Execution Service (version 5.0)

All TRANSIMS programs are derived from the Execution Service class. This class:

1. Manages the command line interface;
2. Processes the command line flags;
3. Creates the program report file;
4. Reads and processes the configuration file;
5. Reads and processes the control file and control keys;
6. Manages the report and print interface;
7. Manages the message, problem and error services;
8. Generates help messages and the XML interface;
9. Manages partition processing data;
10. Controls multi-threading and MPI options;
11. Processes a global set of control keys; and
12. Executes the program.

Command Line Syntax

TRANSIMS programs are console-based programs that run in a command window on either 32 or 64 bit Windows or Linux operating systems. The programs are executed from the command prompt or through a batch file using one of the following syntax statements:

Program [-flag] [control_file]

Program [-flag] [control_file] [partition]

Program [-flag] [control_file] [parameter]

The flag parameters are optional. Any combination of the following flag parameters can be included on the command line:

-H[elp]	-C[ontrol]	-K[eyCheck]
-P[ause]	-N[oPause]	-Q[uiet]
-D[etail]	-X[ML]	-R[eport]

–Help

The help flag writes the program syntax, control keys, and report options to the screen to provide a quick reminder of the control options and the text used to reference various keys and reports. The help flag can be specified with or without a control file. If a control file is not provided, the program writes the help messages to the screen and waits for the user to press the enter key to exit.

A typical help file is shown below. The control key section of a help message shows the key name and identifies if the key is required or optional, the data type of the key value, and the default value for the key.

```
****************************************
|                                      |
|         LineSum - Version 5.0.0      |
|  Copyright 2012 by TRANSIMS Open-Source  |
|         Wed May 16 19:49:50 2012     |
|                                      |
****************************************

Syntax is LineSum [-flag] [control_file]

Optional Flags:
  -H[elp]     = show program syntax and control keys
  -C[ontrol]  = create/update a default control file
  -K[eyCheck] = list unrecognized control file keys
  -P[ause]    = pause before exiting
  -N[oPause]  = never pause before exiting
  -Q[uiet]    = execute without screen messages
  -D[etail]   = execute with detailed status messages
  -X[ML]      = write an XML file with control keys
  -R[eport]   = write control keys and report names

Control File Keys:
  TITLE                          Opt.Text
  REPORT_FILE                    Opt.New
  REPORT_FLAG                    Opt.Bool = FALSE
  PROJECT_DIRECTORY              Opt.Path
  DEFAULT_FILE_FORMAT            Opt.Text = TAB_DELIMITED
  PEAK_RIDERSHIP_FILE_#          Opt.File
  PEAK_RIDERSHIP_FORMAT_#        Opt.Text = DBASE
  OFFPEAK_RIDERSHIP_FILE_#       Opt.File
  OFFPEAK_RIDERSHIP_FORMAT_#     Opt.Text = DBASE
  NEW_PEAK_RIDERSHIP_FILE        Opt.New
  NEW_PEAK_RIDERSHIP_FORMAT      Opt.Text = DBASE
  NEW_OFFPEAK_RIDERSHIP_FILE     Opt.New
  NEW_OFFPEAK_RIDERSHIP_FORMAT   Opt.Text = DBASE
  STOP_NAME_FILE                 Opt.File
  STOP_NAME_FORMAT               Opt.Text = TAB_DELIMITED
  LINE_REPORT_TITLE_#            Opt.Text = Line Report
  LINE_REPORT_LINES_#            Opt.List = ALL
  LINE_REPORT_MODES_#            Opt.List = ALL
  LINE_REPORT_ALL_NODES_#        Opt.Bool = False
  LINK_REPORT_TITLE_#            Opt.Text = Link Report
  LINK_REPORT_LINKS_#            Opt.List
  LINK_REPORT_MODES_#            Opt.List = ALL
  LINK_REPORT_LINES_#            Opt.List = ALL
  LINK_REPORT_ONEWAY_#           Opt.Bool = False
  NEW_LINK_REPORT_FILE_#         Opt.New
  ACCESS_REPORT_TITLE_#          Opt.Text = Access Report
  ACCESS_REPORT_STOPS_#          Opt.List = ALL
  ACCESS_REPORT_MODES_#          Opt.List = ALL
  ACCESS_REPORT_DETAILS_#        Opt.Bool = False
  NEW_ACCESS_REPORT_FILE_#       Opt.New
  STOP_REPORT_TITLE_#            Opt.Text = Stop Report
  STOP_REPORT_STOPS_#            Opt.List
  STOP_REPORT_MODES_#            Opt.List = ALL
  STOP_REPORT_LINES_#            Opt.List = ALL
  NEW_LINK_RIDER_FILE_#          Opt.New
  NEW_LINK_RIDER_FORMAT_#        Opt.Text = TAB_DELIMITED
  LINK_RIDER_MODES_#             Opt.List = ALL
  LINK_RIDER_LINES_#             Opt.List = ALL
  LINK_RIDER_PEAK_HOURS_#        Opt.Dec. = 6.0
  LINK_RIDER_PEAK_FACTOR_#       Opt.Dec. = 1.0
  LINK_RIDER_PEAK_CAPACITY_#     Opt.Dec. = 1.0
  LINK_RIDER_OFFPEAK_HOURS_#     Opt.Dec. = 10.0
  LINK_RIDER_XY_FILE_#           Opt.File
  LINK_RIDER_XY_FORMAT_#         Opt.Text = TAB_DELIMITED
  BASE_ROUTE_FILE_#              Opt.File
  BASE_ROUTE_FORMAT_#            Opt.Text = DBASE
  ALTERNATIVE_ROUTE_FILE_#       Opt.File
  ALTERNATIVE_ROUTE_FORMAT_#     Opt.Text = DBASE
  LINESUM_REPORT_#               Opt.Text

Report Options:
  LINE_REPORT
  LINK_REPORT
  ACCESS_REPORT
  STOP_REPORT
  TOTAL_REPORT
  DIFFERENCE_REPORT

Press Enter to Continue
```

–Control

The control flag creates a control file template using the default values or updates an existing control file by adding keys that were not previously included. If a control file name is not provided, the program creates a file using the program name with the ".ctl" extension. If a control file name is include, the file is read, the default values are replaced with the key values from the control file, the program saves the updated control file and proceeds to execute the program. If the user wants to update the control file without executing the program, the flag –CX can be used to exit the program after the control file is updated.

A typical control template is shown below. It lists the keys in service groups, includes the default value if one exists, and adds a comment message that lists the acceptable value range, value options, or syntax examples.

```
TITLE                              LineSum Default Control Keys
REPORT_FILE                                    //---- [report_directory]filename[_partition][.prn]
REPORT_FLAG                        FALSE       //---- TRUE/FALSE, YES/NO, 1/0, T/F, Y/N
PROJECT_DIRECTORY
DEFAULT_FILE_FORMAT                TAB_DELIMITED    //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…

#---- LineSum Control Keys ----

PEAK_RIDERSHIP_FILE_1                          //---- [project_directory]filename
PEAK_RIDERSHIP_FORMAT_1            DBASE       //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
OFFPEAK_RIDERSHIP_FILE_1                       //---- [project_directory]filename
OFFPEAK_RIDERSHIP_FORMAT_1         DBASE       //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
NEW_PEAK_RIDERSHIP_FILE                        //---- [project_directory]filename
NEW_PEAK_RIDERSHIP_FORMAT          DBASE       //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
NEW_OFFPEAK_RIDERSHIP_FILE                     //---- [project_directory]filename
NEW_OFFPEAK_RIDERSHIP_FORMAT       DBASE       //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
NEW_TOTAL_RIDERSHIP_FILE                       //---- [project_directory]filename
NEW_TOTAL_RIDERSHIP_FORMAT         DBASE       //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
STOP_NAME_FILE                                 //---- [project_directory]filename
STOP_NAME_FORMAT                   TAB_DELIMITED    //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
LINE_REPORT_TITLE_1                Line Report      //---- Report Title
LINE_REPORT_LINES_1                ALL              //---- e.g., LINE1, LINE2, LINE1..LINE10, AB..AB|
LINE_REPORT_MODES_1                ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
LINE_REPORT_ALL_NODES_1            False            //---- TRUE/FALSE, YES/NO, 1/0, T/F, Y/N
LINK_REPORT_TITLE_1                Link Report      //---- Report Title
LINK_REPORT_LINKS_1                                 //---- e.g., 100-200, 300-400-500
LINK_REPORT_MODES_1                ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
LINK_REPORT_LINES_1                ALL              //---- e.g., LINE1, LINE2, LINE1..LINE10, AB..AB|
LINK_REPORT_ONEWAY_1               False            //---- TRUE/FALSE, YES/NO, 1/0, T/F, Y/N
ACCESS_REPORT_TITLE_1              Access Report    //---- Report Title
ACCESS_REPORT_STOPS_1              ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
ACCESS_REPORT_MODES_1              ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
ACCESS_REPORT_DETAILS_1            False            //---- TRUE/FALSE/MODE, YES/NO/MODE, 1/0/2, T/F/M,…
NEW_ACCESS_REPORT_FILE_1                           //---- [project_directory]filename
NEW_ACCESS_REPORT_FORMAT_1         TAB_DELIMITED    //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
STOP_REPORT_TITLE_1                Stop Report      //---- Report Title
STOP_REPORT_STOPS_1                                 //---- e.g., 1, 2, 4..10, 100..200, 300
STOP_REPORT_MODES_1                ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
STOP_REPORT_LINES_1                ALL              //---- e.g., LINE1, LINE2, LINE1..LINE10, AB..AB|
TOTAL_REPORT_TITLE_1               Total Report     //---- Report Title
TOTAL_REPORT_LINES_1               ALL              //---- e.g., LINE1, LINE2, LINE1..LINE10
NEW_TOTAL_REPORT_FILE_1                            //---- [project_directory]filename
NEW_LINK_RIDER_FILE_1                              //---- [project_directory]filename
NEW_LINK_RIDER_FORMAT_1            TAB_DELIMITED    //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
LINK_RIDER_MODES_1                 ALL              //---- e.g., 1, 2, 4..10, 100..200, 300
LINK_RIDER_LINES_1                 ALL              //---- e.g., LINE1, LINE2, LINE1..LINE10, AB..AB|
LINK_RIDER_PEAK_HOURS_1            6.0              //---- 1.0..10.0
LINK_RIDER_PEAK_FACTOR_1           1.0              //---- 1.0..10.0
LINK_RIDER_PEAK_CAPACITY_1         1.0              //---- 1.0..1000.0
LINK_RIDER_OFFPEAK_HOURS_1         10.0             //---- 1.0..20.0
LINK_RIDER_XY_FILE_1                               //---- [project_directory]filename
LINK_RIDER_XY_FORMAT_1             TAB_DELIMITED    //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
BASE_ROUTE_FILE_1                                  //---- [project_directory]filename
BASE_ROUTE_FORMAT_1                DBASE            //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
ALTERNATIVE_ROUTE_FILE_1                           //---- [project_directory]filename
ALTERNATIVE_ROUTE_FORMAT_1         DBASE            //---- TEXT, BINARY, FIXED_COLUMN, COMMA_DELIMITED,…
LINESUM_REPORT_1                                   //---- program report name
```

–KeyCheck

The key-check flag generates a list of warning messages for any key included in the control file that is not recognized by the program. By default, a control file can include keys that are used by multiple programs. Some of the keys may be unique to one program and ignored by other programs. Based on this behavior, a key that is intended for a given program, but is defined improperly or misspelled will be ignored. The key check option enables the user to check if their control keys are specified correctly or remove keys that are not used by a particular program.

–Pause

If the pause flag is included on the command line, the program with require the user to press the enter key before the program exits and returns control to the operating system or the batch file. This can be a useful way of reviewing the on-screen messages before the operating system deletes the command window.

–NoPause

If the no-pause flag is included, the program will always exit without waiting for user intervention. Normally, when an error is detected, the program writes an error message and waits for the user to press the enter key before exiting the program. The no-pause option permits the program to automatically exit even when an error is detected. In this case, a batch procedure should check for an error return code from the program and take appropriate action.

–Quiet

The quiet flag suppresses output to the computer screen unless or until an error message is encountered. It is intended for applications that run on multiple remote processors or for batch applications that capture screen output to a log file. In this case the log file will include the batch commands but not the long list of status messages that are typically written to the screen. The log file will include the banner page and error message when the program terminates with an error.

–Detail

The detail flag is intended for program applications that wish to save screen output to a log file. When a program is processing a given file or making calculations, it writes a progress message to the screen once every second to provide the user with general information about the processing rate. Normally the progress counter overwrites the previous value each second. When screen output is written to a file, the backspace commands used to overwrite the counter generates undesirable output. In this case, the detail flag can be used to write each progress counter on a separate line.

–XML

The XML flag writes the same type of information as the control flag in an XML file format. If a control file is provided, the XML file will include the key values specified in the control file plus the default values for other keys. The output filename will be the name of the program with the ".xml" extension or the name of the input control file with the file extension replaced with ".xml".

A small section of a sample XML file is shown below. Notice how the LEVEL_KEYS section is coded. In this case the KEY_CODE 806 defines the syntax for a generic key group. The input control file, however, defined two instances of this key. These instances are listed in the LEVEL_KEYS section along with the values provided by the control file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<TRANSIMS>
<PROGRAM NAME="Router" VERSION="5.0.46" COPYRIGHT="2012 by TRANSIMS Open-Source" PARTITIONS="TRUE" />
<CONTROL_KEYS>
<KEY CODE="200" NAME="TITLE" REQUIRED="false" TYPE="Text" VALUE="Router Test" />
<KEY CODE="204" NAME="PROJECT_DIRECTORY" REQUIRED="false" TYPE="Path" VALUE="../" />
<KEY CODE="213" NAME="MAX_WARNING_EXIT_FLAG" REQUIRED="false" TYPE="Bool" DEFAULT="TRUE" RANGE="TRUE/FALSE, YES/NO, 1/0, T/F, Y/N" />
<KEY CODE="214" NAME="MAX_PROBLEM_COUNT" REQUIRED="false" TYPE="Integer" DEFAULT="0" RANGE=">= 0" />
<KEY CODE="215" NAME="NUMBER_OF_THREADS" REQUIRED="false" TYPE="Integer" DEFAULT="1" RANGE="1..64" VALUE="30" />
<KEY CODE="300" NAME="NODE_FILE" REQUIRED="true" TYPE="Net" RANGE="[project_directory]filename" VALUE="network/node.txt" />
<KEY CODE="400" NAME="NODE_FORMAT" REQUIRED="false" TYPE="Text" DEFAULT="TAB_DELIMITED" RANGE="TEXT, BINARY, FIXED_COLUMN,
COMMA_DELIMITED, SPACE_DELIMITED, TAB_DELIMITED, CSV_DELIMITED, DBASE, ARCVIEW, SQLITE3, VERSION3" HELP="2" />
<KEY CODE="303" NAME="LINK_FILE" REQUIRED="true" TYPE="Net" RANGE="[project_directory]filename" VALUE="network/link.txt" />
<KEY CODE="403" NAME="LINK_FORMAT" REQUIRED="false" TYPE="Text" DEFAULT="TAB_DELIMITED" RANGE="TEXT, BINARY, FIXED_COLUMN,
COMMA_DELIMITED, SPACE_DELIMITED, TAB_DELIMITED, CSV_DELIMITED, DBASE, ARCVIEW, SQLITE3, VERSION3" HELP="2" />
<KEY CODE="805" NAME="LINK_DELAY_FLOW_FACTOR" REQUIRED="false" TYPE="Decimal" DEFAULT="1.0" RANGE="1..100000" VALUE="1.0" />
<KEY CODE="806" NAME="EQUATION_PARAMETERS_#" REQUIRED="false" TYPE="List" DEFAULT="BPR, 0.15, 4.0, 0.75" RANGE="BPR, 0.15, 4.0, 0.75" >
<LEVEL_KEYS>
<LEVEL NAME="EQUATION_PARAMETERS_1" VALUE="BPR, 0.2, 8.5, 1.0" />
<LEVEL NAME="EQUATION_PARAMETERS_2" VALUE="BPR, 0.25, 9.0, 0.75" />
</LEVEL_KEYS>
</KEY>
<KEY CODE="1" NAME="UPDATE_PLAN_RECORDS" REQUIRED="false" TYPE="Bool" DEFAULT="FALSE" RANGE="TRUE/FALSE, YES/NO, 1/0, T/F, Y/N" />
<KEY CODE="2" NAME="REROUTE_FROM_TIME_POINT" REQUIRED="false" TYPE="Time" DEFAULT="0:00" />
<KEY CODE="11" NAME="NEW_TRIP_CONVERGENCE_FILE" REQUIRED="false" TYPE="NewFile" RANGE="[project_directory]filename"
VALUE="demand/trip_gap.txt" />
<KEY CODE="216" NAME="ROUTER_REPORT_#" REQUIRED="false" TYPE="Text" RANGE="program report name" HELP="1" >
<LEVEL_KEYS>
<LEVEL NAME="ROUTER_REPORT_1" VALUE="LINK_GAP_REPORT" />
<LEVEL NAME="ROUTER_REPORT_2" VALUE="TRIP_GAP_REPORT" />
<LEVEL NAME="ROUTER_REPORT_3" VALUE="ITERATION_PROBLEMS" />
</LEVEL_KEYS>
</KEY>
```

If the XML flag is specified as –XH, the output file will include additional help messages defined in the TRANSIMS help file. The path to the help file is set using the operating system environment variable TRANSIMS_HELP_FILE. An example of the help message section of an XML file is shown below.

```
<HELP_CODES>
<HELP CODE="1" >
<LINE NUM="1" TEXT="  Reports are requested through a nested key with syntax:" />
<LINE NUM="2" TEXT="  PROGRAM_REPORT_# = REPORT_NAME" />
<LINE NUM="3" TEXT="" />
<LINE NUM="4" TEXT="  For Example:" />
<LINE NUM="5" TEXT="  LINKSUM_REPORT_1 = TOP_100_LINK_FLOWS" />
<LINE NUM="6" TEXT="  LINKSUM_REPORT_2 = TOP_100_LANE_FLOWS" />
<LINE NUM="7" TEXT="  LINKSUM_REPORT_3 = LINK_VOLUME_GREATER_THAN_1.3" />
<LINE NUM="8" TEXT="  LINKSUM_REPORT_4 = LINK_VOLUME_GREATER_THAN_2.5" />
<LINE NUM="9" TEXT="" />
<LINE NUM="10" TEXT=" Note that the last two reports request the same report with different filtering crite…
<LINE NUM="11" TEXT=" This report is defined with a wildcard code (LINK_VOLUME_GREATER_THEN_#) that enables…
<LINE NUM="12" TEXT=" to specify a filter parameter. Multiple reports of this type can be generated." />
<LINE NUM="13" TEXT="" />
<LINE NUM="14" TEXT=" In most cases, the reports are printed in the report file in the report key order." />
</HELP>
<HELP CODE="2" >
<LINE NUM="1" TEXT="  Format keys are used to define how data files are read or created." />
<LINE NUM="2" TEXT="  If a format key is not provided, the value of the DEFAULT_FILE_FORMAT key is used." />
<LINE NUM="3" TEXT="  The default value of the DEFAULT_FILE_FORMAT key is TAB_DELIMITED. " />
<LINE NUM="4" TEXT="" />
<LINE NUM="5" TEXT="  In most cases, TRANSIMS constructs a *.def file for each file it creates. " />
<LINE NUM="6" TEXT="" />
<LINE NUM="7" TEXT="  The *.def file enables the software to identify the file format and field names for …
<LINE NUM="8" TEXT="  If a *.def file is available, the format key is ignored." />
<LINE NUM="9" TEXT="  If a *.def file is not available, the format key tells the program how to build a *….
<LINE NUM="10" TEXT=" For Delimited files, the software reads the header line     and the first 100 records…
<LINE NUM="11" TEXT=" The software cannot build *.def files for Binary and Fixed Columm files." />
<LINE NUM="12" TEXT="" />
<LINE NUM="13" TEXT=" The VERSION3 format option is provided for backwards compatibility.  In many cases,…
<LINE NUM="14" TEXT=" read a Version 3 or Version 4 file without modification.  The Version 5 software …
</HELP_CODES>
</TRANSIMS>
```

–Report

The report flag writes the program syntax, control keys and report information to a tab delimited file to assist with program documentation. The output file includes the program name with the ".doc" extension. If a control file is provided, the document file does not include any information from the control file.

Selected rows of a sample document file read into an Excel spreadsheet are shown below.

LineSum (5.0.1)					
Syntax:					
	LineSum [-flag] [control_file]				
Optional Flags:					
	-H[elp]	show program syntax and control keys			
	-C[ontrol]	create/update a default control file			
	-K[eyCheck]	list unrecognized control file keys			
	-P[ause]	pause before exiting			
	-N[oPause]	never pause before exiting			
	-Q[uiet]	execute without screen messages			
	-D[etail]	execute with detailed status messages			
	-X[ML]	write an XML file with control keys			
	-R[eport]	write control keys and report names			
Execution Service Keys:					
	TITLE	Optional	Text		
	REPORT_DIRECTORY	Optional	Directory		
	REPORT_FILE	Optional	Output File		[report_directory]filename[_partition][.prn]
	REPORT_FLAG	Optional	Boolean	FALSE	TRUE/FALSE, YES/NO, 1/0, T/F, Y/N
	PROJECT_DIRECTORY	Optional	Directory		
	DEFAULT_FILE_FORMAT	Optional	Text	TAB_DELIMITED	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
LineSum Control Keys:					
	PEAK_RIDERSHIP_FILE_#	Optional	Input File		[project_directory]filename
	PEAK_RIDERSHIP_FORMAT_#	Optional	Text	DBASE	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	OFFPEAK_RIDERSHIP_FILE_#	Optional	Input File		[project_directory]filename
	OFFPEAK_RIDERSHIP_FORMAT_#	Optional	Text	DBASE	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	NEW_PEAK_RIDERSHIP_FILE	Optional	Output File		[project_directory]filename
	NEW_PEAK_RIDERSHIP_FORMAT	Optional	Text	DBASE	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	NEW_OFFPEAK_RIDERSHIP_FILE	Optional	Output File		[project_directory]filename
	NEW_OFFPEAK_RIDERSHIP_FORMAT	Optional	Text	DBASE	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	NEW_TOTAL_RIDERSHIP_FILE	Optional	Output File		[project_directory]filename
	NEW_TOTAL_RIDERSHIP_FORMAT	Optional	Text	DBASE	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	STOP_NAME_FILE	Optional	Input File		[project_directory]filename
	STOP_NAME_FORMAT	Optional	Text	TAB_DELIMITED	TEXT, BINARY, FIXED_COLUMN, COMMA_DELI
	LINE_REPORT_TITLE_#	Optional	Text	Line Report	Report Title
	LINE_REPORT_LINES_#	Optional	List	ALL	e.g., LINE1, LINE2, LINE1..LINE10, AB..AB|
	LINE_REPORT_MODES_#	Optional	List	ALL	e.g., 1, 2, 4..10, 100..200, 300
	LINE_REPORT_ALL_NODES_#	Optional	Boolean	FALSE	TRUE/FALSE, YES/NO, 1/0, T/F, Y/N
	LINESUM_REPORT_#	Optional	Text		program report name
Report Options:					
	LINE_REPORT				
	LINK_REPORT				
	ACCESS_REPORT				
	STOP_REPORT				
	TOTAL_REPORT				
	DIFFERENCE_REPORT				

Control File

The control_file field on the command line is the directory path and file name of a text file that contains the control strings expected by the program. If a file name is not provided, the program will prompt the user to enter a file name. The program automatically creates a printout file based on the control file name. If the file name includes an extension (e.g., ".ctl"), the extension is removed and ".prn" is added. The printout file will be created in the current working directory and will overwrite an existing file with the same name.

If the program command syntax includes the partition option, the program can be instructed to process a subset of file partitions by specifying a partition number or partition range after the control file name. For example, the Router can execute a subset of partitions using a command line like:

```
Router.exe  Router.ctl  10

Router.exe  Router.ctl  0..4
```

The first command generates plans for the households assigned to partition 10. The second command generates plans for households assigned to partitions 0 through 4. In these cases, the printout file generated by the program includes the partition number or range in the file name:

```
Router_10.prn

Router_0-4.prn
```

If the program command syntax includes the parameter option, the printout file will include the parameter information. For example, the command:

```
RunSetup.exe  TripModel.ctl  2010
```

will create the printout file:

```
TripModel_2010.prn
```

Program Controls

Program control parameters are defined using a control key followed by a string or number. The control parameters can be specified in any order and are not case sensitive. If a given key is defined more than once, the last instance of the key is used. A given program can define a given key as required or optional. If the key is required and not included in the control file, an error message is written and the program is terminated. The program may also assign a default value and value range to each key. If the default value is appropriate, the key does not need to be included in the file. If the user-provided value is out of range, an error message is written and the program is terminated.

Each key includes a value type that defines how the key is processed. Value types include integer or decimal numbers, text strings, Boolean (true/false) flags, time strings, input and output files, directory paths, and lists. A list key includes one or more values or value ranges. TRANSIMS defines a value range using two periods (e.g., 100..200).

Some keys may also permit multiple instances or multiple nesting levels. In these cases, the key name is followed by a number that defines a particular instance. Several examples of multi-level keys are shown in the –Report section above. The PEAK_RIDERSHIP_FILE_# key implies that multiple peak ridership files can

be processed by the program. The control file identifies multiple files by replacing the "#" with a number. For example:

```
PEAK_RIDERSHIP_FILE_1        Myfile1.dbf
PEAK_RIDERSHIP_FILE_2        Myfile2.dbf
```

If the application does not require multiple files, the _# extension is not required. In other words, the control file could include the following key:

```
PEAK_RIDERSHIP_FILE          Myfile.dbf
```

In addition to defining multiple instances of a given key, the level code is used to define groups of keys. The –Report section above also includes an example of a key group. The four keys:

```
LINE_REPORT_TITLE_#
LINE_REPORT_LINES_#
LINE_REPORT_MODES_#
LINE_REPORT_ALL_NODES_#
```

function as a group. The instances of each key that share the same level code are processed together. In this case a line report has four control parameters and multiple line reports with difference selection criteria can be generated by the program.

Note that comment lines or extraneous keys can be included in a control file. They will be ignored by the program. Comment lines or messages are identified by text strings that start with one of the following character sequences:

```
##      #-      #*      //      /-      /*      ;;      ;-      ;*
```

All test in the record after the comment characters is ignored. For example:

```
LINE_REPORT_MODES_2          1..3     //---- local bus, express bus, and Metrorail ----
##LINE_REPORT_ALL_NODES_2             TRUE
```

The first line shows a comment message after the key value. The second line shows a convenient method of disabling a given key.

Execution Service Keys

The execution service manages a number of control keys that are common to all programs. These keys are described below:

TITLE (optional, text)

Any text string can be used on this line. This text is printed on the top of each output page.

REPORT_DIRECTORY (optional, path)

If the report directory key is specified, it is added to the report file name specified by the Report File key or the default report file name derived from the control file name. By default, the report file is created in the same directory as the control file. If the control file name includes path information, the path string is removed and replaced by the report directory string.

REPORT_FILE (optional, output file)

If a report file name is not provided, the program automatically creates a report file name based on the input control file name plus the partition number. The report file will overwrite an existing file with the same name if the Report Flag key is False or not specified.

REPORT_FLAG (optional, flag, FALSE)

If the report flag key is YES or TRUE, the report file or default printout file will be opened in "Append" mode rather than "Create" mode. This permits the user to consolidate the output of several programs into a single report file.

PROJECT_DIRECTORY (optional, text)

If the project directory key is specified, it is added to all file names referenced by the program. If it is not specified, all file names should fully specify the file path relative to the current directory.

DEFAULT_FILE_FORMAT (optional, text, TAB_DELIMITED)

This key can be used to change the default file format. By default, TRANSIMS creates new files in TAB_DELIMITED format. Other options include BINARY, DBASE, COMMA_DELIMITED, SPACE_DELIMITED, FIXED_COLUMN and SQLITE3.

TIME_OF_DAY_FORMAT (optional, text, DAY_TIME)

The time of day format defines how the time data are written to the output files and reports. The default format will display values in DAY_TIME format (e.g., 0:00:00 to 1@3:00:00 refers to midnight to 3:00 AM the next day). The format options include SECONDS, MINUTES, HOURS, HOUR_CLOCK (e.g., 0:00 to 27:00), and TIME_CODE. Time codes combine a day code with an hour clock (e.g. TUE08:00). Day code options include SUN, MON, TUE, WED, THU, FRI, SAT, WKE, WKD, and ALL.

MODEL_START_TIME (optional, time, 0:00)

The model start time defines the time-of-day at the beginning of the modeling process. The default value is 0:00 or midnight. Many activity-based models consider the start of the day to be 3:00 AM when most people are at home in bed.

MODEL_END_TIME (optional, time, 24:00)

The model end time defines the time-of-day at the end of the modeling process. The default value is 24:00. Since there tends to be a significant number of trips that start near midnight and may take some time to reach their destination, the model end time is often increased to a value such as 27:00 to ensure that all trips are completed. Other applications may wish to model travel over multiple days (e.g., hurricane evacuation studies). In this case, this control key can be set to 48:00 or 72:00.

MODEL_TIME_INCREMENT (optional, time, 15 minutes, 2..240 minutes)

The model time increment defines the standard time period resolution used for dynamic assignments. The default value is 15 minutes. The combination of time increments and model start and end times established the number of time periods used for defining link travel times and speeds. For example, the default parameters create 96 different travel time values for each link.

UNITS_OF_MEASURE (optional, text, METRIC, ENGLISH/METRIC)

The default distance and speed units included in data files or control keys are assumed to be in METRIC units. This key can be used to specify the units of measure as ENGLISH or METRIC. If a particular key value includes data units, the program will automatically convert the value to the specified units of measure. The standard data files created by the TRANSIMS Version 5 software identify the units associated with each data field in the definition file (*.def).

RANDOM_NUMBER_SEED (optional, integer, 0, >= 0)

The random number seed key specifies the starting point for a list of random numbers. Any positive integer can be specified. If the value is zero or if no key is provided, the program uses the computer clock to set the random number seed. The selected seed value is written to the printout report to enable the user to re-run the model using the same random number sequence.

MAX_WARNING_MESSAGES (optional, integer, 100000, >= 0)

When the program generates a warning message, a counter is incremented and the total number of warning messages is reported and a warning return coded (2) is set at the end of the execution. By default the program prints up to 100,000 warning messages to the printout file. If more than 100,000 warning messages are sent, the program stops printing additional messages to the file or terminates the program with an error message based on the MAX_WARNING_EXIT_FLAG. This parameter enables the user to modify the default warning limit.

MAX_WARNING_EXIT_FLAG (optional, flag, true)

If the maximum number of warning messages is exceeded, this flag directs the program in what to do. If the flag is TRUE (the default), the program is terminated with an error message about the warning messages. If the flag is FALSE, the program continues execution, but no additional warning messages are sent to the screen or written to the printout file. The warning message counter continues to count the messages and reports the total at the end of the execution.

MAX_PROBLEM_COUNT (optional, integer, 0, >= 0)

The maximum problem count defines the number of modeling problems that are permitted before the problem terminates execution. The default value of zero disables this feature.

NUMBER_OF_THREADS (optional, integer, 1, 1..64)

This parameter is only used for programs where multi-thread processing is enables. TRANSIMS uses the Boost library to implemented processing threads. The software can be compiled with or without this library. If the library is included and the program is thread enabled, the number of threads key instructs the program on the number of CPUs that will be used for parallel data processing. The key value can range from 1 to 64. The user can disable the multi-thread processing by setting this key to 1. If the key value is greater than one and the particular program or compiled executable does not support multi-threading, a warning message is written to the screen.

Configuration File

In most TRANSIMS applications there are a significant number of keys that are common to all programs. Many of the Execution Service keys fall into this category. They tend to be global keys that define the default behavior of the model. If the modeler wishes to set these keys once and use them in all model

applications, a TRANSIMS configuration file can be created. A configuration file is exactly like any other control file and can include any number of control keys and key values. Each TRANSIMS program looks for a configuration file using the operating system environment variable TRANSIMS_CONFIG_FILE. The variable points to a file name that stores the configuration keys. The program reads the configuration keys into memory before it reads the control file keys. If a control key is defined in both files, the value from the control file will override the value in the configuration file.

The path to a configuration file can be set dynamically for a particular application using the SET command within a batch file or at the command prompt. For example:

```
SET  TRANSIMS_CONFIG_FILE=c:\myproject\config.txt
```

Status Codes

TRANSIMS programs return a status code to the operating system based on the results of the application. A return code of zero indicates a successful completion with no warning messages. A return code of 1 indicates that the application was terminated with an error message. A return code of 2 indicates that the program ran to completion, but at least one warning message was generated. The user can detect these return codes within a batch file using the following command:

```
Router.exe  Router.ctl
if %ERRORLEVEL% == 1 exit 1
```

Definition Files

TRANSIMS uses definition files to interpret and define data fields within most input and output files generated by the modeling process. A definition file is automatically created when the file is created. It has the same path and file name as the data file with a ".def" extension added at the end. For example, the program control keys:

```
NEW_LINK_FILE          network\link.txt
NEW_LINK_FORMAT     TAB_DELIMITED
```

create a new link file in the network directory called "link.txt". The format key indicates that the link file will be created in tab delimited format. A definition file called "link.txt.def" will also be created in the network directory. The definition file is a standard text file containing the following information:

```
TRANSIMS50, TAB_DELIMITED, 1
LINK, INTEGER, 1, 10
NAME, STRING, 2, 40
NODE_A, INTEGER, 3, 10
NODE_B, INTEGER, 4, 10
LENGTH, DOUBLE, 5, 8.1, FEET
TYPE, STRING, 10, 12, FACILITY_TYPE
AREA_TYPE, UNSIGNED, 12, 3
LANES_AB, UNSIGNED, 14, 2
SPEED_AB, DOUBLE, 15, 5.1, MPH
FSPD_AB, DOUBLE, 16, 5.1, MPH
CAP_AB, UNSIGNED, 17, 8, VPH
USE, STRING, 22, 128, USE_TYPE
```

The first record in the *.def file specifies the software version that created the file (TRANSIMS 5.0), the data file format (tab delimited), and the number of header records in the data file (1). The header record is followed by one record for each data field. These records include the field name, the data type, the field offset within the data record, the maximum field length and number of decimal places, and, if appropriate, the units or enumeration type of the field. The units field facilitates conversions between English and metric systems. It also automates the process of converting text strings to internal type codes (i.e., enumerations) and back again. Binary files, for example, store the type codes as numbers rather than strings to reduce file size and improve performance.

When an existing file is read by a program, the program looks for the definition file to automatically determine how to read the file and process the data fields. If a definition file is not found, the program will look for a *.FORMAT control key where the user identifies the file format. In many cases, the program can used the file format information to read header records from the data file and construct a definition file. If the file is delimited, the program will read the first 100 records of the file to estimate the data types and field widths. This information is written to a new definition file constructed for the data file. If the estimation process is inaccurate, the user can edit the definition file to correct any inaccuracies.

Binary and fixed column file format definition files cannot be constructed automatically. These file formats do not store field header information in the data file. All information about how to read and interpret the file must be provided in the definition file. The user must manually create a definition file for these file types if they are to be read into a TRANSIMS program. This is also true for delimited files that do not include field names as the first record in the file.

TRANSIMS also supports nested files that include two record types. The first record is the master record that includes a field that identifies the number of nested records that follow. A link delay file is a typical example of a nested data file. The master records define the link, time period, flow and travel time on the link while the nested records define the turning movement links, flows, and travel times.

LINK	DIR	TYPE	START	END	FLOW	TIME	NCONNECT
OUT_LINK		OUT_FLOW		OUT_TIME			
37	0	0	2:00	2:15	2.0	19.4	2
44	1.0	19.4					
41	1.0	19.4					
37	1	0	2:00	2:15	0.5	19.4	0
39	0	0	2:00	2:15	8.0	63.8	3
42	4.0	63.8					
46	11.0	63.8					
43	1.0	63.8					
40	1	0	2:00	2:15	2.0	63.8	1
10	2.0	63.8					
41	0	0	2:00	2:15	1.0	63.8	1
45	1.0	63.8					
41	1	0	2:00	2:15	3.2	63.8	3
37	1.0	63.8					
40	2.0	63.8					
44	1.0	63.8					
42	1	0	2:00	2:15	3.8	63.8	1
41	4.0	63.8					
43	0	0	2:00	2:15	0.8	63.8	0

The definition file for the link delay file shown above looks like this:

```
TRANSIMS50, TAB_DELIMITED, 2, NESTED
LINK, INTEGER, 1, 10
DIR, INTEGER, 2, 1
TYPE, INTEGER, 3, 1
START, TIME, 4, 16, HOUR_CLOCK
END, TIME, 5, 16, HOUR_CLOCK
FLOW, DOUBLE, 6, 8.1, VEHICLES
TIME, TIME, 7, 8.1, SECONDS
NCONNECT, INTEGER, 8, 2, NEST_COUNT
OUT_LINK, INTEGER, 1, 10, NO, NESTED
OUT_FLOW, DOUBLE, 2, 8.1, VEHICLES, NESTED
OUT_TIME, TIME, 3, 8.1, SECONDS, NESTED
```

The first record indicates that the data file has two header records and includes the NESTED key word. The field specifications for the master record are exactly like any other definition file. The nested fields add the NESTED key word after the units field. Note that the record offsets restarts from 1 as well. The field with the NEST_COUNT identifier is used to determine how many nested records follow each master record.