-weighted VMT using DayWeighting3Normalized	Activity-weighted VMT using MonthWeighting
OpMode Distribution (contents from user input)	Activity-weighted average opModeFraction using HourWeighting1	Activity-weighted average opModeFraction using DayWeighting1	
SourceTypeDayVMT		Activity-weighted VMT using DayWeighting2Normalized	Activity-weighted VMT using MonthWeighting
SourceType Hour	Hour-level idleSHOFactors are summed	Activity-weighted average idleSHOFactor using DayWeighting2	
SHO (contents 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 from user input)	Simple summation of starts	Simple summation of starts	Simple summation of starts
Extended IdleHours (contents 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	

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)
SampleVehicleDay		Set dayID = 0 for all vehicle days. Assumes vehIDs are unique across dayIDs	
StartsPerVehicle **(contents from user input)**	hourDayID = dayID Simple summation of startsPerVehicle	hourDayID = 0. Simple summation of startsPerVehicle	
FuelSupply			Activity-weighted marketshare using MonthGroupWeighting. (default values produced by DefaultDataMaker, must be considered)
AverageTank Temperature **(contents from user input)**	Activity weighted averageTankTemperature using HourWeighting4	Activity weighted averageTankTemperature using DayWeighting3	Activity weighted averageTankTemperature using MonthWeighting
SoakActivity Fraction **(contents from user input)**	Activity-weighted soakActivityFraction using HourWeighting2	Activity weighted average soakActivityFraction using DayWeighting2	Activity weighted average soakActivityFraction using MonthWeighting

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 and day 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.

Result Data Aggregation and Engineering Units Conversion

This function aggregates the results produced by the MOVES emission calculators 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. Engineering unit conversion only occurs after all worker bundles have been successfully retrieved by the master. As such, output for partially completed runs will be in MOVES' internal units of grams, miles, kilojoules, and hours.

How Result Aggregation Levels are Specified

The level of aggregation in the MOVES results is specified on the MOVES GUI Output Emissions Detail 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 pre-aggregation 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 pre-aggregation of the database was performed. Output is always distinguished by pollutant, year, and MOVES run ID.

When reporting for entire months, MOVES scales up the results by the number of weeks in each month (i.e. by the number of days in the month divided by seven). When reporting annual results, MOVES sum over all the months selected by the user.

Vehicle type output may be distinguished by Source Type, Source Classification Code (SCC) or regulatory class. Specific rules for acceptable combinations of these options are enforced by the MOVES GUI.

Output may optionally be distinguished by:

- Model Year
- Fuel Type
- Emission Process
- Road Type

Nonroad output may be further distinguished by:

- Fuel Subtype
- Sector
- Engine Technology
- Horse Power Class

In general, any combination of these distinctions may be specified. Output by SCC implies that roadtype, fueltype, emission process, and source type 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.

Result Aggregation Algorithm - Logical Level Specification

The raw emissions output of MOVES is distinguished by year, month, day, hour, state, county, zone, link, road type, pollutant, process, source use type, fuel type, fuel subtype (Nonroad only), model year, regulatory class, SCC, sector (Nonroad only), engine technology (Nonroad only), and horse power class (Nonroad only). In an Onroad run three fields are always null, including: sector, engine technology, and horse power class. In a Nonroad run, different fields will be null (hour, zone, link, road type, source use type, and regulatory class). Taken together these fields may uniquely label the full-detail emissions output in a model run that doesn't have any aggregation; however, if MOVES aggregations are performed they may assume the null value.

At National Scale, 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 National Scale, 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.

- Output may be aggregated to be by source type only, by SCC and source type, or neither. The GUI enforces specific rules to ensure all data required to create an SCC code is retained in the output.
- Output may be aggregated to be by regulatory class. This is optional and available with many other aggregations.
- 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.

- 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. One seventh of these days are considered to be Mondays, one seventh Tuesdays, etc. without regard to calendar year. A warning is generated if this aggregation is performed and not all days are 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 RunSpec.
- Months are totaled to years if the output time period is year. A warning is generated if this aggregation is performed and not all months are 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 RunSpec.
- Road types are combined into county totals if the Road Type check box is not 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. A warning message is generated if this aggregation is performed and all road types are not included in the RunSpec.
- 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 ensure that all desired states are included in domain totals.
- Fuel types are combined within source use types if the Fuel Type check box is not selected in the Output Emission Detail panel. It is the user's responsibility to ensure that all desired fuel types are included.
- Fuel subtypes are combined within fuel types if the Fuel Subtype checkbox is not selected in the Output Emission Detail panel.
- Emission processes are combined if the Emission Process check box not selected in the Output Emission Detail panel. It is the user's responsibility to ensure that all desired processes are included.

Nonroad activity includes Average Horsepower (activity type 9), Fraction Retrofitted (10), and Load Factor (12). These three activity types are themselves weighted averages that cannot be simply summed like all other emissions and activity. Instead, these activities are weighted using their corresponding Source Hours (2) activity:

$$Aggregated\ Activity = \frac{\sum(source\ hours * activity)}{\sum source\ hours}$$

Note: Fraction Retrofitted (10) is not currently output by the model but is included in the aggregation weighting algorithm for completeness.

Engineering Unit Conversion

Engineering units are selected by the user in the run specification for mass, energy, and distance. In the MOVES GUI, these are specified on the Outputs panel within the General Output Screen.

Supported are:

- 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. Note that the distance units are only required if the RunSpec 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. When the time unit equals the output time period specified on the Output Emissions Detail screen, the quantities reported amount to an inventory for that time period.

Fuel Subtype Detail

Nonroad emissions output can be disaggregated to include fuel subtype. The Nonroad model outputs data by fuel type. MOVES' Go-based external calculator splits the output by the market share of each fuel formulation, and hence for each fuel subtype. This is a linear split by market share.

After the output of the Nonroad model has been split by market share, further emissions are computed at the fuel formulation level and aggregated to either the fuel subtype or fuel type level. Pollutants computed by fuel formulation include HC speciation pollutants (methane, VOC, NMOG, TOG, NMHC), NonHAPTog, and air toxics (including dioxins, metals, benzene, etc).

MOVES Calculators and Generators

The inputs, outputs and detailed algorithms for MOVES2014 are available in a separate HTML document, The MOVES2014a Module Reference. This allows the reader to use live links to follow cross-references. The MOVES2014a Module Reference provides details on the following MOVES modules:

- Activity Calculator
- AirToxicsCalculator
- Average Speed Operating Mode Distribution Generator
- Base Rate Generator
- BaseRateCalculator
- CO2AERunningStartExtendedIdleCalculator
- Crankcase Emission Calculator
- Distance Calculator
- Evaporative Emissions Operating Mode Distribution Generator
- Evaporative Permeation Calculator

- Fuel Effects Generator
- HC Speciation Calculator
- Link Operating Mode Distribution Generator
- Liquid Leaking Calculator
- Meteorology Generator
- NO Calculator
- NO2 Calculator
- NonroadEmissionCalculator
- PM10 Brake Tire Calculator
- PM10 Emission Calculator
- Rates Operating Mode Distribution Generator
- Refueling Loss Calculator
- SO2 Calculator
- SourceBin Distribution Generator
- Start Operating Mode Distribution Generator
- Sulfate PM Calculator
- TOG Speciation Calculator
- Tank Fuel Generator
- Tank Temperature Generator
- Tank Vapor Venting Calculator
- Total Activity Generator

The MOVES Algorithm Reference also provides detailed information on all the tables of the MOVES default database.

In addition, MOVES2014a now contains several new Go Language calculators. These are listed below.

- HCSpeciation
- nrAirToxics
- nrHCSpeciation

The MOVES Algorithm Reference does not currently provide detailed information on the tables or algorithms in the Go Language calculators. However, this improvement is planned for the future.

Chapter 5

Design Concepts

Introduction

This chapter describes important concepts and terminology used in the MOVES model. The functional scope of MOVES2014 is characterized as follows:

Geographic Bounds: 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.

Vehicles/Equipment: All highway vehicle sources, divided into 13 use types. (Nonroad equipment is also covered by MOVES2014, but the Nonroad components are not covered by this document.)

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, VOC, air toxics, dioxins and furans, sulfates, air quality TOG species, and several forms of particulate matter (PM).

Emission Processes: running, start, extended idle (e.g. heavy-duty truck hoteling), brakewear, tirewear, evaporative permeation, evaporative fuel vapor venting, evaporative fuel leaks, crankcase, refueling, and auxiliary power exhaust.

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.

Geographic Bounds

The default geographic modeling domain in MOVES 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, we list all counties that have existed in the time period from 1990 to July 2014, including counties such as Clifton Forge, VA that existed only in some of those years. These correspond to the 3228 political subdivisions of the states. Counties must belong to a single state.

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 and counties in the default database are further divided into links. The default database of each county is divided into five links; four of these represent actual roadways and one represents locations not on the county's roadway network.

The nine road types in MOVES are:

1. Locations which are off of the highway network
2. Rural restricted access roadways (i.e. freeways and interstates), including ramps
3. Rural roads to which vehicle access is unrestricted
4. Urban restricted access roadways (i.e. freeways and interstates), including ramps
5. Urban roads to which vehicle access is unrestricted
6. Rural restricted access roadways (i.e. freeways and interstates) without ramps
7. Urban restricted access roadways (i.e. freeways and interstates) without ramps
8. Ramps for Rural restricted access roadways (i.e. freeways and interstates)
9. Ramps for Urban restricted access roadways (i.e. freeways and interstates)

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 eight 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.

MOVES at National Scale has links that are a combination of a road type and a county or zone. Road types, while not geographic locations, help to define links at the macroscale. MOVES can also be run in an emission rate table lookup mode in which a link represents all highway segments in the county or zone which have the same roadtype and average speed. When modeling at project scale, links may represent segments of actual roadways and road types serve only to classify links.

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 MOVES default database 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 shorter 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 not account for leap years.

In order to model an actual historical time period, data corresponding to the unique time period could be supplied by you, and you 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 RunSpec when estimating evaporative emissions and this area of the model does assume an hourly time sequence.

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 do this mapping internally, so that the casual user of MOVES will not have to deal with external mapping. Doing this, however, requires some complexity in the design.

Onroad 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.

MOVES Source Use Type Definitions

HPMS Class	MOVES Use Type	Description
Light Duty Vehicles	21. Passenger Car	
	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/acceleration 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. Because Source Classification Codes (SCCs) have been and are used extensively in emission inventories, the current version of MOVES also offers the option of reporting results by SCC. The SCC scheme in MOVES2014 is new and is and not compatible with SCC results in previous versions. Unlike the prior version, SCC is native to MOVES2014 and is available without impacting performance or data size.

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 vehicle types). Second, the factors that influence fuel consumption and emission production are different from those that affect 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 emissions or energy consumption. 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.

Source bins are defined independently from use types, but are mapped to use types internal to MOVES by the SourceBinDistributionGenerator using the data within the SampleVehiclePopulation table.

Nonroad equipment types are explained in the NONROAD model user guide.[2]

Emission Pollutants

MOVES estimates four fundamentally different kinds of results: energy consumption, mass, toxic equivalency, and gram-mole emissions. For convenience, all these quantities are considered to be emissions. Energy emissions estimated by MOVES are total energy consumption, fossil fuel energy consumption, and petroleum fuel energy consumption. The more familiar mass emissions estimated by MOVES include 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. Toxic equivalency (TEQ) emissions are a measure of human toxicity and are used for dioxins and furans. Gram-mole emissions are used for air

[2] User's Guide for the Final NONROAD2005 Model, EPA420-R-05-13, December 2005. http://www3.epa.gov/otaq/models/nonrdmdl/nonrdmdl2005/420r05013.pdf

quality modeling of chemical speciation mechanisms. The exact set of pollutants and processes available in a release of the MOVES model is available via the MOVES GUI.

Emission Processes

On-road vehicles consume energy and produce emissions through several mechanisms or pathways, which are known within MOVES as emission processes or processes and are accounted for and can be reported 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, fuel refueling, tire wear and brake wear.

The processes for MOVES are:

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

Start Exhaust: 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: 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 and is meant to account for the issue of overnight hoteling at truck stops.

Auxiliary Power Exhaust: The energy consumed, or the vehicle emissions produced, during operation of auxiliary power units on heavy duty diesel vehicles.

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

Evaporative Fuel Vapor Venting: 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: The gross leaking of fuel, in liquid form, from the vehicle. This is assumed to subsequently evaporate, outside the vehicle, into the atmosphere.

Brakewear: The formation of airborne particles of brake components which are formed during operation of vehicle brakes.

Tirewear: The formation of tire material airborne particles during vehicle operation

Crankcase Emissions: The formation of engine combustion products which escape directly from the engine or a road draft tube rather than through the vehicle tailpipe. Such emissions can be produced in three processes - Crankcase Running, Crankcase Start and Crankcase Extended Idle.

Refueling Emissions: The expulsion into the atmosphere of fuel vapor during refueling. Also included are vapors from liquid fuel spilled during refueling.

Well-To-Pump: The energy and emissions produced from processing and distributing vehicle fuel from raw feedstock to the fuel pump. These emissions are not available in MOVES2010 or MOVES2014.

Vehicle Fuel Classifications

Implicit in this source bin classification scheme described above is the idea 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."

MOVES2014a is designed to model vehicles of the following fuel types for onroad vehicles:

- Gasoline
- Diesel Fuel
- Compressed Natural Gas (CNG)
- Electricity
- Ethanol (E85)

MOVES2014a is designed to model vehicles of the following fuel types for nonroad equipment types:

- Gasoline
- Diesel Fuel
    - Land equipment
    - Marine equipment (recreation only)
- Compressed Natural Gas (CNG)
- Liquid Propane Gas (LPG)

In the MOVES2014a default database, on-road vehicles and nonroad equipment use the same gasoline fuel formulations, *except nonroad has no E15 fuels*. Diesel fuel formulations are different for onroad and nonroad, and nonroad has different diesel formulations for land and marine use. Also, the nonroad model does not model high ethanol fuels (E85) or electric fuels. However, it does model LPG fuels. Conversely, the on-road models electric fuels but no LPG fuels.

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). Onroad diesel fuel has three subtypes: conventional, biodiesel, and Fischer-Tropsch diesel. Nonroad diesel fuel has two fuel types (*no further fuel subtype distinctions*): Land equipment and Marine equipment. The Marine diesel fuel is for recreational equipment only and does not contain heavy fuel types such a bunker fuels. 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.

This fuel classification scheme further divides 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 distinction is necessary because these fuel characteristics affect emissions. 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.

Onroad MOVES handles flexible fueled vehicles. Populations for these vehicles are stored in the Sample Vehicle Population table or imported by their base fuel type, such as Ethanol or Electricity. At runtime, allocations for the actual type of fuel being used by such vehicles are applied using the fuelusagefraction table. For instance, it is possible to assign 75% of ethanol flexible-fuel vehicles to actually consume gasoline and 25% to consume E85. These allocations can vary by county, month, and calendar year. Emission results are reported by the fuel consumed. As such, imported vehicle population counts may differ from output population counts by fuel type since imported data is by equipment and output data is by activity. The nonroad model does not have the capability to model fuel usage fractions between two fuel types.

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 with regard to energy consumption and emissions? The first question is quantified in MOVES by the metric Total Activity which is the total amount of vehicle activity for source use types in the given location and time which the user has selected in the RunSpec. The basis of total activity depends on the emission process. In MOVES, the Total Activity Generator (TAG) estimates Total Activity.

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 RunSpec. The same as number of sources times 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 RunSpec. 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 RunSpec. The same as number of sources times per-source starts
Extended Idle	Hotelling Hours	The total number of hours spent by truck drivers at their trucks during mandated “down” periods between trips during long haul operations.
Auxiliary Power	Hotelling Hours	

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. In the MOVES design, these operating modes are allowed to vary by emission process and pollutant. For some pollutant-processes, Total Activity is not further divided into multiple operating modes.

For the running emission process for all pollutants except 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) or Scaled Tractive Power (STP). STP is used for heavy duty vehicles.

Operating Mode Bin Definitions for most Running Emissions

Brakewear Stopped (Bin 501) 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	-	-
<6			Bin 33
6 to 12	-	-	Bin 35
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

The start exhaust process emissions are distinguished into operating modes which represent the length of time the engine was off prior to starting.

Operating Modes for most Start Emissions

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 - opModeID=150 and cold soaking – opModeID=151. 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 brakewear 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. The Brakewear pollutant-process combination contains emission rates for only selected opModeIDs values. It contains values for the three speed ranges, for the braking operation and for idle operation. The Braking from Stopped

Vehicles operating mode applies only in cases where the vehicle is stopped and the brakes are applied such as in the single second when a vehicle is decelerating into a full stop.

Operating Modes for Brakewear

opModeID	VSP Range	opModeID
0		**Braking (Bin 0)**
1		**Idle (Bin 1)**
11	< 0 kW/tonne	11
21	< 0 kW/tonne	21
33	< 6	33
501		Braking from Stopped Vehicles

Tirewear is a function of only speed and the operating mode reflect the standard MOVES speed bins. The opModeID of 400 is used in MOVES to model tire wear from stopped vehicles. The tirewear from stopped vehicles occur in the second that the car is either accelerating from rest or decelerating to a full stop.

Operating Modes for Tirewear

opModeID	Lower Speed	Upper Speed
400	Idle	
401	0	2.5
402	2.5	7.5
403	7.5	12.5
404	12.5	17.5
405	17.5	22.5
406	22.5	27.5
407	27.5	32.5
408	32.5	37.5
409	37.5	42.5
410	42.5	47.5
411	47.5	52.5
412	52.5	57.5
413	57.5	62.5
414	62.5	67.5
415	67.5	72.5
416	72.5	Greater than 72.5

Hotelling and Extended Idle

Hotelling hours are a type of activity that need additional explanation. Hotelling hours is nominally the total number of hours spent by truck drivers at their trucks during mandated "down" periods between trips during long haul operations. During this time, emissions may be generated while powering heating, air conditioning, fans and other accessories. "Extended idling" refers to the operation of the main truck engine during hotelling. "Auxiliary power units" (APUs) are small diesel fueled engines installed on long haul trucks for powering accessories without running the main truck engine. Accessories can also be

powered by an AC plug in, where available, or with battery power. Hotelling hours can also be spent with all engines and accessories off.

The HotellingActivityDistribution table includes the activity fraction for each of the hotelling operating modes, show below. Further, this distribution varies by model year, allowing phase-ins of APUs and electrification.

OpModeID	Description
200	Extended Idling (truck engine idling)
201	Hotelling Diesel Aux (APU engine operation)
203	Hotelling Battery AC (electrical hookup)
204	Hotelling APU Off (all engines off, parked)

Note the inclusion of the “Hotelling APU Off” operating mode (204). This is a catch-all to ensure all hotelling hours have been accounted for in a scenario. As such, this distribution will always sum to 1.0.

Chapter 6

MOVES Input and Default Databases

Introduction

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. Generally, users should not change the default database. Instead, you can change MOVES defaults using the MOVES GUI program and MOVES RunSpecs, which allow you to specify one or more input databases whose table records are to be appended 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. As described in the MOVES User Interface Reference Manual, MOVES includes "importer" programs that help develop appropriate inputs.

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.

Changing the MOVES Default Database

Generally users should not change the default database. It is possible to override values in almost any table in the database using the Manage Input Data option of MOVES without making changes to the default database tables. However, great care is needed to assure that the data provided using the Manage Input Data option to make changes completely replaces the default rows. You should contact EPA if you are unsure of how this can be done.

If for some reason, a user prefers to make their changes permanent so that alternate tables are not required for all local MOVES runs, EPA suggests that a copy of the default MOVES database be made with a new database name that can be edited to include the alternate values. The alternate database should be carefully documented to describe in detail all changes made to the default tables. This documentation should be provided with any submission of results from MOVES using any alternate database.

MOVES Importers and Importer Validation Scripts

MOVES importers help users create input databases that facilitate replacement of MOVES defaults with user-supplied data. These are described generally in the MOVES User Guide. Specific importers that require more explanation are described in an appendix to this document.

Within the importers, MOVES uses scripts to validate information imported to user databases. The mechanism passes information to the script and relies upon the script for the red/green icon determination.

These validation scripts make extensive use of MySQL stored procedures to error check imported data. Using stored procedures within the MOVES/MySQL scripting system requires some special handling. Stored procedures require the use of semicolons within the stored procedure, yet all the SQL for a stored procedure must be submitted to MySQL as a single SQL statement. BeginBlock and EndBlock are words that should be put before and after a MySQL “create procedure” statement and code block and should not use semicolons themselves. An abbreviated example from IMCoverageImporter.sql:

```
BeginBlock
create procedure spCheckIMImporter()
begin
……..
end
EndBlock
```

In this way, the MOVES SQL scripting engine can present the entire stored procedure block to MySQL as a single statement as MySQL requires.

Many RunSpec variables are passed to the importer scripts.

Variable	Description
##defaultDatabase##	Name of the default database. Use this to avoid hardcoded database names in the importer scripts.
##mode##	0 when the script is called to validate imported data, 1 when called to provide red/green indication by adding either NOT_READY or OK to the importTempMessages table.
##countyIDs##	Comma separated list of counties in the RunSpec. Will be “0” if there are no selections.
##dayIDs##	Comma separated list of days in the RunSpec. Will be “0” if there are no selections.
##fuelSubtypeIDs##	Comma separated list of fuel subtypes in the RunSpec. Will be “0” if there are no selections.
##fuelTypeIDs##	Comma separated list of fuel types in the RunSpec. Will be “0” if there are no selections.
##fuelYearIDs##	Comma separated list of the fuel years implied by the calendar year selections in the RunSpec. Will be “0” if there are no selections.
##hourDayIDs##	Comma separated list of hour/days in the RunSpec. Will be “0” if there are no selections.
##hourIDs##	Comma separated list of hours in the RunSpec. Will be “0” if there are no selections.
##hpmsVtypeIDs##	Comma separated list of HPMS vehicle types in the RunSpec. Will be “0” if there are no selections.
##monthGroupIDs##	Comma separated list of month groups in the RunSpec. Will be “0” if there are no selections.
##monthIDs##	Comma separated list of months in the RunSpec. Will be “0” if there are no selections.
##polProcessIDs##	Comma separated list of pollutant/processes in the RunSpec. Will be “0” if there are no selections.
##processIDs##	Comma separated list of emission processes in the RunSpec. Will be “0” if there are no selections.
##regionIDs##	Comma separated list of fuel regions based upon the geography within the RunSpec. Will be “0” if there are no selections.
##roadTypeIDs##	Comma separated list of road types in the RunSpec. Will be “0” if there are no selections.
##sourceFuelTypeIDs##	Comma separated list of source/fuel type combinations in the RunSpec. These are expressed as (sourceTypeID*100 + fuelTypeID), similar to the mechanism used with pollutant/process combinations. Will be “0” if there are no selections.
##sourceTypeIDs##	Comma separated list of source types in the RunSpec. Will be “0” if there are no selections.
##stateIDs##	Comma separated list of states in the RunSpec. Will be “0” if there are no selections.
##yearIDs##	Comma separated list of years in the RunSpec. Will be “0” if there are no selections.
##zoneIDs##	Comma separated list of zones in the RunSpec. Will be “0” if there are no selections.

Note that many of these variables can never be empty, getting the value of "0" if the RunSpec contains no selections for the item. This feature allows these variables to be safely used within SQL IN clauses such as:

```
…. WHERE sourceTypeID IN (##sourceTypeIDs##) ….
```

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 single- or double-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.

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 many 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 Emission Calculators, (which can be considered to implement the MOVES Core Model), use as their principal inputs. These are called core model input tables (CMITs). The reason CMITs are important to the advanced

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). 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.

More Information about Database Tables and Their Use

The MOVES technical reports, available on the MOVES website, describe the sources and derivations of MOVES emission rates, vehicle activity data, and other the data stored in the MOVES information tables. The MOVES Algorithms Reference document, available on the MOVES web page and also easily generated from the MOVES code, is an htm document that lists each of the tables in the MOVES database and provides each table's "CREATE TABLE" statement. This provides format information for each of the fields in every table. In addition, for each table, the Algorithms Reference lists all the MOVES modules that use the table.

More detailed documentation of the MOVES database is contained in a readme directory within the MOVES default database directory. This readme directory contains a Microsoft WORD document named MOVESDB.doc that lists 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 forms of .pdf and .tif files.

The graphical conventions which apply to these diagrams are:

- 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 referred 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.

The location of the readme folder depends on your MOVES installation. If the MySQL data path was installed at the EPA suggested location on your hard drive and MOVES was installed with the MOVES installation package, the readme directory will be either "C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.6\data\ MOVESDByyyymmdd\readme" (for Windows XP or 2003 server), or "C:\Users\Public\MySQL\MySQL Server 5.6\ data\MOVESDByyyymmdd\readme" (for

Windows 7,8, or 2008 server), where yyyymmdd is the version date. However, if the MySQL data path was installed at a default location provided by MySQL install program or at a location specified by the user, the readme directory could be "C:\ProgramData\MySQL\MySQL Server 5.6\ data\MOVESDByyyymmdd\readme", "C:\MySQL\data\ MOVESDByyyymmdd\readme", a directory under a user specified data directory, or a directory under a data directory chosen by the MySQL install program.

Chapter 7

MOVES Output Database

Introduction

The MOVES GUI/Master program reports the results of each simulation run in the MySQL database named in the run specification. The results of multiple runs may be stored in the same database and are identified by MOVESRunID. This output database consists of several tables.

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 RunSpec upon which this run was based. The contents of the RunSpec may have been changed since the run was performed.
- The runSpecDescription field contains the description portion of the RunSpec.
- The runspecFileDateTime field indicates the time and date stamp the RunSpec file had when the run was executed. If this date-time differs between two runs using the same RunSpec file name, this may be an indication that some change was made to the RunSpec 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.

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. Warning-level and error level messages are written to the output file; informational 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 MOVES.

Other Substantive MOVES Output Tables

This section discusses the MOVESActivityOutput, MOVESOutput, and rate output tables. These tables contain the substantive results of each model run. All 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 MOVESOutput table is used to report the pollutant emission results, including energy consumption.

The rate output tables are only populated when runs request Rates instead of Inventories. Such runs require special indexing that would be a burden on normal MOVES runs and are separated for performance purposes.

The fields of the output 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. This use is not supported in MOVES2014.
- 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 RunSpec.
- 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 RunSpec.
- 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 RunSpec.
- 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 RunSpec.

- 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 pre-aggregation computational shortcuts are taken, 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 MOVES 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. For county-level runs, 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. For project-level runs, the values are user-defined. A null or zero value indicates that the record pertains to all links in the county or zone that were included in the RunSpec. In the default database distributed with MOVES, this corresponds to all road types in the county that were included in the RunSpec. This field has a special meaning in rates runs.
- 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. 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. A null value indicates that the output record does not pertain to a particular SCC.
- 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 RunSpec.
- The fuelSubTypeID 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. This input is available only for a nonroad run. An onroad run will always report a null value. Whether results are to be distinguished by fuel type is the nonroad GUI selection and is included in the RunSpec.
- 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 RunSpec.
- 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 MOVES, there are several roadtypes intended to classify actual highways, plus a RoadType representing locations in the zone which are not on the highway network. MOVES2014 also allows users to distinguish output for ramps.

- 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.
- The processID field numerically identifies the emission process 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. 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 RunSpec.
- The emissionQuant field is present only in the MOVESOutput table and 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 RunSpec, and output in the MOVERun table.
- The emissionQuantMean field is present only in the MOVESOutput table. It contains a zero value.
- The emissionQuantSigma field is present only in the MOVESOutput table. It contains a zero value.
- The activity field is present only in the MOVESActivityOutput table and contains the a measure of the activity associated with the activityTypeID. It's engineering unites depend on the activity type and user selections.
- The activityTypeID field is present only in the MOVESActivityOutput table. It references the ActivityType table which defines the available types of activity information.

The RatePerDistance table has the following fields:

- MOVESScenarioID
- MOVESRunID
- yearID
- monthID (may be aggregated out)
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- linkID
- pollutantID
- processID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)

- modelYearID (may be aggregated out)
- roadTypeID
- avgSpeedBinID (as determined through joins with the LinkAverageSpeed Table)
- temperature (as determined through joins with the ZoneMonthHour table)
- relHumidity (as determined through joins with the ZoneMonthHour table)
- ratePerDistance (Zero miles for any link leads to divide-by-zero errors. In such cases there are no emissions reported.)

The RatePerHour table has the following fields:

- MOVESScenarioID
- MOVESRunID
- yearID
- monthID (may be aggregated out)
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- linkID
- pollutantID
- processID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)
- modelYearID (may be aggregated out)
- roadTypeID
- temperature (as determined through joins with the ZoneMonthHour table)
- relHumidity (as determined through joins with the ZoneMonthHour table)
- ratePerHour (Zero hours for any link leads to divide-by-zero errors. In such cases there are no emissions reported.)

The RatePerProfile table has the following fields:

- MOVESScenarioID
- MOVESRunID
- temperatureProfileID
- yearID
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- pollutantID
- processID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)
- modelYearID (may be aggregated out)
- temperature (as determined through joins with the ZoneMonthHour table)
- relHumidity (as determined through joins with the ZoneMonthHour table)

- ratePerVehicle (rate per existing vehicle even if the vehicle had zero activity)

The RatePerStart table has the following fields:

- MOVESScenarioID
- MOVESRunID
- yearID
- monthID (may be aggregated out)
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- zoneID
- pollutantID
- processID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)
- modelYearID (may be aggregated out)
- temperature (as determined through joins with the ZoneMonthHour table)
- relHumidity (as determined through joins with the ZoneMonthHour table)
- ratePerStart (Zero starts for any zone leads to divide-by-zero errors. In such cases there are no emissions reported.)

The RatePerVehicle table has the following fields:

- MOVESScenarioID
- MOVESRunID
- yearID
- monthID (may be aggregated out)
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- zoneID
- pollutantID
- processID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)
- modelYearID (may be aggregated out)
- temperature (as determined through joins with the ZoneMonthHour table)
- relHumidity (as determined through joins with the ZoneMonthHour table)
- ratePerVehicle (rate per existing vehicle even if the vehicle had zero activity)

The StartsPerVehicle table has the following fields:

- MOVESScenarioID
- MOVESRunID
- yearID
- monthID (may be aggregated out)
- dayID (may be aggregated out)
- hourID (may be aggregated out)
- zoneID
- sourceTypeID (may be aggregated out)
- regClassID (may be aggregated out)
- SCC
- fuelTypeID (may be aggregated out)
- modelYearID (may be aggregated out)
- startsPerVehicle (starts per existing vehicle even if the vehicle had zero activity)

Additional Output Tables

Other output tables are useful for reference and diagnostic purposes: These tables are:

- BundleTracking—used for tracking the progress of large runs
- MovesEventLog—used for diagnostics
- MovesTablesUsed—used to track which input tables have been used for a run
- MovesWorkersUsed—used for diagnostics when running MOVES on multiple workers

Appendix A

List of Acronyms

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
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
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 Monitoring System
ID	identification

IM	inspection/maintenance
kW	kilowatt
LDT	light duty truck
LDV	light duty vehicle
LPG	liquefied propane gas
LTLP	lookup table link producer
MAR	mileage accumulation rate
MB	megabyte
MOVES	MOtor Vehicle Emissions Simulator
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
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

Running MOVES from the Command Line

The MOVES command line allows MOVES to be executed without running its graphical user interface. It is useful in situations where repeated or unattended runs are needed, or when another computer program executes MOVES. This interface presumes that a MOVES RunSpec file has been prepared and that you are running from the DOS prompt.

These instructions presume some familiarity with DOS commands. The DOS commands to execute the MOVES Command Line Interface are:

```
java gov.epa.otaq.moves.master.commandline.MOVESCommandLine -r runspecfile
```

[OR]

```
java -XmxNNNm gov.epa.otaq.moves.master.commandline.MOVESCommandLine -r runspecfile
```

[OR]

```
java gov.epa.otaq.moves.master.commandline.MOVESCommandLine -rl runspeclistfile
```

[OR]

```
java XmxNNNm gov.epa.otaq.moves.master.commandline.MOVESCommandLine -rl runspeclistfile
```

where

-XmxNNNm is an optional parameter and NNN is a three digit integer number. For example, –Xmx200m specifies you are asking Java to allocate 200 megabytes of heap memory for your runs. This optional parameter is required only if the MOVES runs need more heap memory than the default determined automatically by Java and your machine. The actual maximum amount of heap memory that you can ask for depends on your machine. You can increase or decrease the amount of heap memory as your machine allows. For example,

```
java -Xmx512m gov.epa.otaq.moves.master.commandline.MOVESCommandLine -rl runspeclistfile
```

[OR]

```
java –Xmx512m gov.epa.otaq.moves.master.commandline.MOVESCommandLine -r runspecfile
```

runspecfile is the name of a file containing a saved MOVES run specification (e.g. C:\YourPATHto\YourRunSpecName.mrs) and *runspeclistfile* is the name of a text file containing a list of run specification file names, with one per line. The rest of the syntax is

literal. The spelling of MOVESCommandLine is case sensitive because it is a Java class name. When doing a batch run, you can identify the RunSpec list text file in the command line using quotes because it is located in a folder with spaces (e.g. C:\My Documents). However, the command line doesn't work if you put quotes in the RunSpec list file itself. The command line will think the file doesn't exist.

By executing one of these commands, you run the Java interpreter (java.exe). MOVESCommandLine is a DOS parameter telling java.exe which Java class file to begin executing, and the last two tokens are parameters passed to MOVESCommandLine. Either version of this command can be executed from a DOS batch file, and batch files can contain multiple commands. Prior to executing the command, the active directory should be set to the location where MOVES is installed, typically C:\Program Files\MOVES, and the SETENV.BAT file should be executed.

It can be difficult for DOS to find everything. For the command to work, three elements must be found:

- The Java interpreter
- The MOVESCommandLine Java class
- Any runspeclistfile and all runspecfiles

Running SETENV.BAT insures that the Java interpreter is found. Running SETENV.BAT also insures that the procedure described in the next paragraph results in the MOVESCommandLine Java class being found. Java experts can also use the CLASSPATH environment variable more directly to locate Java class files.

If MOVES has been installed in the default location, C:\Program Files\MOVESyyyymmdd or C:\Users\Public\MOVESyyyymmdd,then MOVESCommandLine.class is located at C:\Program Files\MOVESyyyymmdd\gov\epa\otaq\moves\master\commandline. This can be made the active directory, or the command can specify whatever part of the path is needed. For example, if the active directory is C:\Program Files\MOVESyyyymmdd, the command line interface class would be specified as gov.epa.otaq.moves.master.commandline.MOVESCommandLine.

One way to insure that the runspecfile or the runspeclistfile is found is to specify the full path. If a simple file name is used the file should be located in the active DOS directory.

If the GUI is used to create an importer XML file. The XML file can be edited and executed via the command line. When using the command line, text output is not sent to the screen but instead stored in MOVESBatch.log. An example command line is:

```
java gov.epa.otaq.moves.master.commandline.MOVESCommandLine
 -i importAllFromTemplates.xml
```

Place all of this on one line. The “-i” option directs MOVES to the XML file that describes the import actions to occur. Note that this XML file contains a summary of the RunSpec within it, thus allowing wildcards even when importing via the command line. For MOVES installations on Windows 2000 platforms, there may be a problem with the DOS command

string length not allowing more than 126 characters. To resolve this, move MOVES to the root of C: drive (C:\MOVES\), so that the length of the command line string will be shorter. Note that Windows2000 is not a recommended or supported environment in which to run MOVES.

If the GUI is used to create an importer XML file. The XML file can be edited and executed via the command line. When using the command line, text output is not sent to the screen but instead stored in MOVESBatch.log.

Additional ANT commands supported by MOVES2014 include the following commands. Some of these are explained in more detail in the appendix on configuring MOVES.

docs	generate Java docs
rungui	display the MOVES Master user interface
runworker	display the MOVES Worker user interface
crungui	compile and run the MOVES Master user interface
crunworker	compile and run the MOVES Worker user interface
errormessages	Generate error message RTF file providing more detail on errors
algorithms	Generate algorithm HTML file
startmysql	start a local copy of MySQL
stopmysql	stop the local copy of MySQL
1worker	start a local copy of MySQL and 1 local worker using the manyworkers.txt configuration file. The worker will shutdown after there have been no TODO or InProgress files present for 2 minutes. After the worker stops, the local copy of MySQL is stopped as well. Before using, edit manyworkers.txt to use your system's sharedDistributedFolderPath. If the MySQL daemon is already running, a local copy of MySQL will not be started nor will thethe daemon be stopped, but the worker(s) will still be started. The ANT flag -Dnoshutdown=1 will start the worker but never shut it down even when work is no longer available.
2workers	like 1worker but with 2 concurrent workers and 1 MySQL instance
4workers	like 2workers but with 4 concurrent workers
6workers	like 2workers but with 6 concurrent workers
8workers	like 2workers but with 8 concurrent workers

12workers	like 2workers but with 12 concurrent workers
14workers	like 2workers but with 14 concurrent workers
20workers	like 2workers but with 20 concurrent workers
24workers	like 2workers but with 24 concurrent workers
28workers	like 2workers but with 28 concurrent workers
32workers	like 2workers but with 32 concurrent workers
maketodo	start a local copy of MySQL and 1 master, set to only generate TODO files without getting DONE files or deleting TODO, DONE, or InProgress files upon exiting. Use -Drunspec= to name the one runspec that will be used. Before using, edit maketodo.txt to use your system's system's sharedDistributedFolderPath. If the MySQL daemon is already running, a local copy of MySQL will not be started nor will the daemon be stopped, but the master will still be started. IMPORTANT: Use setlogin first. ant -Drunspec=c:\myrunspecs\runspec1.mrs maketodo
master1worker	start a local copy of MySQL and 1 master, set to start a single worker and wait for its DONE files. Like maketodo, use -Drunspec= to specify the runspec. Before using edit both maketodo.txt and manyworkers.txt for your system's master and worker path settings. If the MySQL daemon is already running, a local copy of MySQL will not be started nor will the the daemon be stopped, but the master, and its will still be started. IMPORTANT: Use setlogin first.
pickup	start a local copy of MySQL and a master that will pickup DONE files. Use -Dpdspec= to Before using, edit maketodo.txt to use your system's sharedDistributedFolderPath. If the MySQL daemon is already running, a localcopy of MySQL will not be started nor will the daemon be stopped, but the master will still be started. IMPORTANT: Use setlogin first. ant -Dpdspec=c:\myinfo\filestoget.xml pickup
convert	run a database conversion script. IMPORTANT: Use setlogin first.ant -Dinput=2010Binput -Doutput=2014input -Dscript=Convert2010B_CDM_PDM.sql convert
run	Execute a runspec. IMPORTANT: Use setlogin first. Use -Drunspec= to name the one runspec that will be used. ant -Drunspec=c:\myrunspecs\runspec1.mrs run
setlogin	store database user and password ant -Duser=moves -Dpassword=secret setlogin

Appendix C

Configuring MOVES

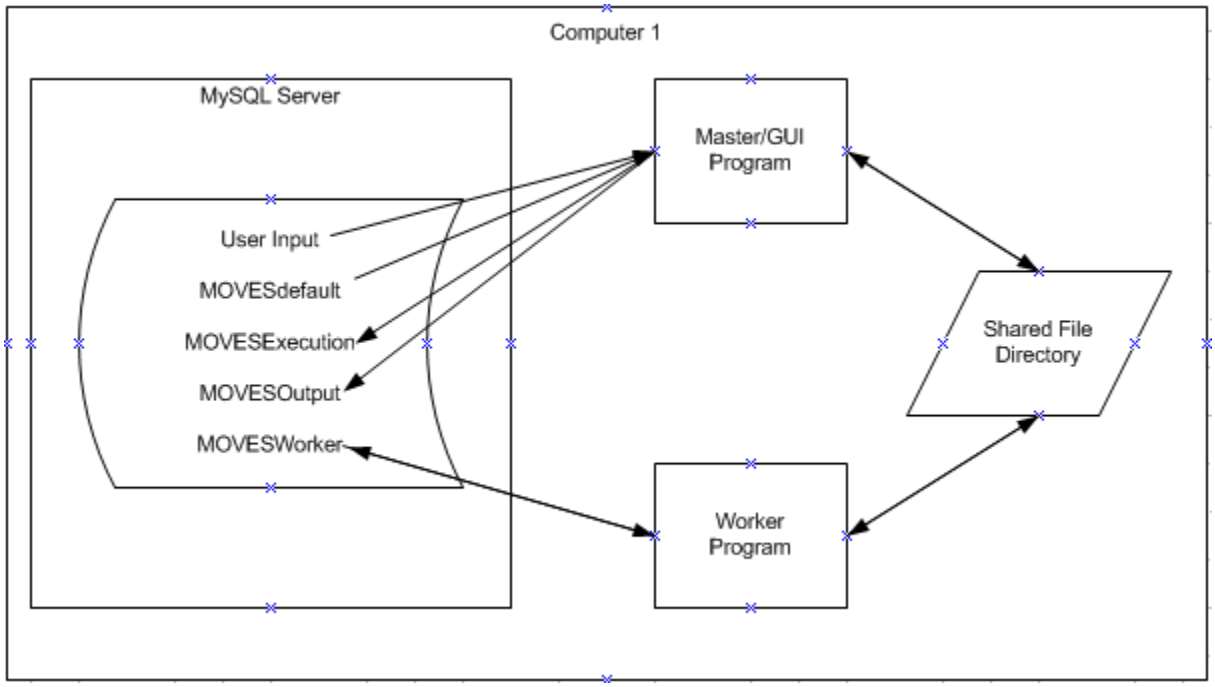
Introduction

This appendix discusses some technical details associated with MOVES configuration, DOS Command Line instructions and ANT options. MOVES may be configured beyond the default configuration built by the MOVES Installation Suite. **This appendix is meant for advanced users who are thoroughly familiar with MOVES and who desire ways to obtain maximum performance from MOVES.**

MOVES Configurations

All MOVES application components may be installed on a single computer system as shown below.

MOVES Default Configuration

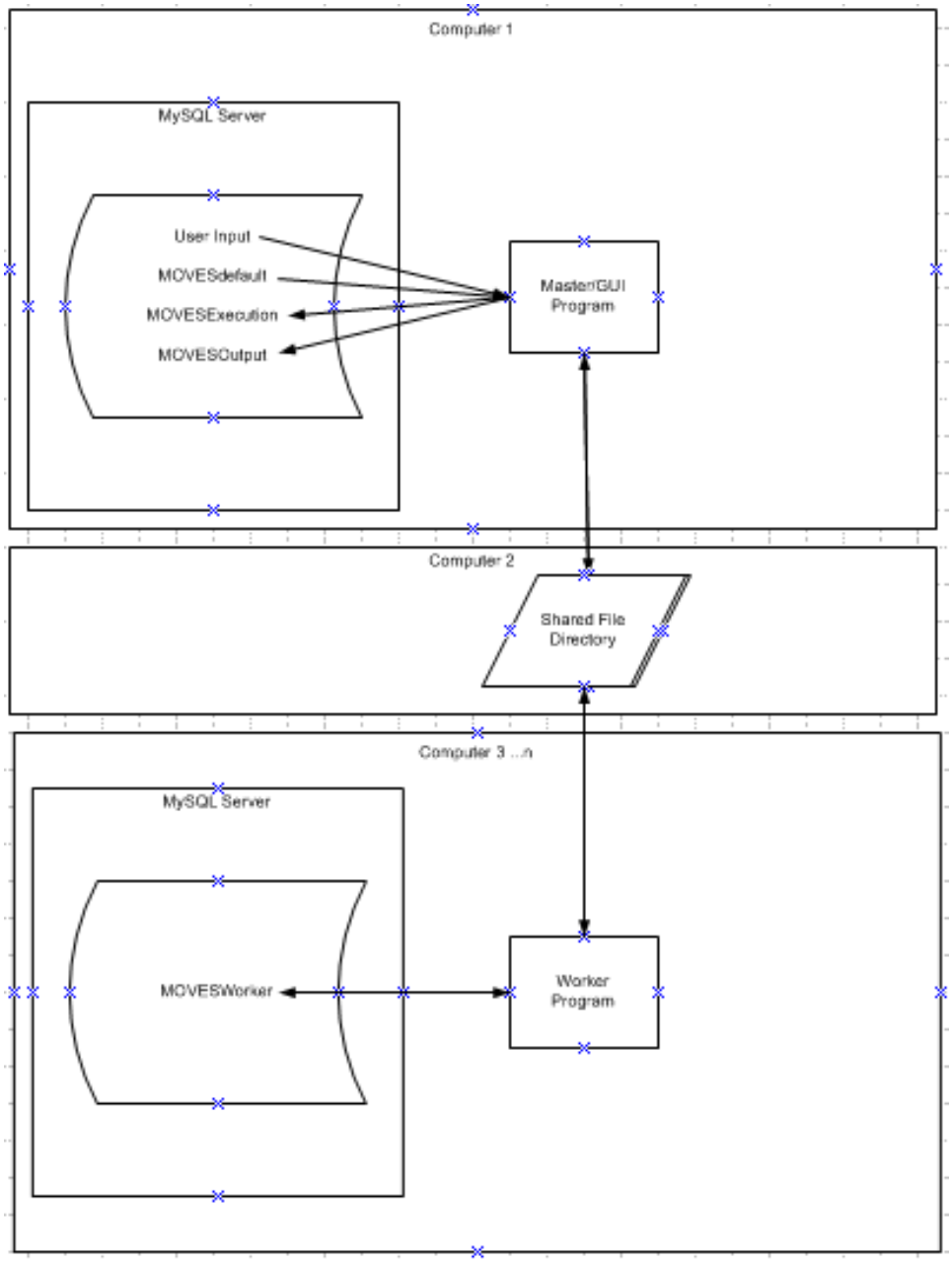


MOVES may also be configured so that several computers work together to execute model simulation runs. This can significantly improve execution time performance of large simulations. Improvements diminish, however, as more worker computers are added. The number of worker computers needed to approach 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. 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 MOVES Default 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.

MOVES Multiple Computer Configuration



Enhanced Grid Operations

The following MOVES features enhance LINUX cluster and GRID (PC-based) usage:

- All of the MOVES components, Master, Worker, and MySQL can run on separate computing nodes.
- MOVES can work with a non-local MySQL server. This is useful for a centralized default database, but not recommended for multiple worker and master execution databases due to contention on the server's hard drive I/O bus.
- MOVES can start and/or stop a local MySQL server without needing MySQL to be pre-installed as a daemon/service and without requiring administrator privileges.
- MOVES can start a master that will only produce TODO files then exit (optionally combined with the start/stop of a local MySQL).
- MOVES can start N workers simultaneously on a single node and optionally have those workers quit when they run out of TODO files (optionally combined with the start/stop of a local MySQL).
- MOVES masters can process DONE files for other masters (optionally combined with the start/stop of a local MySQL), storing them to an output database (either local or centralized).
- MOVES has simplified environment variables. Only Java and ANT must be in the path and the ANT script sets the class path. This ensures that MOVES runs only need relative paths, hoping to make shell scripting MOVES easier. This also allows a work flow of TODO creation, TODO processing into DONE files, and DONE file storage, making batched/queued processing much easier to accomplish.

These features are implemented within the ANT script that MOVES uses (build.xml), making all features portable between Windows and Linux.

Configuring MOVES for Use with Multiple Workers

The following sections describe typical configurations and additional functionality that assist with using MOVES in the command line, batched, shared environment of a typical Linux-based or GRID cluster. These sections describe command line options for which there are often no GUI counterparts. The setup of a local copy of MySQL, one that does not require a pre-installed service/daemon on each computing node, is also described.

Quick Setup of 2 Workers

To quickly start two concurrent workers on either Linux or Windows using the default MOVES configuration, follow these steps.

1. Edit the manyworkers.txt file to use the same shared directory as is set in the MOVESConfiguration.txt file.
2. Start a command line and CD to the directory containing your MOVES installation, such as C:\Users\Public\EPA\MOVES\MOVES2014a\
3. At the command line, type: setenv.bat
4. At the command line, type: ant -Dnoshutdown=1 2workers
5. Two worker windows will start and stay running. Do not close the command line window.

Choosing a Configuration

Configuration 1: Silo with Local MySQL

A Silo configuration places both a master and one or more workers on a single computer. Variations include the number of concurrent workers and serial or concurrent execution of the master and worker(s). The default configuration of MOVES is a silo with local MySQL with one worker running concurrently with the master.

To implement Configuration 1 on a single computer node:

1. Ensure Java is available.
2. Edit maketodo.txt which was installed along with the MOVES source code. Ensure the defaultDatabaseName value is the name of the database you will be using. Ensure sharedDistributedFolderPath value is the correct location of the directory your master and workers will use.
3. Edit manyworkers.txt which was installed along with the MOVES source code ensuring the sharedDistributedFolderPath value is the correct location of the directory your master and workers will use.
4. Install MySQL onto the computing node you will be using or create a version that can be used without installation. A no-installation version is provided with the Windows version of MOVES (under the mysql folder within the MOVES base installation directory). Installing on Linux will require compiling a version of MySQL that has default directories that are actually available on your computing node. Such compilation is beyond the scope of this document.
5. Within your MOVES code folder, create a PDSpec XML file named pickupall.xml suitable for use in your environment and name a specific database to hold the

results. Set the <directory> entry to the name of your shared work folder. Such a file is:

```
<pdspec>
	<pdentry>
		<masterid>*</masterid>
		<directory>c:\moves\sharedwork</directory>
		<output servername="" databasename="movesresults" />
	</pdentry>
</pdspec>
```

To use Configuration 1:

1. Set the JAVA_HOME environment variable to the location of your Java installation, such as:

```
C:\Program Files (x86)\Java\jdk1.7.0.
```

2. Set the ANT_HOME environment variable to the location of your MOVES ant folder, such as:

```
C:\Users\Public\EPA\MOVES\MOVES2014a\ant.
```

3. Set the PATH environment variable to include JAVA_HOME\bin and ANT_HOME\bin.
4. Execute any of the MOVES ant commands such as “ant –Drunspec=c:\myrunspecs\runspec1.mrs run”.

Configuration 2: Silo with Local MySQL, Serial Master and Two Workers

The Silo with Local MySQL, Serial Master and Two Workers configuration places a single master, a MySQL instance, and two workers on a single computer. The workers execute after the master, allowing TODO and DONE files to be preserved as well as the output database. When the workers complete and DONE files picked up, no further calculations are needed on the RunSpec. An external scripting tool must be used to execute a series of RunSpecs. A single instance of MySQL is used, running on the same computer node that runs the master and workers. This limits network access and ensures the MySQL security system denies access from other network nodes - a benefit when securing a large cluster of machines.

To implement Configuration 2 on a single computer node:

1. Ensure Java is available.
2. Edit maketodo.txt which was installed along with the MOVES source code. Ensure the defaultDatabaseName value is the name of the database you will be using. Ensure sharedDistributedFolderPath value is the correct location of the directory your master and workers will use.
3. Edit manyworkers.txt which was installed along with the MOVES source code ensuring the sharedDistributedFolderPath value is the correct location of the directory your master and workers will use.
4. Install MySQL onto the computing node you will use or create a version that can be used without installation. A no-installation version is provided with the Windows version of MOVES (under the mysql folder within the MOVES base installation directory). Installing on Linux will require compiling a version of MySQL that has default directories that are actually available on your computing node. Such compilation is beyond the scope of this document.
5. Within your MOVES code folder, create a PDSpec XML file named pickupall.xml suitable for use in your environment and name a specific database to hold the results. Set the <directory> entry to the name of your shared work folder. Such a file is:

```
<pdspec>

      <pdentry>

            <masterid>*</masterid>

            <directory>c:\moves\sharedwork</directory>

            <output servername="" databasename="movesresults" />

      </pdentry>

</pdspec>
```

To use Configuration 2:

1. Set the JAVA_HOME environment variable to the location of your Java installation, such as:

```
C:\Program Files (x86)\Java\jdk1.7.0.
```

2. Set the ANT_HOME environment variable to the location of your MOVES ant folder, such as:

```
C:\Users\Public\EPA\MOVES\MOVES2014a\ant.
```

3. Set the PATH environment variable to include JAVA_HOME\bin and ANT_HOME\bin.

4. Execute the "ant maketodo" command, passing your RunSpec name using the "-Drunspec=" parameter, such as:

```
ant -Drunspec=c:\myrunspecs\runspec1.mrs maketodo
```

5. If desired, save all *TODO files from your shared work folder.
6. Execute the "ant 2workers" command. This will start 2 concurrent workers which will terminate after completing all TODO files.
7. If desired, save all *DONE files from your shared work folder.
8. Execute the "ant pickup" command, passing your pickupall.xml filename using the "-Dpdspec=" parameter, such as:

```
ant -Dpdspec=pickupall.xml pickup
```

9. Copy your output database files to long term storage. The database is the one named in the pickupall.xml file, not the one used by the RunSpec.

Configuration 3: Worker Pool with Local MySQL

This configuration places multiple workers and an instance of MySQL on each computing node. An instance of MySQL is used running on the same computer node that runs the workers. This limits network access and ensures the MySQL security system denies access from other network nodes - a benefit when securing a large cluster of machines.

To implement Configuration 3:

1. Ensure Java is available.
2. Edit manyworkers.txt which was installed along with the MOVES source code. Ensure the sharedDistributedFolderPath value is the correct location of the directory your masters and workers will use. This will be a network accessible directory since external masters will deposit TODO files and retrieve DONE files from this location.
3. Install MySQL onto each computing node you will be using or create a version that can be used without installation. Such a no-installation version is provided with the Windows version of MOVES (under the mysql folder within the MOVES base installation directory), but under Linux will require compiling a version of MySQL that has default directories that are actually available on your nodes. Such compilation is beyond the scope of this document.

To use Configuration 3:

1. Set the JAVA_HOME environment variable to the location of your Java installation, such as:

```
C:\Program Files (x86)\Java\jdk1.7.0.
```

2. Set the ANT_HOME environment variable to the location of your MOVES' ant folder, such as:

```
C:\Users\Public\EPA\MOVES\MOVES2014a\ant.
```

3. Set the PATH environment variable to include JAVA_HOME\bin and ANT_HOME\bin.
4. Execute the "ant 2workers" command with the "-Dnoshutdown=1" parameter. This will start 2 concurrent workers which will never terminate. Example:

```
ant -Dnoshutdown=1 2workers
```

Ant Tasks and MOVES Command Line Options

MOVES use of Ant has been expanded to include features normally accomplished with shell scripts, thereby bringing these features to all platforms. In addition, production-related Ant tasks now set their class paths internally using relative directories. Such tasks, such as 2workers, only require the calling shell script to set a path to Java and Ant.

Ant Task	Description
Startmysql	Utility to start a local MySQL instance if another instance is not already running.
Stopmysql	Utility to stop any MySQL server using the standard localhost port 3306. This will stop MySQL started with the "startmysql" task or any preexisting MySQL service/daemon.
1worker	Use a local copy of MySQL and a single local worker using the manyworkers.txt configuration file and -autoshutdown worker command line option. The Ant flag "-Dnoshutdown=1" can also be given. See the full description in the multiple workers section below.
2workers **3workers** **4workers** **5workers** **6workers** **7workers** **8workers**	Use a local copy of MySQL and two or more local workers using the manyworkers.txt configuration file and -autoshutdown worker command line option. You may use more 8 workers if your MOVES version provides. Example command line: ant 2workers The Ant flag "-Dnoshutdown=1" can also be given to cause the spawned workers to never shutdown even if no work is available, by omitting the "-autoshutdown" flag when spawning worker instances. Example command line: ant -Dnoshutdown=1 2workers
maketodo	Start a local copy of MySQL and one master that is set to only generate TODO files without getting DONE files or deleting TODO, DONE, or InProgress files upon exiting. Before using, edit the maketodo.txt file to use your system's sharedDistributedFolderPath. Use "-Drunspec=" to specify the runspec. Example command line: ant -Drunspec=MyCounty2030.mrs maketodo
pickup	Start a local copy of MySQL and one master that will retrieve and process DONE files. Before using, edit the maketodo.txt file to use your system's sharedDistributedFolderPath. Use "-Dpdspec=" to name the required PDSpec XML file. Example command line: ant -Dpdspec=c:\myinfo\filestoget.xml pickup
master1worker	Start a local copy of MySQL with one master that is set to create one worker and wait for its DONE files. Before using, edit maketodo.txt and manyworkers.txt. Like maketodo, use "-Drunspec=" to specify the runspec. Example command line: ant -Drunspec=MyCounty2030.mrs master1worker
setlogin	Store a database username and password into the configuration files. Example command line: ant –Duser moves –Dpassword=secret setlogin

run	**Execute a runspec. Use setlogin first to ensure a connection to the databases can be established. Example command line: ant –Drunspec=c:\myrunspecs\runspec1.mrs run**

Note: Several Ant tasks make use of a local MySQL instance. These tasks detect the presence of a running MySQL instance and will start, use, then stop a local MySQL instance, if one was not already running.

Master Command Line Options

The MOVES master supports the following command line options to its Java class (gov.epa.otaq.moves.master.commandline.MOVESCommandLine).

Option	Description
-i IMPORTXMLFILE	Perform all data import operations as specified in the import XML file. This allows automation of the operations normally performed via the GUI using the County Data Manager or Project Data Manager. An example import XML file can be obtained through these GUI panels. **Note:** MOVES requires access to a graphical user interface, such as Windows or an X Windows server, to use this feature even though no graphical elements are displayed during the process.
-maketodo Or -onlytodo	Causes the command line master to only produce TODO files, ignoring DONE files and to not delete its TODO, InProgress, or DONE files upon exit. This is an essential feature for cluster computing where a master may not be allowed to run while workers process its TODO files. This mode also prevents a local worker from being started.
-noworker	Causes the command line master to not start a local worker. This behavior is automatic with the "-maketodo"/"-onlytodo" option.
-workerconfig=FILENAME	Specify an alternate configuration file to be used if a local worker is started. Ant tasks use the manyworkers.txt configuration file extensively. The default file used is WorkerConfiguration.txt.
-p PDSPECXMLFILE	Collect all DONE files specified in the PDSpec XML file. The master GUI has a tool to generate these XML files.
-r RUNSPEC	Run a single RunSpec file name (with optional path).
-rl RUNSPECLISTFILE	Specify a text file containing a list of RunSpec file names (with optional paths) to be run, one runspec per line. This is equivalent to using the "-r" option on each RunSpec named in the text file.

Setting the Master Configuration File

The master loads its configuration file before its command line processing takes place. As such, there is no command line option for the master to alter its use of a configuration file. Instead, the master uses a Java Property named "MOVES_CONFIGURATION_FILE_NAME" to specify an alternate configuration file that should be read. When this property is set and set to non-empty, its value is used as the name and optional path to the master's configuration file.

Worker Command Line Options

The MOVES worker supports the following command line options to its Java class (gov.epa.otaq.moves.worker.WorkerCommandLine):

Option	Description
-autoshutdown	Close the worker instance after no TODO or InProgress files were found in the worker's shared work folder for 2 minutes. Workers will continue to exist, idling, if there are InProgress files in the shared work folder, in case the workers performing the work in those files fail. This option is useful on a shared computer where workers cannot be left running indefinitely, as is their default without this command line option.
-longshutdown	Similar to -autoshutdown but uses a 2 hour timeout instead of a 2 minute timeout.
-config=FILENAME	Specify an alternate configuration file to be used. Ant tasks use the manyworkers.txt configuration file extensively.

Configuration Files

MOVES uses several configuration files, all of which have case-sensitive names.

File	Purpose
MOVESConfiguration.txt	Default master configuration file. Rarely used with cluster computing. Used with the master GUI ("ant -rungui"). defaultServerName = localhost defaultDatabaseName = movesdb20150925 executionServerName = localhost executionDatabaseName = MOVESExecution outputServerName = 127.0.0.1 outputDatabaseName = MOVESOutput nonroadExePath = C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08a\NONROAD.exe sharedDistributedFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork computerIDPath = C:\Users\Public\EPA\MOVES\MOVES2014a\MOVESComputerID.txt masterFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a saveTODOPath =

	mysqlUserName = <User Name> mysqlPassword = <Password>
WorkerConfiguration.txt	Default worker configuration file. Rarely used with cluster computing. Used with the traditional worker GUI ("ant -runworker"). workFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\WorkerFolder workerDatabaseName = MOVESWorker computerIDPath = workerServerName = localhost sharedDistributedFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork concurrentStatements = 1 nonroadApplicationPath = C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08a\nonroad.exe nonroadWorkingFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08a calculatorApplicationPath = C:\Users\Public\EPA\MOVES\MOVES2014a\calc\go\externalcalculatorgo64.exe workerDebug = False mysqlUserName = <User Name> mysqlPassword = <Password>
maketodo.txt	Master configuration file specified by Ant tasks for cluster computing. Modify this file to specify your system's shared folder for TODO and DONE files. Do not modify data server or database names in this file. defaultServerName = localhost defaultDatabaseName = movesdb20150925 executionServerName = localhost executionDatabaseName = * outputServerName = localhost outputDatabaseName = MOVESOutput nonroadExePath = C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08a\NONROAD.exe sharedDistributedFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork computerIDPath = masterFolderPath = . saveTODOPath = mysqlUserName = <User Name> mysqlPassword = <Password>
manyworkers.txt	Worker configuration file specified by Ant tasks for cluster computing. Modify this file to specify your system's shared folder for TODO and DONE files. Do not modify data server or database names in this file. sharedDistributedFolderPath = C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork workFolderPath = manyworkers/workerfolder workerDatabaseName = *

```
computerIDPath =
workerServerName = localhost
concurrentStatements = 1
nonroadApplicationPath =
C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08
a\nonroad.exe
nonroadWorkingFolderPath =
C:\Users\Public\EPA\MOVES\MOVES2014a\NONROAD\NR08
a
calculatorApplicationPath =
C:\Users\Public\EPA\MOVES\MOVES2014a\calc\go\externalc
alculatorgo64.exe
mysqlUserName = <User Name>
mysqlPassword =  <Password>
```

Accessing Remote MySQL Servers

It may be desirable to centralize some of the databases used by MOVES, such as the default database and county - or project-specific input databases. Storing these databases on a single MySQL instance can make management easier, especially if the data is changed frequently, as often occurs during initial stages of data testing.

RunSpec files can contain the server name of input databases, county domain databases, project domain databases, and even output databases. These all have servername XML attributes that when left blank indicate use of localhost.

The default database used by the MOVES master can be set in MOVESConfiguration.txt and maketodo.txt. Both support providing not only the server name, but also a user name and password, if required. The relevant configuration entries are:

- defaultServerName
- defaultDatabaseName
- defaultUserName
- defaultPassword

The database used by the master to store execution data can be stored on a remote MySQL server and is set within MOVESConfiguration.txt and maketodo.txt. The relevant configuration entries are:

- executionServerName
- executionDatabaseName
- executionUserName
- executionPassword

The executionDatabaseName entry also supports use of a single asterisk ("*") to indicate use of an automatically assigned database name. If two or more masters wish to use the same MySQL instance (whether remote or local), each must have a different execution database name. This can be accomplished by manually configuring each to have a specific name or by setting each with "executionDatabaseName = *" in their configuration files.

The database used by each worker to store temporary calculations can be stored on a remote MySQL server and is set within WorkerConfiguration.txt and manyworkers.txt. The relevant configuration entries are:

- workerServerName
- workerDatabaseName
- workerUserName
- workerPassword

The workerDatabaseName entry supports use of a single asterisk ("*") to indicate use of an automatically assigned database name. If two or more workers wish to use the same MySQL instance (whether remote or local), each must have a different worker database name. This can be accomplished by manually configuring each to have a specific name or by setting each with "workerDatabaseName = *" in their configuration files.

Local MySQL

Ant can start a local instance of MySQL, one that has not been pre-installed and that runs entirely within the MOVES installation folder. When Ant uses such a local instance, it follows these steps:

1. Checks for an existing MySQL instance by checking the standard MySQL server TCP/IP port 3306.
2. If a local MySQL exists, then Ant performs the desired operation that uses MySQL, such as running a master or worker.
3. Otherwise, Ant starts the MySQL server process, performs the desired operation that uses MySQL, then shutdowns the MySQL server process.

It is important to note that with the above steps, Ant tasks that use a local MySQL instance will not shutdown a MySQL service/daemon that was already running. However, the Ant task stopmysql will stop an existing service/daemon, since the task has no information available about how MySQL was started.

When Ant launches a MySQL server process, the process is bound to the localhost/127.0.0.1 network only, making it inaccessible from other nodes. MySQL's security policies are still enforced though, requiring user IDs and passwords.

Before you can use the local MySQL features, MySQL files that are appropriate for your system must be provided, be it Windows or Linux, 32-bit or 64-bit. To begin this process, locate the mysql directory under the installation folder of MOVES. It contains a file called my.ini. This is the local MySQL's configuration and you may update the contents to optimize the memory usage for your system. Do not, however, modify any directory paths in this file, especially the basedir and datadir entries. It is also advisable to edit manyworkers.txt and maketodo.txt to your system's shared work folder.

Creating TODO Files

The MOVES master has a mode that only generates TODO files without picking up, or processing DONE files. If using the direct command line, use the "-r" or "-rl" command line options along with the "-maketodo" option. If using Ant, use the "maketodo" task, such as: ant -Drunspec=c:\myrunspecs\runspec1.mrs maketodo.

Before using the Ant "maketodo" task, configure your local MySQL and edit maketodo.txt to use your system's shared work folder. Do not alter server or database names in maketodo.txt.

Running More than 2 Workers on One Node

Multiple MOVES workers can be run on a single node. Each worker requires a unique database name and the easiest method of doing this is by setting the workerDatabaseName entry to "*" in WorkerConfiguration.txt and manyworkers.txt. There are technically other methods of accomplishing this with earlier versions of MOVES but they are substantially more error prone and will not be covered here.

The Ant 1worker, 2workers, and other tasks use the manyworkers.txt configuration file, passing it to the workers using the -config=FILE command line parameter along with the -autoshutdown flag. Workers launched with these tasks will terminate gracefully after 2 minutes without TODO or InProgress files.

Before using the Ant tasks, configure your local MySQL and edit manyworkers.txt to use your system's shared work folder. Do not alter server or database names in manyworkers.txt.

Processing DONE Files - Command Line DONE Files

A MOVES master can pick-up DONE files created for other masters. When doing so, it creates records in its own output database (as opposed to the output database designated in the original RunSpec). The master "-p" command line option is used to automate the retrieval of these files. If using Ant, use the "pickup" task such as:
ant -Dpdspec=c:\myinfo\filestoget.xml pickup

When using these options, you must provide a "PDSpec" XML file (Pickup Done Specification). An example file follows:

```
<pdspec>
	<pdentry>
		<masterid>5880486781340065678</masterid>

	<directory>C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork</directory>
		<output servername="" databasename="PDTest" />
	</pdentry>
	<pdentry>
		<masterid>3686433322585912670</masterid>

	<directory>C:\Users\Public\EPA\MOVES\MOVES2014a\SharedWork</directory>
		<output servername="" databasename="PDTest" />
	</pdentry>
```

```
</pdspec>
```

The above file shows two sets of DONE files that will be retrieved, both from the same folder and both destined for the same output database. Each will get a separate MOVESRun entry in the database.

Processing DONE Files - GUI DONE Files

MOVES has a GUI for creating these PDSpec XML files and for processing DONE files. The GUI is activated from the **Tools** menu. To use the GUI, click **Process Done File**, click **Browse DONE Files** and select a **DONE** file. You will be presented with details extracted from the bundle.

If the DONE file is desired, click **OK**. This will populate the Master ID and Folder fields in the PDSpec GUI. Next, specify the output database just as is done with the primary MOVES GUI then click the **Use** button. Click the **Save** button to generate a PDSpec XML file from your selections.

Click the **Load** button to populate the selections list with the contents of an existing PDSpec XML file. Click the **Done** button to close the GUI without processing the DONE files, but with the option to save your selections into an XML file for later use. Click the Initiate **Processing** button to pick up the DONE files immediately.

Appendix D

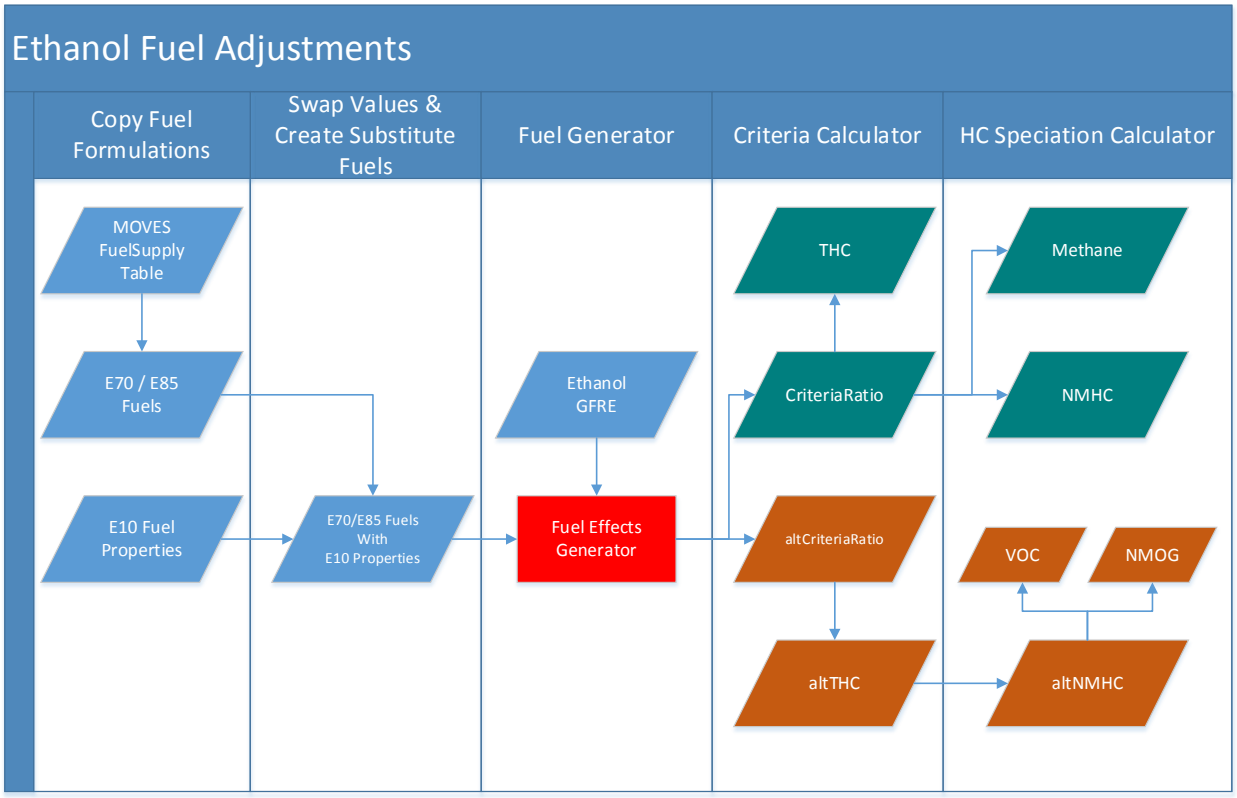
High Ethanol Fuel Adjustments and HC species VOC and NMOG

Based on the available data for MOVES2014, EPA determined that the exhaust emissions of volatile organic compounds (VOC) and non-methane organic gases (NMOG) from high ethanol fuels such as E85 are not statistically significantly different from those from E10 fuels.[3] Within MOVES, VOC and NMOG are derived from non-methane hydrocarbons (NMHC), which is derived from total hydrocarbons (THC). The underlying analysis found that NMHC emissions from high ethanol fuels and E10 fuels are statistically significantly different due to statistically significant differences in methane (CH4) emissions, a situation that would result in differing VOC and NMOG emissions if the standard MOVES fuel effects algorithm were applied. As such, MOVES2014 contains special-case calculations detailed in this section.

High ethanol fuels are fuel subtypes 51 and 52. These fuels require special handling of running exhaust (process 1) and starts exhaust (process 2) fuel effects and calculation of VOC (pollutant 87) and NMOG (pollutant 80). As described earlier, the underlying data indicates that, in model years 2001 and later, these fuel subtypes produce same VOC and NMOG emissions as equivalent E10 fuels, but do not have the same NMHC (pollutant 79) emissions as E10 fuels. Thus, it was necessary to perform a logical “branch” in this chain for high ethanol fuels with a narrow set of pollutants and restricted model years, such that VOC and NMOG values for high ethanol fuels are derived from E10-equivalent THC and CH4.

The overview and detailed steps are as follows.

[3] USEPA (2015). *Fuel Effects on Exhaust Emissions from On-road Vehicles in MOVES2014*. Ann Arbor, MI, Assessment and Standards Division. Office of Transportation and Air Quality. US Environmental Protection Agency. 2015. EPA-420-R-15-002.



Step 1 Copy fuel formulations

High ethanol fuel formulations that are actually in use must be duplicated. New fuel formulation entries are added with identical properties and new fuelFormulationID values. The new formulations will be the only formulations used in subsequent steps. The original formulations are preserved in case future MOVES algorithms require the data.

The FuelSupply table is searched for all high ethanol fuel formulations (fuel subtypes 51 and 52) and their region, fuel year, and month group noted. The matching formulations are duplicated and FuelSupply entries changed to use the new formulations. A new fuel formulation entry is created for each unique combination of fuel formulation, region, fuel year, and month group. This is done to guarantee formulation independence required by the next step.

Step 2 Swap values to create substitute fuels

Some, but not all, of the properties of the new high ethanol fuel formulations must be altered to match E10 fuels. In addition, the original RVP must be preserved while also noting an E10 RVP.

A field named “altRVP” is added to the FuelFormulation table and set to the original RVP.

The E10FuelProperties table provides E10 properties by region, fuel year, and month group. Each new formulation and its associated region, fuel year, and month group is matched with an entry in the E10FuelProperties table and the new formulation’s properties updated. When E10FuelProperties contains a NULL value for a property, the value in the new formulation is left unaltered. This is accomplished using the SQL “coalesce” function. For example, to update the sulfurLevel field in the FuelFormulation table (“ff”) with the value from the E10FuelProperties table (“e10”), the SQL fragment would read:

```
ff.sulfurLevel = coalesce(e10.sulfurLevel, ff.sulfurLevel)
```

It is expected that E10FuelProperties table will contain NULL values in its ETOHVolume, sulfurLevel, and benzeneContent fields. If not, the code ignores the value the user’s fuelformulation table and uses the values in the E10FuelProperties table.

The RVP field is a special case. The new fuel formulation’s RVP is not altered, remaining at the high ethanol RVP. Rather, the formulation’s altRVP is set to the E10 fuel’s RVP. The high ethanol RVP will be used to create THC (1) and the E10 altRVP will be used to create intermediate “altTHC” values. The term intermediate “altTHC” value means the THC that would be produced by the E10 version of the user’s input fuel where RVP is the critical fuel property. This value is not reported to the user but is needed for subsequent calculations in Step 5.

Step 3 Fuel Generator

The FuelEffectsGenerator populates the CriteriaRatio table that is used to calculate THC. For ethanol fuels (fuel type 5), the data comes from equations in the GeneralFuelRatioExpression (“GFRE”) table. In addition to normal THC CriteriaRatio entries, entries for THC equivalent to E10 fuel must be calculated for high ethanol fuel formulations.

Within FuelEffectsGenerator, this is accomplished by duplicating and altering the GFRE entries for fueltypeid = 5. The GFRE table is read into memory as a collection of Java objects. These objects are searched for entries that are applicable for fueltypes, THC pollutant, Running Exhaust (1) or Start Exhaust (2), and model years 2001 and later. Matching entries are duplicated. The duplicates are altered to apply to pollutant “altTHC” (10001), restricted to only fuel subtypes 51 and 52, and to begin no earlier than the 2001 model year. Further, the equations in the duplicates are altered to use “altRVP” instead of “RVP”, thus making the equations generate altTHC based upon E10 RVP not E70/E85 RVP.

The GFRE objects are then processed as normal, populating the CriteriaRatio table. In addition, altTHC entries are used to populate the altCriteriaRatio table. This table has the same schema as CriteriaRatio and is used to ratio altTHC to THC in the next step.

Step 4 Criteria Calculator

Within BaseRateCalculator.sql, THC, but not altTHC, calculations proceed normally based upon data within the CriteriaRatio table. After all emission rates and adjustments to THC have been performed, altTHC is calculated using the ratio between altCriteriaRatio and CriteriaRatio:

altTHC = THC * (altCriteriaRatio.ratio for altTHC) / (CriteriaRatio.ratio for THC)

At this point, altTHC inventory or rates exist for the new high ethanol fuel formulations, running and start exhausts, and model years 2001 and later. Further, these altTHC values were derived using E10 RVP.

Step 5 HC Speciation Calculator

Within HCSpeciationCalculator.sql, methane (5) and NMHC (79) are calculated from THC for all fuel types, as normal using speciation factors that match the fuel type. That is, for E85 emissions, methane and NMHC are calculated using the E85 methane/THC ratios.

Pollutant altNMHC (10079) is calculated from altTHC (10001) for high ethanol fuels using methane ratios for E10 fuels. AltNMHC is needed for the calculation of VOC and NMOG for high ethanol fuels. It is never reported to the user.

Pollutants VOC (87) and NMOG (80) are created from altNMHC for high ethanol fuels, running and start exhausts, and model years 2001 and later, using HC speciation factors and ethanol level for E10 fuels. VOC and NMOG are the only pollutants from E85 that are assumed to be the same as those from E10 fuels.

Step 6 Toxics and other pollutants

The calculation of toxics from E85 that are chained to VOC proceed normally, using the adjustments specific for the high ethanol fuels.

All other pollutants, including CO, NOx, Total Energy, and PM, use the new “alt” fuel formulations that contain the input for high ethanol’s RVP, sulfur, and benzene, combined with E10 values for the remaining fuel properties.

Appendix E

Importer Algorithms

Introduction

This appendix provides details on the algorithms behind some of the MOVES importers. Additional details on MOVES data importers is available in the user guide and user reference.

Retrofit Importer

MOVES supports on-road vehicle retrofits using the Retrofit Data Importer. It allows the user to select a portion of the overall fleet, and reduce its emission rates by a selected fraction.

The Retrofit Data Importer populates the onRoadRetrofit table within a user input, county domain, or project domain database. The onRoadRetrofit table's fields are:

Column	Description
pollutantID	MOVES' numeric pollutant identifier.
processID	MOVES' numeric emission process identifier. For instance, Running Exhaust is 1 and Start Exhaust is 2.
fuelTypeID	MOVES' numeric fuel type identifier. For instance, gasoline is 1, diesel is 2, ethanol is 5.
sourceTypeID	MOVES' numeric source type identifier. For instance, light commercial trucks are 32.
retrofitYearID	The calendar year in which vehicles were retrofitted. All months within a calendar year are affected equally by the retrofit.
beginModelYearID	The lowest model year affected by the retrofit.
endModelYearID	The highest model year affected by the retrofit. This must be less than or equal to the retrofit calendar year.
cumFractionRetrofit	Cumulative fraction of vehicles that have been retrofit (for the pollutant/process, fuel, and source type combination) as of the end of the retrofit calendar year.
retrofitEffectiveFraction	The fraction of emission reduction achieved by a retrofit. It is computed from a non-retrofit emission baseline. All values up to 1.0 (i.e. 100% reduction or a total elimination of emissions) 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 1.0 is not permitted because it implies negative emissions will be generated.

Not all vehicles and fuels can be retrofit. MOVES limits retrofits to only Diesel (2) vehicles and to only the source types: Light Commercial Truck (32), Intercity Bus (41), Transit Bus (42), School Bus (43), Refuse Truck (51), Single Unit Short-haul Truck (52), Single Unit Long-haul Truck (53), Combination Short-haul Truck (61), Combination Long-haul Truck (62).

Retrofit Algorithm

The general form of the retrofit algorithm is shown below.

Retrofit Emissions = [

(Fraction Retrofit1*(1-Retrofit Effectiveness1))

+ (Fraction Retrofit2*(1-Retrofit Effectiveness2)) …

+ … (Fraction Retrofit (i) *(1-Retrofit Effectiveness(i)))

+ (1- Fraction Retrofit1 - Fraction Retrofit2 … - … Fraction Retrofit (i))

] * Base Emissions

Where :

Fraction Retrofit1 + Fraction Retrofit2 … + Fraction Retrofit(i) <= 1.0

I/M Coverage Importer

The I/M Coverage Importer imports data from an Excel spreadsheet or delimited text file. It is completely analogous to the other importers in functionality. It is accessed via the Pre Processing GUI menu using any of the importer options. The Data Importer, County Data Manager, and Project Domain Manager all contain the I/M coverage importer tab.

The importer utilizes Source Use Type as a key variable, because most state and local modelers organize their I/M test programs by Source Use Type (i.e., car, light truck, bus, etc) rather than EPA regulatory classes.

Importing I/M programs improperly has caused many issues with MOVES users in the past. It was previously possible to accidentally mix user input IMCoverage records with MOVES default records. MOVES no longer imports its own default IMCoverage records if any user supplied data exists in an imported IMCoverage table. For instance, if a user imports a single record for passenger cars, MOVES will not use any default data for any model year, pollutant/process, or source type. Therefore, it is very important to export the default IMCoverage information as a basis for user supplied data.

For county domain and project domain user input databases, MOVES will never use its default IMCoverage data.

The IMCoverage table includes a column named UseIMYN which can take on the values of ‘Y’ or ‘N’. When ‘Y’, the I/M program described by the record will be applied. When ‘N’ (or blank or anything other than capital ‘Y’), the record will not be applied. EPA recommends exporting all default IMCoverage and explicitly setting each record to ‘N’ before completing your IMCoverage data for import.

The IMCoverage table’s primary key is polProcessID, stateID, countyID, yearID, sourceTypeID, fuelTypeID, and IMProgramID. That is, each combination of pollutant,

process, state, county, calendar year, source use type, fuel, and I/M program identifier can have only one record in the IMCoverage table. Each combination gets a single model year range, inspection frequency, test standard, compliance factor, and UseIMYN flag. It is important to make the IMProgramID a unique and otherwise unused integer within the IMCoverage table. MOVES enforces I/M rules such that any single pollutant/process on a physical vehicle is only subject to a single I/M benefit.

Appendix F

Stage II Refueling Control Programs

Introduction

Stage II refueling emission control programs are intended to reduce hydrocarbon and associated air toxics emissions by reducing the amount of gasoline vapor that escapes to the atmosphere during refueling. The Refueling Displacement Vapor Losses and the Refueling Spillage Losses processes are affected by Stage II controls, which capture displaced vapors at the fueling station and reduce the fuel spilled due to spitback through the filler neck. No other emission processes are affected by Stage II control programs. The amount of reduction depends on whether the vehicle has an onboard recovery system and the level of uncontrolled emissions. The uncontrolled emissions are calculated from inputs such as fuel RVP, vehicle fuel economy, and temperature parameters. Stage II programs are run by state, local or tribal governments and the fleet-wide effects of these programs will vary depending on the number of locations and the sales volume of stations equipped with Stage II recovery systems and their average state of repair.

The MOVES default database contains information about all of the existing Stage II programs by county based on the parameters used for the 2005 National Emission Inventory (NEI). The estimated effects of Stage II programs can be altered by manually editing the MOVES Stage II tables. Future versions of MOVES may include graphical user interface (GUI) tools to assist in altering Stage II program effects for individual counties. Until these tools are available, this appendix is intended to assist users who need to edit MOVES Stage II default parameters.

Stage II refueling emission control programs can only affect refueling losses that occur during refueling at stations with Stage II controls. Onboard Refueling Vapor Recovery (ORVR) systems on modern vehicles are designed to minimize the refueling losses without Stage II controls by capturing refueling vapors in the vehicle's charcoal canister. The ORVR reductions are already accounted for by MOVES, meaning Stage II programs only affect the remaining refueling losses.

Only the most recent 31 model years of vehicles are simulated during a MOVES run. When the fleet is entirely composed of vehicles subject to ORVR rules, the only value of Stage II controls will be to control refueling vapors from vehicles with damaged ORVR systems. For earlier calendar years, having a Stage II control program will reduce refueling losses, since there are always some vehicles without ORVR controls that will benefit from a Stage II program. The remaining Stage II benefits depend on the program design, including the scope (which stations have the program) and effectiveness (how much of the equipment is operational). MOVES does not account for any effects on the emissions from refueling station gasoline storage tanks when Stage II is used in combination with vehicles equipped with ORVR systems.

County Year Table

The MOVES default database contains a table named CountyYear which contains information about each county for every calendar year. This table contains two fields that determine the Stage II program effects: refuelingVaporProgramAdjustment and refuelingSpillProgramAdjustment.

The refuelingVaporProgramAdjustment is a number between zero and one (1.0) which indicates the fractional reduction in full uncontrolled refueling displacement vapor losses that result from the Stage II recovery program in that county in that calendar year. The refuelingSpillProgramAdjustment field is a number between zero and one (1.0) which indicates the fractional reduction in full uncontrolled refueling fuel spillage losses that result from the Stage II recovery program in that county in that calendar year. A value of zero would indicate that the program had no effect and a value of one would indicate that all uncontrolled vehicle refueling emissions had been eliminated.

Updating Refueling Adjustments Using MySQL Workbench

The simplest way to change an existing Stage II program effect is to use the MySQL Workbench application to open the appropriate table and manually change the existing values. Users should NEVER change values in a default MOVES database provided by EPA, but instead should create and use an input database using the Manage Input Data Sets panel on the MOVES graphical user interface (GUI). Using the Manage Input Data Sets panel, type the name you wish to use for the database which will hold your Stage II estimates into the Database drop-down box. Now click the Create Database button and you will create a database to hold your information. The name of your database will have to meet the criteria for MySQL database names (no spaces, no special characters) and we suggest it include text to help you identify its contents and purpose (for example, StageII_Input).

Once you have an input database, you can open MySQL Workbench to work with the tables. If the Workbench is already open, you may need to refresh the list of databases by clicking on the two circular arrows near the Schemas word. Once you find your database, double-click on the name and that will show you all of the tables it contains. The database will contain empty versions of every table used by MOVES.

The table that contains the Stage II program adjustments is called CountyYear and contains only four fields:

- countyID
- yearID
- refuelingVaporProgramAdjustment
- refuelingSpillProgramAdjustment

When you create an alternate database, the CountyYear table in that database will be empty. You can enter values into each field using the Edit button at the bottom of the panel. Once you click on the Edit button, you can double click on each empty cell and enter a value. Be sure to execute the Apply Changes button to save any alterations.

In order to see the contents of the table, right-click on the County-Year table and choose the “Select Rows – Limit 1000” option. This will create a SQL command and execute it.

```
SELECT * FROM countyyear c;
```

You can execute that command by pressing the lightning bolt button. This will execute the

SQL command and display the contents of the table. There are thousands of lines for every county and year in the nation in the default database. You can edit the SQL command to select only your location. For Washtenaw County, Michigan, for the 2010 calendar year, the appropriate SQL command would be:

```
SELECT * FROM countyyear c where countyID=26161 and yearend=2010;
```

All SQL commands must end in a semi-colon.

Now return to the Manage Input Data Sets panel and select your input database from the drop-down menu. You can add additional description text and then click the add button. When you include this input database in your run specification, MOVES will use your values in preference to any default values in the MOVES database.

The other empty tables may be deleted from your database. Leaving them will not harm your RunSpec (since the tables are empty), but they can be confusing. To delete empty tables, go to the Workbench, double click on the database to show the tables and right click on the table you wish to delete. Choose drop table from the menu and the table will be removed from that database. Multiple tables can be selected by using the CTRL or Shift buttons.

Appendix G

Go Language in MOVES

Installing Go Language for MOVES Use

MOVES releases starting with MOVES2014a perform some calculations using code written in the Go programming language. EPA compiles the Go code and releases this component of MOVES as a compiled executable (.exe), so **MOVES users are not required to have a Go compiler installed.**

MOVES uses Go version 1.3.3. Other versions have not yet been tested. As of summer 2015, no known security risks have been documented with version 1.3.3.

Native installation of Go compiler

For Windows users who want to install Go, download the 32-bit or 64-bit MSI file for Go 1.3.3 from https://golang.org/dl/. MSI files are installers that will ensure all required systems are present for Go. For 32-bit systems, the file is go1.3.3.windows-386.msi. For 64-bit systems, the file is go1.3.3.windows-amd64.msi.

Double-click the downloaded MSI and follow its instructions. Use as many default settings as possible.

For users with 64-bit computer systems who want to compile both 64-bit and 32-bit versions of the calculator, first perform the native 64-bit installation outlined above. Second, download the 32-bit ZIP file go1.3.3.windows-386.zip. Being a ZIP file, an installer is not used and you must manually put the files into the correct location.

Extract the ZIP file to find a directory within named “go”. Rename this “go” directory to “go32” and copy it to your main MOVES installation directory. If MOVES was installed to the default location, this would result in a directory named C:\Users\Public\EPA\MOVES\MOVES2014a\go32\.

Using Ant Go

MOVES provides an Ant command to compile the Go source into all EXEs capable of being made by the local computer. To use it, open a CMD window, CD to your MOVES code directory (by default C:\Users\Public\EPA\MOVES\MOVES2014a), run the setenv.bat command, then run the “ant go” command.

“ant go” will create an EXE compiled in the native format of the computer. This EXE is calc\go\externalcalculatorgo.exe off of the MOVES code directory. This should always occur.

“ant go” will attempt to create a 32-bit EXE using the native compiler and the go32 installation. If the computer system is capable, the EXE is calc\go\externalcalculatorgo32.exe. This EXE will not be created on 64-bit systems that do not have the go32 directory outlined above.

“ant go” will attempt to create a 64-bit EXE using the native compiler. If the computer system is capable, the EXE is calc\go\externalcalculatorgo64.exe. This EXE will not be created on 32-bit systems.

After compiling, update the calculatorApplicationPath entry of the WorkerConfiguration.txt configuration file. The entry should reference an EXE that is compatible with the computer system and that was recompiled with the latest code changes.

Obtaining Detailed Debug Information for the Go Calculator

Detailed information for the inputs and outputs of the external calculator can be obtained through modification of the Worker Java code. This action requires use of the Java JDK which is not normally installed with MOVES. Further, this action accumulates large amounts of data on a worker computer's hard disk and should only be done for local workers on runs with few bundles (ideally just a single bundle).

To enable the detailed output, first edit the file C:\Users\Public\EPA\MOVES\MOVES2014a\gov\epa\otaq\moves\worker\framework\RemoteEmissionsCalculator.java (assuming a default installation location). Change these lines from:

```
/** Control cleanup of Nonroad and other other temporary files, true to keep the files **/

boolean isTest = false;
```

to:

```
/** Control cleanup of Nonroad and other other temporary files, true to keep the files **/

boolean isTest = true;
```

Close your text editor to ensure the file is saved.

Use the "ant compileall" command to ensure all files, including the one just changed, are recompiled.

To undo the changes, change the line back to its original text and use the "ant compileall" command again.

Once changed, each bundle will leave a directory in \WorkerFolder\ of the main MOVES installation directory, such as WorkerFolder, WorkerFolder25, etc. Within this directory will be a file named ExternalCalculatorProcessOutput.txt, containing notes on the execution. For example:

```
ExternalCalculatorGo starting...
Reading the global configuration values...
Done reading the global configuration values.
Needs module NRAirToxicsCalculator
Needs module NRHCSpeciationCalculator
Needs module nrNonHAPTOG
Needs module outputfulldetail
Loaded 1 fuel types, 16 fuel subtypes, 6 fuelformulations
Loaded 1 fuel supply entries, 3 Nonroad fuel supply entries, maxEntriesPerFuel = 2
needsDetailOutput true
needsHCSpeciation false
```

```
needsNRHCSpeciation true
needsNRAirToxics true
needsNRNonHAPTOG true
Sizes: NRATRatio: 1204, NRATRatioProcesses: 1
Sizes: NRPAHGasRatio: 222, NRPAHGasRatioProcesses: 1
Sizes: NRPAHParticleRatio: 222, NRPAHParticleRatioProcesses: 1
Sizes: NRDioxinEmissionRate: 222, NRDioxinEmissionRateProcesses: 1
Sizes: NRMetalEmissionRate: 222, NRMetalEmissionRateProcesses: 1
Sizes: NRIntegratedSpecies: 15
NRHCSpeciation finished reading setup files.
NRAirToxics finished reading setup files.
Memory: current=3895976
Started writing detail output to:  newmovesworkeroutput_detail
Started writing output to:  newmovesworkeroutput
Memory: current=4542096
MOVESWorkerOutput had 24814 lines, parsed as 249 sets.
End conditions detected.
Considered 249 MWOBlocks for writing.
Considered 827474 FuelBlocks for writing.
Considered 1338788 MWOEmissions for writing.
Wrote 802660 lines to the output file.
Wrote 1338788 lines to the detail output file.
ExternalCalculatorGo done with 998 events.
maxMWOInternalCount=80, maxMWOWriteCount=242
MWOReadEnds: 1,MWOBlockBegins: 249,MWOBlockEnds: 249,MWOWriteBegins:
249,MWOWriteEnds: 249
Memory: max=189176272, current=2716272
externalcalculatorgo took this amount of time: 13.8428473s
```

This output includes key facts such as the duration (13.84 seconds), the amount of RAM memory needed (189MB), and the number of output records (802,660).

Also in this directory is the loaddetails.sql file. This file contains SQL commands that can be run within the movesworker database. These commands will read the detailed external calculator output into a MySQL table then perform basic aggregation. The detailed information is provided at the fuel formulation level, making it possible to verify benchmarks that depend upon fuel properties.

The detailed output by fuel formulation is already split by market share and will be in the ExtMOVESWorkerOutputDetailSum table. The ExtMOVESWorkerOutputDetail table may contain multiple entries for each pollutant as occurs when calculating NonHAPTog (pollutant 88).