United States
Department
of Agriculture

Forest Service

**Forest
Management
Service Center**

Fort Collins, CO

April, 2005

User's Guide to the Post Processors for the Forest Vegetation Simulator

Michael G. Van Dyck

User's Guide to the Post Processors for the Forest Vegetation Simulator

Introduction

Post processors are stand-alone applications that produce specialized output using, as input, files that have been produced by the Forest Vegetation Simulator (FVS). The required FVS output files vary among the post processors. Many require files that are only produced when specific keywords are included in the FVS simulation file. Others require that keywords be included to write specific information to the main FVS output file. Without these keywords the output files are either not created or do not contain the proper information, and the post processors can not operate correctly.

The *Suppose* graphical user interface program has a facility for launching the post processors as part of a simulation run. When using the *Suppose* program, the post processors will be run without any input from the user. Defaults will be used for most of the program options that are available in the post processors. Default output filenames will also typically be used. When a post processor has completed its calculations, the output is displayed. In most cases, the user may then select from the available options, at which point the post processor will run through its calculations again and display the updated output.

The post processors may also be run outside of the *Suppose* environment. The executable program files for the post processors are typically found in the FVSBIN directory. They can be run by simply double-clicking on the icon for the desired program. They can also be run from a command line by typing the name of the executable program file at a command prompt.

Typically, to run the post processors from a command line, a command prompt window (a DOS window on a PC, or an AIX terminal window in AIX/UNIX) is opened, and the user navigates to the directory in which the required input files reside. The executable program name is then entered. Additional information can be supplied to the post processor through the use of command line arguments. The arguments can include information such as input filename, output filename, and program options. Output from the post processors is written to the same directory in which the required input files were found, unless otherwise specified as a command line argument.

Running a post processor from a command line with the required input file name(s) provided as command line arguments will cause the program to run immediately, processing the input files and displaying the output. If the required file names are not provided, the post processor will open, but it will be up to the user to select the appropriate input file names. If command line arguments for program options are provided, the post processor will use those options unless they are changed in the user interface by the user. Default values will be used for any program options for which a command line argument was not provided, and for which the user did not select a different option in the interface.

Command line arguments for the post processors are separated from the program name and from other command line arguments by a space. Each argument must be immediately preceded by a dash(-). The following is an example of a command line call of the Average Summary Table post processor.

```
sumavg -example.sum -example.avg -W2
```

In this example, the executable program name is `sumavg.exe` (or `sumavg` on an AIX/UNIX system). This is followed by three command line arguments, each of which is immediately preceded by a dash, and is separated from the other arguments by a single space. For now, it is not important what the command line arguments do. That will be explained in the discussion of the individual post processors. This example is provided only to illustrate the correct syntax for calling a post processor using the command line arguments.

Underlined Items

Throughout this document there are items that are double-underlined. Each of those items has a corresponding section that describes the topic in detail. Usually that section is within the same post processor chapter as the underlined item. The chapter devoted to the *Suppose* parameters file is at the end of the document.

Average Summary Table

Variants: All
Keywords required: **EchoSum**
Program filename: SumAvg.exe

The Average Summary Table post processor produces an output file containing a single Summary Statistics table that contains values that have been averaged among all stands in the simulation. The table is formatted the same as the Summary Statistics tables in the main FVS output. Output is created only for cycles that are common to all of the stands in the simulation. Values will be averaged using one of three user-defined weighting methods: weighting equally, by sampling weight (which is usually acreage), or by the number of plots found in the input data for the stand. Additionally, the averages in the "Removals" section of the table can be made to include information only for stands in which a harvest actually occurred (resulting in per-acre averages only for the acres on which a harvest occurred), or they can be made to include information from all stands regardless of whether a harvest occurred (resulting in per-acre averages for the entire area regardless of whether a harvest occurred). This second option may result in values of 0 (zero) being included in the calculation of the average value. Unless otherwise specified as a command line argument, the output filename will have an `.avg` extension.

*The **EchoSum** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

 The Average Summary Table post processor requires an EchoSum file as input. This file is produced by FVS only when the EchoSum keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.sum` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

 The output file will have the same base name as the EchoSum file used as input, but will have an `.avg` extension. For example, if the file `testrun.sum` is used as input, the output file will be named `testrun.avg`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

 There are three methods that may be used to weight the stands when calculating average values. Each is described below. These options may be specified by selecting one of the weighting methods in the Average menu, or by clicking on one of the weight buttons on the toolbar. The output file may also be specified on the command line.

The first weighting method is to give all stands equal weight when calculating the averages. Values are simply summed across all stands in the simulation, and the result divided by the number of stands. This is the default weighting method.

The second weighting method is to weight stands by stand sampling weight when calculating the averages. Values are multiplied by the sampling weight, and the results are then summed. The total is divided by the total sampling weight for all stands combined.

Sampling weight usually represents stand acreage. It is specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file.

The third weighting method is to weight stands by number of plots in the stand inventory. What FVS considers plots may be referred to as inventory points or subplots, depending on the terminology used in the inventory procedures and the unit that was specified as a stand for the purposes of FVS. Values are multiplied by the number of plots in the stand, and the results are then summed. The total is divided by the total number of plots in all stands combined. The number of plots may specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file. If this information is not provided, FVS counts the number of unique plot identification codes found in the tree data file for the stand.

 The user may specify whether the averages that are calculated for harvest values are to exclude stands where no harvest occurred (and therefore have a 0 for the removal values). The per-acre average harvest values that are calculated will then represent only the area of the stands in which a harvest occurred that cycle. This might be desirable if the required information is the average amount of material removed per acre in a particular harvest operation. By default the zero values are not excluded, so the per-acre average values for the harvests represent the entire area of all stands, regardless of whether a harvest occurred that cycle. The option to exclude zero values in the harvest averages may be specified by selecting the Exclude Zeros option in the Average menu, or by clicking on the no-zero button on the toolbar.

Output

In order to produce meaningful output, the Average Summary Table post processor requires a valid EchoSum file as input. This file is produced by FVS only when the EchoSum keyword is included in the simulation. If the EchoSum keyword is not included in the simulation the EchoSum file will not be produced and the post processor will report an error.

The output table for the Average Summary Table Post Processor is formatted the same as the Summary Statistics tables in the main FVS output. A sample output table is shown below, followed by a description of the types of values it contains.

```
FVS SIMULATION: testrun
SIMULATION DONE: 01-01-2003 12:34:56

                                        AVERAGE* SUMMARY STATISTICS BY COMMON CYCLE
-------------------------------------------------------------------------------------------------------------------------------------
              START OF SIMULATION PERIOD                    REMOVALS**            AFTER TREATMENT    GROWTH THIS PERIOD
         ------------------------------------------ ---------------------- ---------------------  ------------------   MAI
         NO OF              DOM      TOTAL MERCH MERCH NO OF TOTAL MERCH MERCH              DOM  RES  PERIOD ACCRE MORT   MERCH
YEAR AGE TREES  BA  SDI CCF HT  QMD  CU FT CU FT BD FT TREES CU FT CU FT BD FT  BA  SDI CCF HT   QMD  YEARS   PER  YEAR   CU FT
---- --- ----- --- ---- --- --- ---- ----- ----- ----- ----- ----- ----- ----- --- ---- --- --- ----  ------ ---- -----   -----
2000  98   518  81  185  74  54  8.6  1828  1446  6037     0     0     0     0  81  185  74  54  8.6      10   29     2     9.0
2010 108   513  94  208  87  55  9.2  2095  1648  7132   685   903   565  1266  56   86  47  55 16.0      10   17     1     9.6
2020 118   853  66  169  58  56  8.1  1638  1401  7073     0     0     0     0  66  169  58  56  8.1      10   21     1     9.7
2030 128   842  77  191  68  57  8.6  1836  1539  7913     0     0     0     0  77  191  68  57  8.6       0    0     0     9.8

*STANDS WERE WEIGHTED BY SAMPLING WEIGHT (USUALLY ACRES.)
**REMOVAL AVERAGES ARE BASED ONLY ON VALUES FROM STANDS IN WHICH A HARVEST OCCURRED THAT CYCLE.

THE FOLLOWING STANDS WERE SUMMARIZED:
  Stand201                  Stand202
```

DESCRIPTION OF THE OUTPUT

The columns of values represent the same attribute as the corresponding columns in the Summary Statistics table in the main FVS output file. All values are per acre for the entire area of the stands (including any nonstockable area), just as the values in the Summary Statistics table in the main FVS output file are for the entire stand area. The amount of nonstockable area in a stand is specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file.

A footnote is included in the output to indicate the method used to weight the values from the individual stands when calculating the average values. In the example, the stands were weighted by sampling weight. Sampling weight is also specified in the Design

keyword record. The method used to weight the values is a program option specified either in the interface to the Average Summary Table post processor, or on the command line used to launch the post processor.

A second footnote is included in the output to indicate whether the values reported as "removals" include information from all stands regardless of whether a harvest occurred, or only from stands in which a harvest actually occurred that cycle. The per-acre average value, therefore, respectively represents either the entire area of the stands, or the area of stands in which a harvest actually occurred that cycle. The choice of how to deal with zero values in harvests is another program option.

The column headings that are used in the output are described below.

START OF SIMULATION

Year	Year in which the FVS cycle began. Average values are reported only for years that are common to all stands in the simulation.
Age	Average age of the stand
No Of Trees	Average number of live trees per acre
BA	Average basal area (ft2/acre) for all live trees
SDI	Average stand density index calculated for all live trees
CCF	Average crown competition factor for all live trees
Dom Ht	Average height (feet) of the 40 largest diameter live trees in each stand
QMD	Average quadratic mean diameter at breast height (inches) for all live trees
Total Cu Ft	Average total cubic foot volume (western U.S.) or average cubic foot volume of the pulpwood and sawlog portions combined (eastern U.S.)
Merch Cu Ft	Average merchantable cubic foot volume (western U.S.) or average sawlog cubic foot volume (eastern U.S.)
Merch Bd Ft	Average merchantable board foot volume (western U.S.) or average sawlog board foot volume (eastern U.S.)

REMOVALS (see the discussion above regarding calculation of harvest average values)

No Of Trees	Average number of trees per acre removed in harvests
Total Cu Ft	Average removed total cubic foot volume (western U.S.) or average removed cubic foot volume of the pulpwood and sawlog portions combined (eastern U.S.)
Merch Cu Ft	Average removed merchantable cubic foot volume (western U.S.) or average removed sawlog cubic foot volume (eastern U.S.)
Merch Bd Ft	Average removed merchantable board foot volume (western U.S.) or average removed sawlog board foot volume (eastern U.S.)

AFTER TREATMENT

BA	Average basal area (ft2/acre) for residual live trees
SDI	Average stand density index for all residual live trees
CCF	Average crown competition factor for all residual live trees
Dom Ht	Average height (feet) of the 40 largest diameter residual trees in each stand
Res QMD	Average residual quadratic mean diameter at breast height (inches) for all residual live trees

GROWTH THIS PERIOD

Period Years	Length of the cycle (years)
Accre Per Year	Average accretion (ft3/acre/year) for the trees that survived to the end of the cycle
Mort Per Year	Average mortality (ft3/acre/year) for the cycle
MAI Merch Cu Ft	Average mean annual increment (ft3/acre/year)

Command Line Arguments

The Average Summary Table post processor may be run by typing `sumavg` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If

an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*EchoSumFilename*	The name of the EchoSum file that is to be used as input. This file usually has a `.sum` extension. The actual filename is used in place of the designation *EchoSumFilename* shown here. This must be the first command line argument.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the EchoSum file with an `.avg` extension.

-W#	Weighting method, where the # is a number representing the weighting method. The default method is equal weight. All of the possible weighting designations are shown below.
	W1 = equal weight (this is the default)
	W2 = weight by stand weight (which is usually acres)
	W3 = weight by number of points

-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.
-Y	Yes, ignore zero values in averages for "Removals" section of output. The default is not to ignore zero values.

If the -W#, -X, or -Y command line arguments are to be used, the EchoSum filename argument and output filename argument must also be used. In other words, the -W#, -X, and -Y arguments may never appear as the first or second command line argument.

Command Line Example:

```
sumavg -example.sum -example.avg -W2
```

This will run the Average Summary Table post processor using the file `example.sum` as input, using the file `example.avg` for the output, averaging values by stand weight, and not ignoring zero values in harvest averages.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Average Summary Table post processor is shown below.

```
//start ppif.avgsum
name:{Average Summary Table}
command{dos}:{
!fvsbin!\\sumavg.exe -!run!.sum  -!run!.avg}
...
//end ppif.avgsum
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\sumavg.exe` will become `C:\Fvsbin\sumavg.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.sum`, which represents the name of the

EchoSum file used as input for the post processor. This is the default name that FVS will give the EchoSum file. The second command line argument would be `-testrun.avg`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.avg` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to average values by stand weight and ignore zero values in calculating harvest averages, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\sumavg.exe -!run!.sum  -!run!.avg -w2 -y
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Calibration Summary Statistics

Variants: All
Keywords required: **CalbStat**
Program filename: calibsp.exe

The Calibration Summary Statistics post processor produces an output file containing a table containing information on the calibration statistics for all species in all stands for which calibration factors have been calculated by FVS. A second table contains calibration factors, by species, that represent average values calculated from all stands in the simulation. Calibration will be either for the large tree diameter growth model (LD or LG TREE DG), or the small tree height growth model (SH or SM TREE HTG). Unless otherwise specified as a command line argument, the output filename will have a `.csf` extension.

*The **CalbStat** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Calibration Summary Statistics post processor requires a CalbStat file as input. This file is produced by FVS only when the CalbStat keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.chp` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

The output file will have the same base name as the CalbStat file used as input, but will have a `.csf` extension. For example, if the file `testrun.chp` is used as input, the output file will be named `testrun.csf`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

Output

In order to produce meaningful output, the Calibration Summary Statistics post processor requires a valid CalbStat file as input. This file is produced by FVS only when the CalbStat keyword is included in the simulation. If the CalbStat keyword is not included in the simulation the CalbStat file will be empty and the post processor will report an error.

A sample output table is shown below, followed by a description of the types of values it contains.

```
CALIBRATION STATISTICS
GENERATED BY RUNSTREAM : C:\fvsdata\testrun
DATE: 01-01-2003     TIME: 12:34:56   VARIANT: SF 6.31
```

```
*Model type LD is for large tree diameter growth.
 Model type SH is for small tree height growth.
```

STAND ID	MGMT ID	MODEL TYPE*	SPECIES NUMBER	SPECIES CODE	NUMBER OF TREE RECORDS	FVS SCALE FACTOR	RATIO STD. ERROR	BAYES WEIGHT	READCORx KEYWORD MULTIPLIER
Stand217	NONE	LD	20	AS	12	0.897	2.761	0.729	0.861
Stand217	NONE	SH	20	AS	7	1.003	2.661	0.489	1.003
Stand220	NONE	LD	20	AS	6	1.103	2.761	0.589	1.181
Stand220	NONE	LD	18	ES	5	1.383	0.828	0.900	1.434
Stand220	NONE	SH	18	ES	6	1.283	0.728	0.800	1.283

LARGE TREE DIAMETER GROWTH CALIBRATION SUMMARY

SPECIES	MODEL TYPE*	SCALE FACTOR SUMMARY N	MIN	MEAN	MAX	STD DEV.	TOTAL TREE RECORDS	MEAN READCORD MULTIPLIER
18 ES	LD	1	1.383	1.383	1.383	1.382	5	1.434
20 AS	LD	2	0.897	1.000	1.103	0.146	18	0.968

SMALL TREE HEIGHT GROWTH CALIBRATION SUMMARY

SPECIES	MODEL TYPE*	SCALE FACTOR SUMMARY N	MIN	MEAN	MAX	STD DEV.	TOTAL TREE RECORDS	MEAN READCORR MULTIPLIER
18 ES	SH	1	1.283	1.283	1.283	1.282	6	1.283
20 AS	SH	1	1.003	1.003	1.003	1.002	7	1.003

THE FOLLOWING STANDS WERE SUMMARIZED

Stand217 Stand220

DESCRIPTION OF THE OUTPUT

Stand Id	Stand identification code
Mgmt Id	4-character management code from the MgmtId keyword record
Model Type	Code indicating growth model affected LD indicates large-tree diameter growth model SH indicates small-tree height growth model
Species Number	Variant-specific species sequence number
Species Code	2-character species code
Number Of Tree Records	Number of input tree records that included growth information used in calculating the scale factors for that species and growth model
FVS Scale Factor	Initial scale factor used in the FVS simulation for that species and growth model
Ratio Std. Error	Ratio of the standard deviation of the residuals from the input data to the standard error of the data used to develop the FVS variant
Bayes Weight	Weight included in a complex empirical Bayes estimation process
READCORx Keyword Multiplier	Multiplier that would be used in a ReadCorD or ReadCorR keyword record to make the initial scale factor equal 1.0 in that stand for that species and growth model
N	Number of stands that contributed scale factors for that species and growth model
Min	Minimum initial scale factor encountered for that species and growth model
Mean	Maximum initial scale factor encountered for that species and growth model
Max	Mean initial scale factor for that species and growth model
Std Dev.	Standard deviation for initial scale factors for that species and growth model
Total Tree Records	Total number of tree records of that species and growth model that included growth information used in calculating the scale factors
Mean READCORD Multiplier	Mean multiplier (over all tree records of that species and growth model) that should be used in a ReadCorD keyword record to scale the diameter growth of the large trees

Mean READCORR Multiplier	Mean multiplier (over all tree records of that species and growth model) that should be used in a ReadCorR keyword record to scale the height growth of the small trees

Command Line Arguments

The Calibration Summary Statistics post processor may be run by typing `calib` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*CalbStatFilename*	The name of the CalbStat file that is to be used as input. This file usually has a `.chp` extension. The actual filename is used in place of the designation *CalbStatFilename* shown here. This must be the first command line argument.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the CalbStat file with a `.csf` extension.
-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -X command line argument is to be used, the CalbStat filename argument and output filename argument must also be used. In other words, the -X argument may never appear as the first or second command line argument.

Command Line Example:

```
calibsp -example.chp -example.csf
```

This will run the Calibration Summary Statistics post processor using the file `example.chp` as input, and using the file `example.csf` for the output.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Calibration Summary Statistics post processor is shown below.

```
//start ppif.calibrat
name:{Calibration Summary Statistcs}
command{dos}:{
!fvsbin!\\calib.exe  -!run!.chp}
...
//end ppif.calibrat
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\calib.exe` will become `C:\Fvsbin\calib.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.chp`, which represents the name of the CalbStat file used as input for the post processor. This is the default name that FVS will give the CalbStat file. It is highly recommended that the `!run!` designation in the filename not be changed.

If other command line arguments are desired, they should be added after `-!run!.chp` but before the }. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to write the output to a file with a `.xyz` extension, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\calib.exe -!run!.chp  -!run!.xyz
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Computed Variable Tables

Variants: All
Keywords required: **Compute**
Program filename: compspp.exe

The Computed Variable Tables post processor is a single program that has replaced three former post processors called Compute, Compute2, and Average Compute. The list of post processors in the *Suppose* program still shows Compute1, Compute2, and Compute3, corresponding to the three post processors just mentioned, but the same program is now opened for each of them. The options that produce the same type of output as was previously produced with each of these options is provided by *Suppose*, so the transition to the new system is fairly transparent to the user.

When called as the Compute 1 post processor in *Suppose*, the output file contains a table of the variable names and values for each variable, by cycle, for each stand in the simulation. Unless otherwise specified as a command line argument, the output filename will have a `.cmp` extension.

When called as the Compute 2 post processor in *Suppose*, the output file contains a single, comma delimited table of values for the Compute variables for all stands in the simulation. Variables in the table are identified by stand id and cycle number. Variables whose value was not computed for a particular stand or cycle will show a blank for that entry. Unless otherwise specified as a command line argument, the output filename will have a `.cp2` extension.

When called as the Compute 3 post processor in *Suppose*, the output file contains a single table of average values for the Compute variables for all stands in the simulation. Values are written only for cycles that are common to all of the stands in the simulation. Values will be averaged using one of three user-defined weighting methods: weighting equally, by sampling weight (which is usually acreage), or by the number of plots found in the input data for the stand. Unless otherwise specified as a command line argument, the output filename will have a `.avc` extension.

*The **Compute** keyword with valid variable definitions must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Computed Variable Tables post processor requires a main FVS output file as input. Unless Summary Statistics are the only desired output, this file must contain information that was calculated using the Compute keyword in the simulation. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

The output file will have the same base name as the main FVS output file used as input, but will have a `.cmp` extension. For example, if the file `testrun.out` is used as input, the output file will be named `testrun.cmp`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

 There are two main types of variables that can be displayed in the output tables. The first is the set of variables that are defined by the user with the Compute keyword record included in the simulation file. These variables are named by the user and can contain anything that FVS can compute. The second is the set of Summary Statistics variables. These are the values that are displayed in the Summary Statistics table in the main FVS output file. Each of these variables is pre-defined to contain a specific type of information, such as trees per acre.

There are several ways to specify which variables are to be displayed in the output tables. The first is to specify that all computed variables are to be included. With this option, any variable that is successfully computed and displayed in the Activity Summary of the main FVS output file is included. The second option is to list the individual variable names that are to be included. The list is not case sensitive, so VAR1 is the same as var1 and Var1. If the variable name is found in the Activity Summary with a successfully computed value it will be included in the output table. The third option is to include the variable names in a text file and provide the post processor with the name of that file. Each line in the file contains a single variable name. Again, the names are not case sensitive. If the variable name is found in the Activity Summary with a successfully computed value it will be included in the output table.

In addition to the computed variables, any or all of the Summary Statistics variables may be included in the output tables. These are selected individually in the variable selection window. The variable names are pre-defined, and will be displayed with an asterisk (*) as the first character of the name. This is to distinguish them from computed variables.

The variables that are to be included in the output tables are specified in the variable selection window. This window is accessed by using the Set... option in the Variables menu, or by clicking the X= button on the toolbar. Individual variable names or the name of a variable name file can also be specified on the command line. There is also a command line option to include all of the Summary Statistics variables.

 There are three main options for the output table type, which correspond to the old Compute1, Compute2, and Compute3 post processors. Each of these is described below. The table type may be specified by selecting one of the types in the Table menu, or by clicking on one of the table-type buttons on the toolbar. The table type may also be specified on the command line. in conjunction with the table format.

The first table type produces a table for each stand in the simulation, like the old Compute1 post processor did. Only the variables that were present in the section of the main FVS output file for any particular stand are shown in the table for that stand.

The second table type produces a single, concatenated table with the variable names and values from all of the stands in the simulation, like the old Compute2 post processor did. A single header record is produced, which contains the names of all of the variables found in the entire main FVS output file for all stands. The values are listed by stand and year only for those variables that were successfully computed for the particular stand and cycle. The stand identification code is the first item of every data record.

The third table type produces a single table with the average values for all of the variables from all of the stands in the simulation, like the old Compute3 post processor did. The weighting used in calculating the averages is specified using the weighting options described below.

 There are three methods, each of which is described below, that may be used to weight the stands when calculating average values. These options may be specified by selecting one of the weighting methods in the Average menu, or by clicking on one of the weight buttons on the toolbar. The output file may also be specified on the command line. If the output table type (described above) is anything other than average values this option is not available.

The first weighting method is to give all stands equal weight when calculating the averages. Values are simply summed across all stands in the simulation, and the result divided by the number of stands. This is the default weighting method.

The second weighting method is to weight stands by stand sampling weight when calculating the averages. Values are multiplied by the sampling weight, and the results are then summed. The total is divided by the total sampling weight for all stands combined. Sampling weight usually represents stand acreage. It is specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file.

The third weighting method is to weight stands by number of plots in the stand inventory. What FVS considers plots may be referred to as inventory points or subplots, depending on the terminology used in the inventory procedures and the unit that was specified as a stand for the purposes of FVS. Values are multiplied by the number of plots in the stand, and the results are then summed. The total is divided by the total number of plots in all stands combined. The number of plots may specified in the Design keyword record,

which the *Suppose* program builds from information in the stand list file. If this information is not provided, FVS counts the number of unique plot identification codes found in the tree data file for the stand.

 There are two options for the format of the tables, each of which is described below. The table format may be specified by selecting one of the formats in the Table menu, or by clicking on one of the format buttons on the toolbar. The format may also be specified on the command line in conjunction with the table type.

The first option is a standard tabular format where variable values are in columns below the variable name. The items are simply spaced to line up correctly. Additional descriptive information is included before and after the tables.

The second option is a comma delimited format where items are separated by commas. Generally, any information that is included is part of the table itself. This format is typically used to when the values are to be imported into another program, like a spreadsheet or database.

Output

In order to produce meaningful output, the Computed Variable Tables post processor requires a valid main FVS output file as input. If computed variables are desired they must be defined using the Compute keyword. If computed variables are requested and no computed variables appear in the Activity Summary the post processor will report an error. If the only variables requested are Summary Statistics variables the Compute keyword record is not necessary.

Several sample output tables are shown below, followed by a description of the types of information each contains. There are program options that determine the type of table, as well as the format. These program options are specified either in the interface to the Compute Tables post processor, or on the command line used to launch the post processor.

The first sample table was produced using the table type for creating tables for each stand. All computed variables were requested, and the standard format was used. No Summary Statistics variables were selected. This is the type of output generated using the default Compute1 option in *Suppose*, with the exception that the option was selected to include several of the Summary Statistics variables. After the main header there is a table produced for each of the stands in the simulation.

```
                  VALUES COMPUTED IN THE FOREST VEGETATION SIMULATOR
                  CENTRAL ROCKIES SPRUCE-FIR GENGYM  Rev: 01.01.2003

Source: C:\fvsdata\testrun.out  01-01-2003  12:34:56


Stand ID: Stand405
Mgmt ID:  NONE

YEAR  FIRSTVAR  THRDVAR   SECNDVAR  BSDI*     BTCUFT*
----  --------  --------  --------  --------  --------
2000     30.01   2421.90                45       652
2010     33.02   2941.75     20.04      48       748
2020     35.95   3366.67                51       845
2030                                    54       930


Stand ID: Stand407
Mgmt ID:  NONE

YEAR  FIRSTVAR  SECNDVAR  BSDI*     BTCUFT*
----  --------  --------  --------  --------
2000    103.61                248      2363
2010    121.38    736.55      281      2707
2020     80.24                223      1999
2030                          253      2248
```

The second output table, shown below, was produced using the table type for creating a single concatenated table. All computed variables were requested, and the comma delimited format was used. No Summary Statistics variables were selected. This is the type of output generated using the default Compute2 option in *Suppose*, with the exception that the option was selected to include several of the Summary Statistics variables.

```
STAND ID                 ,MGMT,YEAR,FIRSTVAR  ,THRDVAR   ,SECNDVAR  ,BSDI*     ,BTCUFT*   ,
Stand405                 ,NONE,2000,     30.01,   2421.90,          ,       45,       652,
Stand405                 ,NONE,2010,     33.02,   2941.75,     20.04,       48,       748,
Stand405                 ,NONE,2020,     35.95,   3366.67,          ,       51,       845,
Stand405                 ,NONE,2030,          ,          ,          ,       54,       930,
Stand407                 ,NONE,2000,    103.61,          ,          ,      248,      2363,
Stand407                 ,NONE,2010,    121.38,          ,    736.55,      281,      2707,
Stand407                 ,NONE,2020,     80.24,          ,          ,      223,      1999,
Stand407                 ,NONE,2030,          ,          ,          ,      253,      2248,
```

The third output table, shown below, was produced using the table type for creating a single table of average values. All computed variables were requested, and the standard format was used. No Summary Statistics variables were selected. This is the type of output generated using the default Compute1 option in *Suppose*, with the exception that the option was selected to weight the averages by stand sampling weight.

```
                    AVERAGE** VALUES COMPUTED IN THE FOREST VEGETATION SIMULATOR
                    CENTRAL ROCKIES SPRUCE-FIR GENGYM  Rev: 01.01.2003

Source: C:\fvsdata\testrun.out  01-01-2003  12:34:56
** Stands were weighted by sampling weight (usually acres) in calculating the average values.


YEAR  FIRSTVAR  THRDVAR   SECNDVAR  BSDI*     BTCUFT*
----  --------  --------  --------  --------  --------
2000     71.50   2421.90                185      1828
2010     93.77   2941.75    506.56      208      2095
2020     66.40   3366.67                169      1638
2030                                    191      1836

The following stands were summarized:

STAND                       MGMT ID
--------------------------  -------
Stand405                    NONE
Stand407                    NONE
```

DESCRIPTION OF THE OUTPUT

Source	FVS output file from which the values were read, and the date and time that file was produced
Stand ID	26-character stand identification code
Mgmt ID	Management code assigned using the MgmtId keyword
Year	Year in which the FVS cycle began
FIRSTVAR, etc	Variable names specified through use of the Compute keyword. The values are read from the Activity Summary section of the main FVS output file. The values in that section are shown to two decimal places, so the values displayed for those variables are also shown to two decimal places. This is true even if the value being calculated is a whole number (no decimal part). Note that values are never computed for the final year of a simulation in FVS. There will, therefore, never be values displayed for the final year.
BSDI*, etc	Variable names that have been assigned to the Summary Statistics variables. All of these variable names are the same as the corresponding event monitor variable except that they end with an asterisk (*). The values are read from the Summary Statistics section of the main FVS output file. Since the values in that section are rounded to whole numbers, the values displayed for those variables are also shown as whole numbers (no decimal part). Summary Statistics are displayed for the last year of a simulation, so there may be values displayed for that year for Summary Statistics variables when there are no values shown for computed variables.

In the examples shown above, the variable FIRSTVAR was computed for all cycles for both stands, the variable THRDVAR was computed only for Stand405, and the variable SECNDVAR was computed only for the cycle beginning in 2010. The variables BSDI* and BTCUFT* are Summary Statistics variables. The variables that are to be displayed, including the Summary Statistics variables, may be specified as program options.

The first example shown above displays a separate table for each stand in the simulation. The information in the second example is exactly the same as was shown in the first example, with the exception that the header is missing. The only real differences are that the information is now displayed in a single, concatenated table, and the format is now comma delimited. In the third example the values displayed for the variables are averages over all Stands and Management IDs. The averages were weighted by stand sampling weight, which in this case was stand acreage. This is indicated by a note in the header. Individual values are multiplied by the stand acreage, and the results summed together over all stands. The overall result is divided by the total acreage of all stands that contributed to the result. Since the variable THRDVAR was only computed in one of the stands, that is the only stand that contributes to the average in this case. The average values calculated for the Summary Statistics variables are based on individual values that were rounded to whole numbers. This may introduce error into the average values that are calculated.

Command Line Arguments

The Compute Variable Tables post processor may be run by typing `compute` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*FvsOutputFilename*	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. This must be the first command line argument.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the main FVS output file with a `.cmp` extension.
-F	A file contains the names of the variables to be processed. The next command line argument will be read as the filename. The file must be a text file containing a list of variable names, one on each line in the file.

-L#	Layout for the output tables, where the # is a number representing the table layout. All of the possible layout designations are shown below.
	L1 = separate tables for each stand, standard format (this is the default)
	L2 = separate tables for each stand, comma delimited format
	L3 = a single, concatenated table for all stands, standard format
	L4 = a single, concatenated table for all stands, comma delimited format
	L5 = a single table of average values, standard format
	L5 = a single table of average values, comma delimited format

-S	Include all of the Summary Statistics variables
-T	Title. The next command line argument will be read as the title. The default is to have no title.

-W#	Weighting method, where the # is a number representing the weighting method. All of the possible weighting designations are shown below. The default method is equal weight.
	W1 = equal weight
	W2 = weight by stand weight
	W3 = weight by number of points

-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -F, -L#, -S, -T, -W#, or -X command line arguments are to be used, the FVS output filename argument and output filename argument must also be used. In other words, the -F, -L#, -S,- T, -W#, or -X argument may never appear as the first or second command line argument.

Care must be taken to include the required additional command line argument when using the -F and -T arguments.

Command Line Example:

```
compute -example.out -example.avc -T -WeightByAcres -L5 -W2
```

This will run the Computed Variable Tables post processor using the file `example.out` as input, using the file `example.avc` for the output, using "WeightByAcres" as the title, producing a standard-format table of average values, and weighting averages by stand weight. Since the -F argument was not used, the default behavior of processing all variables will be used.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with what is referred to in *Suppose* as the Compute1 post processor is shown below.

```
//start ppif.compute1
name:{Compute1 - Table of Compute Variables (with headers)}
command{dos}:{
!fvsbin!\\compute.exe -!run!.out -!run!.cmp -t -SIMULATION:!run!}
...
//end ppif.compute1
```

A portion of the section of the suppose.prm file that deals with what is referred to in *Suppose* as the Compute2 post processor is shown below.

```
//start ppif.compute2
name:{Compute2 - Table of Concatenated Compute Variables (comma delimited)}
command{dos}:{
!fvsbin!\\compute.exe -!run!.out -!run!.cp2 -l4}
...
//end ppif.compute2
```

A portion of the section of the suppose.prm file that deals with what is referred to in *Suppose* as the Compute2 post processor is shown below.

```
//start ppif.compute3
name:{Compute3 - Table of Compute Variable Averages}
command{dos}:{
!fvsbin!\\compute.exe -!run!.out -!run!.avc -t -SIMULATION:!run! -l5}
...
//end ppif.compute3
```

In each case, the command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\compute.exe` will become `C:\Fvsbin\compute.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them. It is worth noting that all three of the compute post processors listed in Suppose call the same program. They just use different table layouts.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.out`, which represents the name of the main FVS output file used as input for the post processor. This is the default name that FVS will give the main FVS output file. The second command line argument would be `-testrun.cmp`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.cmp` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to average values by stand weight and ignore zero values in calculating harvest averages, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\compute.exe -!run!.out  -!run!.cmp -w2 -y
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Fuels and Potential Fire Table

Variants: All
Keywords required: **FuelOut, PotFire**
Program filename: firetbl.exe

The Fuels and Potential Fire Table post processor produces an output file containing fuels and potential fire information for all stands in the simulation file. Unless otherwise specified as a command line argument, the output filename will have a `.ftb` extension.

*A **FuelOut** and/or **PotFire** keyword must be included in the FVS simulation file.* These keywords are part of the Fire and Fuels Extension (FFE). They are only available when an FVS variant that includes the FFE exists in the directory specified for FVS software (typically C:\Fvsbin on a PC). The FuelOut keyword will provide information about fuels in the stand. The PotFire keyword will provide information about potential fires.

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Fuels and Potential Fire Table post processor.

Program Options

 The Fuels and Potential Fire Table post processor requires a main FVS output file as input. This file must contain information that was produced by including a FuelOut and/or PotFire keyword in the simulation. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

 The output file will have the same base name as the main FVS output file used as input, but will have a `.ftb` extension. For example, if the file `testrun.out` is used as input, the output file will be named `testrun.ftb`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

 There are two main types of variables that can be displayed in the output tables. Each is described below. The types of variables that are to be included in the output table are specified by using the options in the Variables menu, or by clicking the fire or fuels button on the toolbar. Both the menu items and the toolbar buttons act as toggle switches to include or not include the variables of that type. This information can also be specified on the command line.

The first type of variables are those defined when the PotFire keyword record is included in the simulation file. The information used to populate the variables is read from the Potential Fuels Report in the main FVS output file. Each of these variables is pre-defined to contain a specific type of information, such as total flame length in a severe fire.

The second type of variables are those defined when the FuelOut keyword record is included in the simulations file. The information used to populate the variables is read from the All Fuels Report in the main FVS output file. Each of these variables is pre-defined to contain a specific type of information, such as tons per acre of litter.

Output

In order to produce meaningful output, the Fuels and Potential Fire Table post processor requires a valid main FVS output file as input. This file must contain the table created by the FuelOut keyword or the PotFire keyword. If fuels variables are requested and the All Fuels Report does not appear in the main FVS output file the post processor will report an error. Similarly, if the potential fire variables are requested and the Potential Fuels Report does not appear in the main FVS output file the post processor will report an error. The types of variables that are to be displayed are specified as program options, either in the interface to the Fuels and Potential Fire Table post processor, or on the command line used to launch the post processor.

Part of a sample output table is shown below. A description of all the types of information the output files may contain follows the table.

```
Stand Id                ,Year ,Litter   ,Duff     ,SrfDead1 ,SrfDead2 ,SrfDead3 ,SrfDead4 ,SrfDead5 ,
Stand507                , 2002,     0.94,      9.0,      2.3,      5.9,      2.9,      2.9,      0.0,
Stand507                , 2003,     1.01,      9.0,      2.2,      5.8,      2.9,      2.9,      0.0,
Stand507                , 2004,     1.02,      9.0,      2.0,      5.7,      2.8,      2.8,      0.0,
Stand507                , 2005,     1.02,      9.0,      1.9,      5.6,      2.8,      2.8,      0.0,
Stand507                , 2006,     1.03,      9.0,      1.7,      5.5,      2.8,      2.8,      0.0,
Stand507                , 2007,     1.03,      9.0,      1.6,      5.4,      2.7,      2.7,      0.0,
Stand507                , 2008,     1.03,      9.0,      1.5,      5.4,      2.7,      2.7,      0.0,
Stand507                , 2009,     1.03,      9.0,      1.4,      5.3,      2.6,      2.6,      0.0,
Stand507                , 2010,     1.03,      9.0,      1.3,      5.2,      2.6,      2.6,      0.0,
Stand507                , 2011,     1.03,      9.0,      1.2,      5.1,      2.6,      2.6,      0.0,
Stand507                , 2012,     0.00,      2.0,      0.2,      1.1,      0.1,      0.9,      0.0,
```

DESCRIPTION OF THE OUTPUT

Stand Id	26-character stand identification code
Year	Year for which the FVS calculations were done

POTENTIAL FIRE VARIABLES

In all variants:

S_Wind	Windspeed (miles per hour) used in the Severe Fire calculations. This value may be specified on the PotFWind keyword record, otherwise a default value is used.
S_Temp	Ambient temperature (ºF) used in the Severe Fire calculations. This value may be specified on the PotFTemp keyword record, otherwise a default value is used.
S_Mois1	Percent moisture for 1-hour fuels (0.0 - 0.25 inches in diameter) used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_Mois2	Percent moisture for 10-hour fuels (0.25 - 1.0 inches in diameter) used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_Mois3	Percent moisture for 100-hour fuels (1.0 - 3.0 inches in diameter) used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_Mois4	Percent moisture for 1-hour fuels (3.0 inches and greater in diameter) used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_MoisDf	Percent moisture for duff used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_MoisLv	Percent moisture for live fuels used in the Severe Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
S_Mort%B	Potential mortality as a percentage of basal area that would be expected under the conditions specified above for a Severe Fire
S_MortCf	Potential mortality in cubic foot volume that would be expected under the conditions specified above for a Severe Fire
S_Smoke	Potential amount of smoke production (tons/acre) that would be expected under the conditions specified above for a Severe Fire

M_Wind	Windspeed (miles per hour) used in the Moderate Fire calculations. This value may be specified on the PotFWind keyword record, otherwise a default value is used.
M_Temp	Ambient temperature (ºF) used in the Moderate Fire calculations. This value may be specified on the PotFTemp keyword record, otherwise a default value is used.
M_Mois1	Percent moisture for 1-hour fuels (0.0 - 0.25 inches in diameter) used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_Mois2	Percent moisture for 10-hour fuels (0.25 - 1.0 inches in diameter) used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_Mois3	Percent moisture for 100-hour fuels (1.0 - 3.0 inches in diameter) used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_Mois4	Percent moisture for 1-hour fuels (3.0 inches and greater in diameter) used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_MoisDf	Percent moisture for duff used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_MoisLv	Percent moisture for live fuels used in the Moderate Fire calculations. This value may be specified on the PotFMois keyword record, otherwise a default value is used.
M_Mort%B	Potential mortality as a percentage of basal area that would be expected under the conditions specified above for a Moderate Fire
M_MortCf	Potential mortality in cubic foot volume that would be expected under the conditions specified above for a Moderate Fire
M_Smoke	Potential amount of smoke production (tons/acre) that would be expected under the conditions specified above for a Moderate Fire
CnpyBase	Height (feet) to the base of the live crown
BulkDens	Crown bulk density (kg/m3)

In only the western variants:

S_SurFL	Surface fire flame length (feet) calculated for a fire occurring under the conditions specified above for a Severe Fire.
S_TotFL	Total flame length (feet) calculated for a fire occurring under the conditions specified above for a Severe Fire.
S_Type	Type of fire expected under the conditions specified above for a Severe Fire.
S_PTorch	The proportion of small places where torching is possible for a Severe Fire.
S_Torch	20-foot windspeed (miles per hour) required for torching of some of the trees
S_Crown	20-foot windspeed (miles per hour) required for crowning of the entire stand
M_SurFL	Surface fire flame length (feet) calculated for a fire occurring under the conditions specified above for a Moderate Fire.
M_TotFL	Total flame length (feet) calculated for a fire occurring under the conditions specified above for a Moderate Fire.
M_Type	Type of fire expected under the conditions specified above for a Moderate Fire.
M_PTorch	The proportion of small places where torching is possible for a Moderate Fire.
FuelMod1	Fire behavior fuel model given highest weight in the calculations. Refer to the FFE documentation for a description.
FMod1Wt	Weight (%) given to FuelMod1 in the calculations
FuelMod2	Fire behavior fuel model given second highest weight in the calculations.
FMod2Wt	Weight (%) given to FuelMod2 in the calculations
FuelMod3	Fire behavior fuel model given third highest weight in the calculations.
FMod3Wt	Weight (%) given to FuelMod3 in the calculations
FuelMod4	Fire behavior fuel model given fourth highest weight in the calculations.
FMod4Wt	Weight (%) given to FuelMod4 in the calculations

In only the southern variant:

S_Flame	Flame length (feet) calculated for a fire occurring under the conditions specified above for a Severe Fire.
M_Flame	Flame length (feet) calculated for a fire occurring under the conditions specified above for a Moderate Fire.
SFuelMod1	Fire behavior fuel model given highest weight in the calculations for the severe case. Refer to the FFE documentation for a description.
SFMod1Wt	Weight (%) given to FuelMod1 in the calculations for the severe case.
SFuelMod2	Fire behavior fuel model given second highest weight in the calculations for the severe case.
SFMod2Wt	Weight (%) given to FuelMod2 in the calculations for the severe case.
SFuelMod3	Fire behavior fuel model given third highest weight in the calculations for the severe case.
SFMod3Wt	Weight (%) given to FuelMod3 in the calculations for the severe case.
SFuelMod4	Fire behavior fuel model given fourth highest weight in the calculations for the severe case.
SFMod4Wt	Weight (%) given to FuelMod4 in the calculations for the severe case.

MFuelMod1	Fire behavior fuel model given highest weight in the calculations for the moderate case. Refer to the FFE documentation for a description.
MFMod1Wt	Weight (%) given to FuelMod1 in the calculations for the moderate case.
MFuelMod2	Fire behavior fuel model given second highest weight in the calculations for the moderate case.
MFMod2Wt	Weight (%) given to FuelMod2 in the calculations for the moderate case.
MFuelMod3	Fire behavior fuel model given third highest weight in the calculations for the moderate case.
MFMod3Wt	Weight (%) given to FuelMod3 in the calculations for the moderate case.
MFuelMod4	Fire behavior fuel model given fourth highest weight in the calculations for the moderate case.
MFMod4Wt	Weight (%) given to FuelMod4 in the calculations for the moderate case.

FUELS VARIABLES

Litter	Amount of litter present (tons/acre)
Duff	Amount of duff present (tons/acre)
SrfDead1	Amount of woody material smaller than 3.0 inches in diameter (tons/acre)
SrfDead2	Amount of all woody material 3.0 inches and greater in diameter (tons/acre). This value is the sum of the three values that follow.
SrfDead3	Amount of woody material 3.0 - 6.0 inches in diameter (tons/acre)
SrfDead4	Amount of woody material 6.0 - 12.0 inches in diameter (tons/acre)
SrfDead5	Amount of woody material 12.0 - 3.0 inches in diameter (tons/acre)
Herb	Amount of herbaceous material (tons/acre)
Shrub	Amount of shrub material (tons/acre)
SrfTotal	Total amount of surface biomass (tons/acre)
StdDead1	Amount of standing dead woody material smaller than 3.0 inches in diameter (tons/acre)
StdDead2	Amount of standing dead woody material 3.0 inches and greater in diameter (tons/acre)
StdLvFol	Amount of foliage material on live standing trees (tons/acre)
StdLive1	Amount of branch and stem material smaller than 3.0 inches in diameter on live standing trees (tons/acre)
StdLive2	Amount of branch and stem material 3.0 inches and greater on live standing trees (tons/acre)
StdLvTot	Total amount of standing biomass (tons/acre) both live and dead
TotalBio	Total amount of biomass (tons/acre) both surface and standing
BioCons	Total amount of biomass (tons/acre) consumed in a fire
BioRemov	Total amount of biomass (tons/acre) removed in a mechanical treatment

Command Line Arguments

The Fuels and Potential Fire Table post processor may be run by typing `firetbl` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-FvsOutputFilename	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. This must be the first command line argument.
-OutputFilename	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the main FVS output file with a `.ftb` extension.
-F	Include the variables from the All Fuels Report.
-P	Include the variables from the Potential Fire Report.

-X Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -F, -P, or -X command line arguments are to be used, the FVS output filename argument and output filename argument must also be used. In other words, the -F, -P, or -X argument may never appear as the first or second command line argument.

Command Line Example:

```
firetbl -example.out -example.ftb -F -P
```

This will run the Fuels and Potential Fire Table post processor using the file `example.out` as input, using the file `example.ftb` for the output, and including both the fuels variables and the potential fire variables in the output.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with what is referred to in *Suppose* as the Compute1 post processor is shown below.

```
//start ppif.fireTable
name:{Fuels and Potential Fire Table}
command{dos}:{
!fvsbin!\\firetbl.exe  -!run!.out -!run!.ftb -f -p}
...
//end ppif.fireTable
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\compute.exe` will become `C:\Fvsbin\firetbl.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.out`, which represents the name of the main FVS output file used as input for the post processor. This is the default name that FVS will give the main FVS output file. The second command line argument would be `-testrun.ftb`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.ftb` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to include only the potential fuel variables, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\firetbl.exe -!run!.out  -!run!.ftb -p
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

FVSStand

Variants: All
Keywords required: **FVSStand**
Program filename: fvsstand.exe

The FVSStand post processor produces yield reports and standard stand and stock tables. The yield reports may be time-dependent or age-dependent. Values in the stand and stock tables may be reported by diameter class or size class. The reports are designed to be imported into forest planning models.

For a complete discussion of the FVSStand post processor please refer to the document Select Topics for the Forest Vegetation Simulator by D.A. Vandendriesche. That document is available from the Documents section of the Forest Vegetation Simulator web site (which is at www.fs.fed.us/fmsc/fvs at the time of production of this document).

Mountain Pine Beetle Risk Calculations

Variants: Developed for UT
Keywords required: **TreeList**
Program filename: PineBtl.exe

The Mountain Pine Beetle Risk Calculations post processor produces an output file with a table containing ponderosa pine mountain pine beetle risk rating factors for a each stand in a simulation. Risk rating factors are calculated for every cycle for which TreeList output is found. The risk rating factors were developed for use in Utah, but it will produce output for any stand in which ponderosa pine is found. A composite score is calculated based on proportion of ponderosa pine, average tree diameter, and stand density. The outbreak potential and suggested timing for preventative action is based on the composite score. Unless otherwise specified as a command line argument, the output filename will have a `.btl` extension.

*The **TreeList** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Mountain Pine Beetle Risk Calculations post processor requires a TreeList file as input. This file is produced by FVS only when the TreeList keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.trl` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

In addition to the TreeList file, the main FVS output file is used as input. It is assumed that the name of the main FVS output file is the same as the name of the TreeList file, except the filename extension is `.out`. For example, if the TreeList file is named `testrun.trl` the main FVS output file is assumed to be named `testrun.out`.

The output file will have the same base name as the TreeList file used as input, but will have a `.btl` extension. For example, if the file `testrun.trl` is used as input, the output file will be named `testrun.btl`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

The user may enter the number of trees per acre that are currently infested with mountain pine beetles. This is an important component of the risk rating system, but this information is not available from the FVS output. If nothing is entered by the user it is assumed that none of the trees in the stand are infested. The number that is entered is used for all cycles, and for all stands in the simulation.

Output

In order to produce meaningful output, the Mountain Pine Beetle Risk Calculations post processor requires a valid TreeList file and main FVS output file as input. The TreeList file is produced by FVS only when the TreeList keyword is included in the simulation. If the TreeList keyword is not included in the simulation the TreeList file will be empty and the post processor will report an error.

A sample output table is shown below, followed by a description of the types of values it contains.

```
                    FOREST VEGETATION SIMULATOR
                 MOUNTAIN PINE BEETLE RISK RATING
                    Simulation: test

Ratings for individual characteristics are shown in parentheses.  Overall stand
risk and outbreak potential are shown at the far right.  The number of currently
infested trees per acre was supplied as an input and does not change through time.

Stand:   108103.00020000000000000026
Mgmt Id: NONE
        | TPA curr. | PP QMD    | Basal   |     Canopy        | Stand    Outbreak
   Year | Infested  | >5" dbh   | Area    | Min Ht |  % PP    | Risk     Potential
   _____|___________|___________|_________|________|__________|_____________________
   1985 |   5.0 (2)    19.2 (3)    29 (1)     35     28.1 (1) |   7      Moderate
   1995 |   5.0 (2)    20.1 (3)    24 (1)     36     28.1 (1) |   7      Moderate
   2005 |   5.0 (2)    20.9 (3)    13 (1)     38     28.3 (1) |   7      Moderate
   2015 |   5.0 (2)    21.6 (3)    15 (1)     38     29.0 (1) |   7      Moderate
```

DESCRIPTION OF THE OUTPUT

Stand	26-character stand identification code
Mgmt Id	Management code assigned using the MgmtId keyword
Year	Year in which the FVS cycle began
TPA Curr. Infested	Number of trees per acre that are currently infested with mountain pine beetle. This number is entered as input by the user, and remains constant through time. The associated risk value is shown in parentheses based on the following criteria. 1 = less than 3 infested trees per acre 2 = 3-10 infested trees per acre 3 = more than 10 infested trees per acre
PP QMD >5" dbh	Quadratic mean diameter (inches) of all ponderosa pine that are at least 5 inches DBH. The associated risk value is shown in parentheses based on the following criteria. 1 = less than 6.0 inches 2 = 6.0 - 12.0 inches 3 = greater than 12.0 inches
Basal Area	Total basal area of the stand (ft2/acre) for all trees that are at least 5 inches DBH. The associated risk value is shown in parentheses based on the following criteria 1 = less than 80 square feet per acre 2 = 80 - 120 square feet per acre 3 = greater than 120 square feet per acre
Canopy Min Ht	Minimum height (feet) that a tree must be in order to be considered part of the canopy. This height is 75% of the top height (average height of the 40 largest diameter trees per acre) in the stand.
Canopy % PP	Percentage of ponderosa pine in the canopy in terms of trees per acre. The associated risk value is shown in parentheses based on the following criteria 1 = less than 50% 2 = 50 - 65% 3 = greater than 65%
Stand Risk	A composite risk rating found by summing the four risk values assigned for TPA Curr Infested, PP QMD >5" dbh, Basal Area, and Canopy % PP. The range of possible values is 4 - 12.
Outbreak Potential	Rating indicating the relative potential for an outbreak of mountain pine beetles in the stand. This is based on the value of the Stand Risk as follows. Stand Risk — Outbreak Potential 4-5 — Low 6-9 — Moderate 10-12 — High

Command Line Arguments

The Mountain Pine Beetle Risk Calculations post processor may be run by typing `btlrsk` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*FvsOutputFilename*	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. This must be the first command line argument.
-*TreeListFilename*	The name of the TreeList file that is to be used as input. This file usually has a `.trl` extension. The actual filename is used in place of the designation *TreeListFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the TreeList file name. If this argument is not used, the TreeList file will be assumed to have the same base name as the main FVS output file with a `.trl` extension.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the third command line argument. If there are more than two command line arguments, the third will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the main FVS output file with a `.btl` extension.
-I#	Number currently infested, where the # is the number of trees per acre currently infested with mountain pine beetle. If this argument is not used it is assumed that none of the trees are infested.
-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -I# or -X command line arguments are to be used, the FVS output filename argument, TreeList filename argument and output filename argument must also be used. In other words, the -I# and -X argument may never appear as the first, second, or third command line argument.

Command Line Example:

```
pinebtl -example.out -example.trl -example.btl -i7
```

This will run the Mountain Pine Beetle Risk Calculations post processor using the files `example.out` and `example.trl` as input, using the file `example.btl` for the output, and specifying 7 trees per acre as currently infested.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Mountain Pine Beetle Risk Calculations post processor is shown below.

```
//start ppif.btlpine
name:{Mountain Pine beetle Risk Calculations}
command{dos}:{
```

```
!fvsbin!\\pinebtl.exe -!run!.out -!run!.trl -!run!.btl}
...
//end ppif.btlpine
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\pinebtl.exe` will become `C:\Fvsbin\pinebtl.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.out`, which represents the name of the main FVS output file used as input for the post processor. This is the default name that FVS will give the main FVS output file. The second command line argument would be `-testrun.trl`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.btl` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to specify 5 currently infested trees per acre, the command line should be written as below. The curly bracket would follow immediately.

```
!fvsbin!\\pinebtl.exe -!run!.out -!run!.trl  -!run!.btl -i5
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Multistory Elk Hiding Cover

Variants: All
Keywords required: **TreeList**
Program filename: elkcover.exe

The Multistory Elk Hiding Cover post processor produces an output file containing tables of information on the average amount of an elk that will be hidden by the overstory and by the understory at a specified viewing distance. Calculations are based on research by F.W. Smith and J.N. Long. Values are calculated for every cycle for which TreeList output is found. Unless otherwise specified as a command line argument, the output filename will have a `.elk` extension.

*The **TreeList** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Multistory Elk Hiding Cover post processor requires a TreeList file as input. This file is produced by FVS only when the TreeList keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.trl` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

The output file will have the same base name as the TreeList file used as input, but will have a `.elk` extension. For example, if the file `testrun.trl` is used as input, the output file will be named `testrun.elk`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

Output

In order to produce meaningful output, the Multistory Elk Hiding Cover post processor requires a valid TreeList file as input. The TreeList file is produced by FVS only when the TreeList keyword is included in the simulation. If the TreeList keyword is not included in the simulation the TreeList file will be empty and the post processor will report an error.

A sample output table is shown below, followed by a description of the types of values it contains.

```
MULTISPECIES, MULTISTORY ELK HIDING COVER MODEL  -  VERSION 2.0
          DEVELOPED BY F. W. SMITH AND J. N. LONG
                    JANUARY 20, 1987
```

```
STAND ID:  Stand308                  MGMTID: NONE   DATE: 01-01-2003   TIME: 12:34:56   SM  6.3
**********  CYCLE    0  **********

BOTH AN OVERSTORY AND UNDERSTORY EXISTS
OVERSTORY DENSITY =    596 TPA (TREES WITH BASE OF CROWN HIGHER THAN 4.5 FT)
UNDERSTORY DENSITY  =     41 TPA (TREES AT LEAST 7 FT TALL WITH BASE OF CROWN 4.5 FT OR LOWER)



REPLICATION   1
OBSERVATION POINTS:   8
SIGHTING DISTANCE: 200 FT

   THE OVERSTORY STAND STATISTICS ARE --
     TPA --  596   ASD --    8.0   SDI --  414.   BASAL AREA --  206.3
     SUM OF DBH --   3959. INCHES
   THE UNDERSTORY STAND STATISTICS FOR TREES AT LEAST 7 FEET IN HEIGHT ARE --
     TPA --   41   ASD --    6.0  SUM OF CROWN DIA --     409. FEET


   PERCENT OF AN ELK HIDDEN 200 FEET FROM THE DIFFERENT OBSERVATION POINTS

                            -- OBSERVATION POINT --
                     1    2    3    4    5    6    7    8    9   10

     BY OVERSTORY    68.  99. 100.  71. 100.  62.  41.  85.
     BY UNDERSTORY    0.  90.  89.  95. 100.  95.   0.  85.
     BY BOTH         68. 100. 100.  98. 100.  97.  41.  99.

     THE AVERAGE HIDDEN BY THE OVERSTORY IS                --   84.8
     THE 95% CI IS --                           60.5  TO    98.5
     THE AVERAGE HIDDEN BY THE UNDERSTORY IS               --   69.7
     THE 95% CI IS --                           20.3  TO    99.6
     THE AVERAGE HIDDEN BY THE OVERSTORY AND UNDERSTORY IS --   94.5
     THE 95% CI IS --                           76.2  TO    99.9


  PERCENT OF OBSERVATION POINTS AT WHICH AN ELK IS HIDDEN BY SET AMOUNTS AT 200 FEET

                                             -- % HIDDEN --
                  0 - 9  10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90+

     BY OVERSTORY     0.     0.     0.     0.    13.     0.    25.    13.    13.    38.
     BY UNDERSTORY   25.     0.     0.     0.     0.     0.     0.     0.    25.    50.
     BY BOTH          0.     0.     0.     0.    13.     0.    13.     0.     0.    75.
```

DESCRIPTION OF THE OUTPUT

Stand Id	26-character stand identification code
Mgmt Id	4-character management code assigned using the MgmtId keyword
Date	Date the FVS simulation was run. This is read from the TreeList file.
Time	Time the FVS simulation was run. This is read from the TreeList file.
Cycle	FVS cycle. The calculations are done at the end of the cycle. Cycle 0 represents the inventory year.
Overstory Density	Trees per acre with a height to the base of the live crown greater than 4.5 feet. This is displayed only if there are trees that meet this criterion. Overstory trees obscure the view of an elk only with their boles.
Understory Density	Trees per acre that are at least 7 feet tall, and that have a height to the base of the live crown of 4.5 feet or less. This is displayed only if there are trees that meet these criteria. Understory trees obscure the view of an elk with their crowns.
Replication	Number of the current replication. Replications are necessary due to the way a random component is built into the calculations. Increasing the number of replications allows for better analysis of the range of possible values. The number of replications is specified as a program option.
Observation Points	Number of observation points that were used for the purposes of the calculations. They simulate particular points throughout the stand where field observations would be taken. The number of observation points is specified as a program option.
Sighting Distance	Distance from the observations points that an elk is assumed to be. All calculations are based on the amount of an elk at that distance that would be hidden from view of a person standing at the observation point. This value is fixed at 200 feet.
TPA	Trees per acre categorized as overstory or understory. See the description of Overstory Density and Understory Density above for the criteria used.

ASD	Average (quadratic mean) diameter at breast height (inches) for trees categorized as overstory or understory. See the description of Overstory Density and Understory Density above for the criteria used.
SDI	Stand density index for overstory trees. See the description of Overstory Density above for the criterion used.
Basal Area	Basal area (ft^2/acre) for overstory trees. See the description of Overstory Density above for the criterion used.
Sum of Crown Dia	Sum of the widths of the crowns (feet) of all understory trees in one acre of the stand. See the description of Understory Density above for the criteria used.

PERCENT OF AN ELK HIDDEN 200 FEET FROM THE DIFFERENT OBSERVATION POINTS

By Overstory	Percent of an elk standing 200 feet from the observation point that would be hidden by boles of overstory trees. See the description of Overstory Density above for the criterion used.
By Understory	Percent of an elk standing 200 feet from the observation point that would be hidden by crowns of understory trees. See the description of Understory Density above for the criteria used.
By Both	Percent of an elk standing 200 feet from the observation point that would be hidden by boles of overstory trees and crowns of understory trees. See the description of Overstory Density and Understory Density above for the criteria used. Due to overlap between overstory and understory trees as seen from the observation point, this value may be less than the sum of the overstory and understory values.
Average Hidden	Percent of an elk standing 200 feet from the observation point that would be hidden by boles of overstory trees (Hidden By Overstory), crowns of understory trees (Hidden By Understory), or both (Hidden By Overstory And Understory). These calculations are independent of the calculations made for each observation point, so they may not represent the average of the values displayed in the table.
95% CI	Range of values representing the 95 percent confidence interval for the percent-hidden values of the part of the stand (overstory, understory, or both) represented on the line above it. The 95 percent confidence interval information is printed only when there are 5 or more observation points.

PERCENT OF OBSERVATION POINTS AT WHICH AN ELK IS HIDDEN BY SET AMOUNTS

By Overstory	The values represent the percentage of the observation points at which the percent of an elk hidden by the overstory (from the Percent Of Elk Hidden table) fell into the associated %-Hidden range.
By Understory	The values represent the percentage of the observation points at which the percent of an elk hidden by the understory (from the Percent Of Elk Hidden table) fell into the associated %-Hidden range.
By Both	The values represent the percentage of the observation points at which the percent of an elk hidden by both the overstory and understory (from the Percent Of Elk Hidden table) fell into the associated %-Hidden range.

Command Line Arguments

The Multistory Elk Hiding Cover post processor may be run by typing `elkcover` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*TreeListFilename*	The name of the TreeList file that is to be used as input. This file usually has a `.trl` extension. The actual filename is used in place of the designation *TreeListFilename* shown here. This must be the first command line argument.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the TreeList file with a `.elk` extension.
-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -X command line argument is to be used, the TreeList filename argument and output filename argument must also be used. In other words, the -X argument may never appear as the first or second command line argument.

Command Line Example:

```
elkcover -example.trl -example.elk
```

This will run the Multistory Elk Hiding Cover post processor using the file `example.trl` as input, and using the file `example.elk` for the output. All other arguments will use the default settings: all cycles will be processed, both the overstory and understory will be assumed randomly spaced, there will be three replications in each stand, 10 observations per replication, and a viewing distance of 200 feet.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Multistory Elk Hiding Cover post processor is shown below.

```
//start ppif.elkCov
name:{Multistory Elk Hiding Cover}
command{dos}:{
!fvsbin!\\elkcover.exe  -!run!.trl -!run!.elk}
...
//end ppif.elkCov
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\elkcover.exe` will become `C:\Fvsbin\elkcover.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.trl`, which represents the name of the TreeList file used as input for the post processor. This is the default name that FVS will give the TreeList file. The second command line argument would be `-testrun.elk`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.elk` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to write the output to a file with a `.xyz` extension, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\elkcover.exe -!run!.trl  -!run!.xyz
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Rocky Mountain VSS

Variants: CI, CR, TT, UT
Keywords required: **TreeList**
Program filename: rmvsssp.exe

The Rocky Mountain VSS (Vegetation Structural Stage) post processor produces an output file containing tables with a breakdown by structural class and stand density index rating. The tables are produced for each cycle for which TreeList output is found. Unless otherwise specified as a command line argument, the output filename will have a `.vss` extension.

*The **TreeList** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Rocky Mountain Vegetation Structural Stage post processor requires a TreeList file as input. This file is produced by FVS only when the TreeList keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.trl` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

The output file will have the same base name as the TreeList file used as input, but will have a `.vss` extension. For example, if the file `testrun.trl` is used as input, the output file will be named `testrun.vss`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

Output

In order to produce meaningful output, the Rocky Mountain Vegetation Structural Stage post processor requires a valid TreeList file as input. The TreeList file is produced by FVS only when the TreeList keyword is included in the simulation. If the TreeList keyword is not included in the simulation the TreeList file will be empty and the post processor will report an error.

Sample output tables are shown below, followed by a description of the types of values they contain.

```
Loc - Site 001002-0003   Region 3             Year 2000   Cycle    0

 ********************** VEGETATION STRUCTURAL STAGE ************************
 Note: SDI for STRUCTURE STAGE is a summation of SDI by 1" diameter classes
       SDI is based on all live trees 1"+ dbh in all calculations
 ***************************************************************************
```

```
 FOREST COVER TYPE =  PP          MAX SDI FOR TYPE =  450.0
  STAND SITE INDEX =   0.               STAND SDI =   44.5
                                % SDI OF MAX SDI =    9.89

                   ****************** STRUCTURAL CLASSES ****************
                    (1)     (2)      (3)       (4)      (5)        (6)
                   GRASS   SEED/    YOUNG     MID-AGE   MATURE      OLD
                   FORBS   SAPS    FOREST     FOREST   FOREST     GROWTH

BEGIN CLASS DBH      0.0    1.0       5.0      12.0      18.0      14.0

REQUIRED TREES                                                    20.0

# TREES IN CLASS            0.0       0.0      13.6       6.7       9.7

BA FOR CLASS                0.0       0.0      15.0      15.0

QMD OF CLASS                0.0       0.0      14.2      20.2

SDI IN CLASS                0.0       0.0      23.8      20.7

% SDI IN CLASS             0.00      0.00     53.38     46.62


STRUCTURE STAGE = 1


                      STAND DENSITY INDEX RATING

                *          DIAMETER GROUPS                     *
        POINTS     1- 4.9    5-11.9   12-17.9   18-23.9       24+

# TREES                0.0       0.0      13.6       6.7       0.0
BA                     0.0       0.0      15.0      15.0       0.0
QMD                    0.0       0.0      14.2      20.2       0.0
SDI                    0.0       0.0      23.8      20.7       0.0
%SDI                   0.0       0.0      53.4      46.6       0.0
TOTAL         1
LT 15%SDI     1

SDI RATING = 99.00550.10
```

DESCRIPTION OF THE OUTPUT

Loc-Site	Location and site identification code
Region	Forest Service region number
Year	Year in which the FVS cycle began
Cycle	FVS cycle number
Forest Cover Type	Regional forest cover type based on plurality of basal area by species group
Stand Site Index	Site index read from the main FVS output file. Site index is specified in the SiteCode keyword record, which the *Suppose* program builds from information in the stand list file.
Max SDI for Type	Theoretical maximum stand density index (SDI) for the designated forest cover type
Stand SDI	Stand density index (SDI) calculated for all trees at least 1.0 inches DBH in the stand
% SDI of Max SDI	Percent of theoretical maximum stand density index (SDI) that the calculated SDI represents

STRUCTURAL CLASSES

Grass Forbs	Represents the range of diameters at breast height for the smallest vegetation. The diameter range begins at the size shown in the Begin Class DBH for this class, and ends at the size for the Seed/Saps class.
Seed/Saps	Represents the range of diameters at breast height for seedlings and saplings. The diameter range begins at the size shown in the Begin Class DBH for this class, and ends at the size for the Young Forest class.
Young Forest	Represents the range of diameters at breast height for a young forest. The diameter range begins at the size shown in the Begin Class DBH for this class, and ends at the size for the Mid-Age Forest class.
Mid-Age Forest	Represents the range of diameters at breast height for a mid-age forest. The diameter range begins at the size shown in the Begin Class DBH for this class, and ends at the size for the Mature Forest class.
Mature Forest	Represents the range of diameters at breast height for a mature forest. The diameter range includes everything that is at least the size shown in the Begin Class DBH for this class.
Old Growth	Represents old growth, which has minimum requirements for diameter at breast height and trees per acre. The minimum DBH is shown in the Begin Class DBH for this class, and the minimum trees per acre is shown in the Required Trees for this class.

Begin Class DBH	Minimum diameter at breast height (inches) that trees must have in order to be classified in the different structural classes. The range of diameter for the class runs up to, but does not include, the diameter specified for the next class. The values are regional standards.
Required Trees	Minimum number of trees per acre that must meet the requirements for old growth for the stand to be classified as old growth. These trees must have a diameter at breast height of at least the value shown in the Begin Class DBH for old growth. The required number and the minimum diameter are regional standards.
# Trees In Class	Number of trees per acre that fall into the diameter range specified for the structural class. Diameter ranges are specified by the Begin Class DBH values.
BA For Class	Total basal area (ft^2/acre) for all trees that fall into the structural class diameter limits
QMD Of Class	Quadratic mean diameter for all trees that fall into the structural class diameter limits
SDI In Class	Stand density index calculated using all the trees that fall into the stuctural class diameter limits
% SDI In Class	Percent of the sum of the SDI calculations represented by the SDI for the individual structural class

Structural Stage — Vegetation structural stage (VSS) code calculated for the stand. It is a combination of up to three codes: structural stage, canopy cover, and stories. The VSS number corresponds to the number in parentheses above the name of the structural classes at the top of the table. The value is assigned as outlined below.

1	Total stand SDI less than 10% of Max SDI for the forest type, or basal area less than 20 ft^2/acre
2, 3, 4	Total stand SDI at least 10% of Max SDI for the forest type, and minimum number of old growth trees not met. The stand structural class is assigned based on the class with the highest basal area.
6	The number of trees meeting old growth specifications is at least the minimum number required.

Canopy cover is the next part of the code. It is assigned for VSS values 2 and greater as follows.

A	Open (0 - 40% cover). Total stand SDI is 10 - 30% of theoretical maximum SDI for the forest type.
B	Moderately closed (40 - 60% cover). Total stand SDI is 30 - 47% of theoretical maximum SDI for the forest type.
C	Closed (60% or greater cover). Total stand SDI is greater than 47% of theoretical maximum SDI for the forest type.

Story code is the last part of the code. It is assigned for VSS values 3 and greater as follows.

SS	Single story. SDI for selected VSS is at least 60% of total stand SDI.
MS	Multiple story. SDI for selected VSS is less than 60% of total stand SDI.

STAND DENSITY INDEX RATING

Command Line Arguments

The Rocky Mountain Vegetation Structural Stage post processor may be run by typing `rmvss` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-TreeListFilename	The name of the TreeList file that is to be used as input. This file usually has a `.trl` extension. The actual filename is used in place of the designation *TreeListFilename* shown here. This must be the first command line argument.
-FvsOutputFilename	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the main FVS output file name. If this argument is not used, the main FVS output file will be assumed to have the same base name as the TreeList file with a `.out` extension.

-*OutputFilename* The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the third command line argument. If there are more than two command line arguments, the third will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the TreeList file with a `.vss` extension.

-X Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -X command line argument is to be used, the TreeList filename argument, FVS output filename argument, and output filename argument must also be used. In other words, the -X argument may never appear as the first, second, or third command line argument.

Command Line Example:

```
rmvss -example.trl -example.out -example.vss
```

This will run the Rocky Mountain Vegetation Structural Stage post processor using the files `example.trl` and `example.out` as input, and using the file `example.vss` for the output.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Rocky Mountain Vegetation Structural Stage post processor is shown below.

```
//start ppif.rmvss
name:{Rocky Mountain Vegetation Structural Stage}
command{dos}:{
!fvsbin!\\rmvsssp.exe -!run!.trl -!run!.out -!run!.vss}
...
//end ppif.rmvss
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\rmvss.exe` will become `C:\Fvsbin\rmvss.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.trl`, which represents the name of the TreeList file used as input for the post processor. This is the default name that FVS will give the TreeList file. The second command line argument would be `-testrun.out`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.vss` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to write the output to a file with a `.xyz` extension, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\rmvsssp.exe -!run!.trl -!run!.out -!run!.xyz
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Spectrum Export Table

Variants: All
Keywords required: **Compute**
Program filename: fvs2spec.exe

Spectrum is a software package designed to aid in the development of forest plans. It contains an optimization processor that uses linear programming to derive an optimal solution based on restraints that the user specifies.

The Spectrum Export Tables post processor produces an output file containing a table of values for the Compute variables for all stands in the simulation. The output table is formatted so that it can be imported into the Spectrum forest planning software. Variables in the table are identified by strata, year, and stand age. Unless otherwise specified as a command line argument, the output filename will have a `.spc` extension.

*The **Compute** keyword with valid variable definitions must be included in the FVS simulation file.* The simulation should also be set up specifically to be used as input for the Spectrum model.

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

 The Spectrum Export Tables post processor requires a main FVS output file as input. This file must contain information that was calculated using the Compute keyword in the simulation. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

 The output file will have the same base name as the main FVS output file used as input, but will have a `.spc` extension. For example, if the file `testrun.out` is used as input, the output file will be named `testrun.spc`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

 There are three methods that may be used to weight the stands when calculating average values. Each is described below. These options may be specified by selecting one of the weighting methods in the Average menu, or by clicking on one of the weight buttons on the toolbar. The output file may also be specified on the command line.

The first weighting method is to give all stands equal weight when calculating the averages. Values are simply summed across all stands in the simulation, and the result divided by the number of stands. This is the default weighting method.

The second weighting method is to weight stands by stand sampling weight when calculating the averages. Values are multiplied by the sampling weight, and the results are then summed. The total is divided by the total sampling weight for all stands combined. Sampling weight usually represents stand acreage. It is specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file.

The third weighting method is to weight stands by number of plots in the stand inventory. What FVS considers plots may be referred to as inventory points or subplots, depending on the terminology used in the inventory procedures and the unit that was specified as a

stand for the purposes of FVS. Values are multiplied by the number of plots in the stand, and the results are then summed. The total is divided by the total number of plots in all stands combined. The number of plots may specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file. If this information is not provided, FVS counts the number of unique plot identification codes found in the tree data file for the stand.

Output

The output from the Spectrum Export Table post processor is formatted so that it can be easily imported into the Spectrum forest planning software. The FVS simulation must have been designed with the requirements of the Spectrum model in mind. There are specific types of information that are needed by that model. FVS can provide much of the required information through the use of computed variables.

In order to produce meaningful output, the Spectrum Output Table post processor requires a valid main FVS output file as input. The Activity Summary must contain computed variables for which a successful calculation was done. These variables must be defined using the Compute keyword. If no valid computed variables appear in the Activity Summary the post processor will report an error.

A sample output table is shown below, followed by a description of the types of values it contains. The variables that are to be displayed are program options specified either in the interface to the Spectrum Export Table post processor, or on the command line used to launch the post processor.

STRATA	YEAR	AGE	FIRSTVAR	THRDVAR	SECNDVAR
testrun	2005	7	85.43	2536.24	0.00
testrun	2015	8	91.34	2968.74	514.26
testrun	2025	9	62.58	3489.25	0.00

DESCRIPTION OF THE OUTPUT

Strata — Stratum name that will be used in the Spectrum program. This is the name of the main FVS output file without the `.out` extension.

Year — Year in which the FVS cycle began. The Spectrum program expects the values that are reported to be at the midpoint of the decade for which it will do its calculations. If the decade of interest in Spectrum is from 2000 to 2010, the values should be reported in 2005. The FVS simulation must be designed with this in mind.

Age — Average stand age for all stands in the simulation, reported for the year shown in YEAR, and rounded to the nearest decade. For example, an average stand age of 74 will be assigned an AGE value of 7, whereas an average stand age of 75 will be assigned an AGE value of 8. Stand age is specified in the Design keyword record, which the *Suppose* program builds from information in the stand list file (where stand age at inventory is calculated as the difference between stand origin year and inventory year).

All other columns are headed with the computed variable name, and contain the average value calculated for that variable from all stands in the simulation in which that variable was successfully computed and displayed in the Activity Summary of the main FVS output file. In the example above, FIRSTVAR, THRDVAR, and SECNDVAR are the computed variable names that were found. The average values are calculated using the weighting method that was specified as a program option, however there is no indication in the output as to the method that was used.

A value of 0.00 can mean one of two things. Either the actual average value for that variable is 0.00 for that cycle, or there were no stands in which that variable was successfully computed that cycle. No distinction is made in the output between these two cases. In the example above, the variable SECNDVAR was computed only in the cycle beginning in the year 2015, but there is no way to determine that from the output.

Command Line Arguments

The Spectrum Export Table post processor may be run by typing `fvs2spec` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-FvsOutputFilename	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. This must be the first command line argument.
-OutputFilename	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the main FVS output file with a `.spc` extension.
-F	A file contains the names of the variables to be processed. The next command line argument will be read as the filename. The file must be a text file containing a list of variable names, one on each line in the file.

-W#	Weighting method, where the # is a number representing the weighting method. All of the possible weighting designations are shown below. The default method is equal weight.
	W1 = equal weight
	W2 = weight by stand weight
	W3 = weight by number of points

-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -F,-W#, or -X command line arguments are to be used, the FVS output filename argument and output filename argument must also be used. In other words, the -F,-W#, or -X argument may never appear as the first or second command line argument.

Command Line Example:

```
fvs2spec -example.out -example.spc -F -varnames.txt
```

This will run the Spectrum Export Tables post processor using the file `example.out` as input, using the file `example.spc` for the output, and processing only those variables whose names are read from the file `varnames.txt`.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Spectrum Export Table post processor is shown below.

```
//start ppif.spectrum
name:{SPECTRUM Export Tables}
command{dos}:{
!fvsbin!\\fvs2spec.exe  -!run!.out  -!run!.spc}
...
//end ppif.spectrum
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\fvs2spec.exe` will become `C:\Fvsbin\fvs2spec.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.out`, which represents the name of the main FVS output file used as input for the post processor. This is the default name that FVS will give the main FVS output file. The second command line argument would be `-testrun.spc`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.spc` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to write the output to a file with a `.xyz` extension, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\fvs2spec.exe -!run!.out  -!run!.xyz
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Spruce Beetle Risk Rating

Variants: Any with Engelmann spruce
Keywords required: **TreeList**
Program filename: sprucbtl.exe

The Spruce Beetle Risk Rating post processor produces an output file containing a table containing risk rating factors for spruce beetle in Engelmann spruce for a each stand in a simulation. Risk rating factors are calculated for every cycle for which TreeList output is found. The risk rating factors were developed based on information in Schmid, J.M. and R.H. Frye, 1976, Stand Ratings for Spruce Beetles, Research Note RM-309. A composite score is calculated based on site index, basal area, average spruce diameter, and proportion of spruce in the canopy. The outbreak potential is based on the composite score. Unless otherwise specified as a command line argument, the output filename will have a `.sbr` extension.

*The **TreeList** keyword must be included in the FVS simulation file.*

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

The Spruce Beetle Risk Rating post processor requires a TreeList file as input. This file is produced by FVS only when the TreeList keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.trl` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

In addition to the TreeList file, the main FVS output file is used as input. It is assumed that the name of the main FVS output file is the same as the name of the TreeList file, except the filename extension is `.out`. For example, if the TreeList file is named `testrun.trl` the main FVS output file is assumed to be named `testrun.out`.

The output file will have the same base name as the TreeList file used as input, but will have a `.sbr` extension. For example, if the file `testrun.trl` is used as input, the output file will be named `testrun.sbr`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

Output

In order to produce meaningful output, the Spruce Beetle Risk Rating post processor requires a valid TreeList file and main FVS output file as input. The TreeList file is produced by FVS only when the TreeList keyword is included in the simulation. If the TreeList keyword is not included in the simulation the TreeList file will be empty and the post processor will report an error.

A sample output table is shown below, followed by a description of the types of values it contains.

```
                    FOREST VEGETATION SIMULATOR
                     SPRUCE BEETLE RISK RATING
                     Simulation: test

Ratings for individual characteristics are shown in parentheses.
Overall stand risk and outbreak potential are shown at the far right.
If the stand is in a well-drained creek bottom, the site index rating should be
increased to 3, and the stand risk adjusted accordingly.

Stand:   Stand403
Mgmt Id: NONE
        | ES Site | Spruce QMD | Basal    |     Canopy        | Stand    Outbreak
   Year | Index   | >10" dbh   | Area     | Min Ht | % Spruce | Risk     Potential
   _____|_________|____________|__________|________|__________|_____________________
   2000 |  82 (2)    14.0 (2)    211 (3)     69     71.0 (3) |  10      High-moderate
   2010 |  82 (2)    14.6 (2)    219 (3)     70     72.3 (3) |  10      High-moderate
```

DESCRIPTION OF THE OUTPUT

Stand	26-character stand identification code
Mgmt Id	Management code assigned using the MgmtId keyword
Year	Year in which the FVS cycle began
ES Site Index	Site index for Engelmann spruce in the stand. The associated risk value is shown in parentheses based on the following criteria. 1 = Engelmann spruce site index less than 80 2 = Engelmann spruce site index 80 or greater 3 = well-drained creek bottom. Since FVS does not have any information about whether the stand is in a well-drained creek bottom, the value assigned in this category may be inaccurate. If the stand is actually in a well-drained creek bottom the value shown for ES Site Index should be raised to 3, and the Stand Risk and Outbreak Potential adjusted accordingly.
Spruce QMD >10" dbh	Quadratic mean diameter (inches) of all Engelmann spruce that are at least 10 inches DBH. The associated risk value is shown in parentheses based on the following criteria. 1 = less than 12.0 inches 2 = 12.0 - 16.0 inches 3 = greater than 16.0 inches
Basal Area	Total basal area of the stand (ft2/acre). The associated risk value is shown in parentheses based on the following criteria 1 = less than 100 square feet per acre 2 = 100 - 150 square feet per acre 3 = greater than 150 square feet per acre
Canopy Min Ht	Minimum height (feet) that a tree must be in order to be considered part of the canopy. This height is 75% of the top height (average height of the 40 largest diameter trees per acre) in the stand.
Canopy % Spruce	Percentage of Engelmann spruce in the canopy in terms of trees per acre. The associated risk value is shown in parentheses based on the following criteria 1 = less than 50% 2 = 50 - 65% 3 = greater than 65%
Stand Risk	A composite risk rating found by summing the four risk values assigned for ES Site Index, Spruce QMD >10" dbh, Basal Area, and Canopy % Spruce. The range of possible values is 4 - 12.
Outbreak Potential	Rating indicating the relative potential for an outbreak of spruce beetles in the stand. This is based on the value of the Stand Risk as follows.

Stand Risk	Outbreak Potential
4-5	Low
6	Low-Moderate
7-9	Moderate
10	High-Moderate
11-12	High

Command Line Arguments

The Spruce Beetle Risk Rating post processor may be run by typing `sprucbtl` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*FvsOutputFilename*	The name of the main FVS output file that is to be used as input. This file usually has a `.out` extension. The actual filename is used in place of the designation *FvsOutputFilename* shown here. This must be the first command line argument.
-*TreeListFilename*	The name of the TreeList file that is to be used as input. This file usually has a `.trl` extension. The actual filename is used in place of the designation *TreeListFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the TreeList file name. If this argument is not used, the TreeList file will be assumed to have the same base name as the main FVS output file with a `.trl` extension.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the third command line argument. If there are more than two command line arguments, the third will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the main FVS output file with a `.sbr` extension.
-X	Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -X command line argument is to be used, the FVS output filename argument, the TreeList filename argument, and output filename argument must also be used. In other words, the -X argument may never appear as the first, second, or third command line argument.

Command Line Example:

```
sprucbtl -example.out -example.trl
```

This will run the Spruce Beetle Risk Rating post processor using `example.out` as the file containing the main FVS output, using the file `example.trl` as the TreeList input, and using the default file `example.sbr` for the output.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Spruce Beetle Risk Rating post processor is shown below.

```
//start ppif.btlspruc
name:{Spruce Beetle Risk Rating}
command{dos}:{
!fvsbin!\\sprucbtl.exe -!run!.out}
...
```

```
//end ppif.btlspruc
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\sprucbtl.exe` will become `C:\Fvsbin\sprucbtl.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.out`, which represents the name of the main FVS output file used as input for the post processor. This is the default name that FVS will give the main FVS output file. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.out` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to write the output to a file with a `.xyz` extension, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\sprucbtl.exe -!run!.out -!run!.xyz
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Stand and Stock Tables

Variants: All
Keywords required: **TreeList, CutList**
Program filename: standtsp.exe

The Stand and Stock Tables post processor produces an output file containing stand and stock tables by cycle and species for all stands in the simulation file. Unless otherwise specified as a command line argument, the output filename will have a `.stb` extension.

*A **TreeList** and/or **CutList** keyword must be included in the FVS simulation file.* The TreeList keyword will provide information about the stand prior to harvest, and will also provide mortality information. The CutList keyword will provide the harvest information.

Program options may be specified through command line arguments.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call the Average Summary Table post processor.

Program Options

 The Stand and Stock Tables post processor requires a TreeList file as input. This file is produced by FVS only when the TreeList keyword is included in the simulation. It will typically have the same base name as the simulation file with a `.trl` extension. The input filename can be selected using the Open/Run option in the File menu, or by clicking on the open folder and gear button on the toolbar. An input filename can also be specified on the command line. When the file is opened it is immediately processed and the output is saved to a file.

 The output file will have the same base name as the TreeList file used as input, but will have a `.stb` extension. For example, if the file `testrun.trl` is used as input, the output file will be named `testrun.stb`. The output that is displayed may be saved to a different file using the SaveAs option in the File menu, or by clicking on the diskette button on the toolbar. A different output filename can also be specified on the command line.

 There are three main options for the output table type. Each is described below. The table type may be specified by selecting one of the types in the Table menu, or by clicking on one of the table-type buttons on the toolbar. The table type may also be specified on the command line in conjunction with the table format.

The first table type produces standard tables for all species combined. For every cycle that TreeList file contains information, a table is produced that contains all of the reported attributes (cubic foot volume and trees per acre, for example) with the values listed by size class for all species combined. There is a section in every table for live trees, harvested trees, and mortality trees.

The second table type produces standard tables for all species combined (identical to those for the first table type), and in addition produces tables that contain the same information for each individual species present in the stand. For every cycle that TreeList file contains information, a table is produced that contains all of the reported attributes (cubic foot volume and trees per acre, for example) with the values listed by size class for all species combined. Separate tables are then produced for each individual species. There is a section in every table for live trees, harvested trees, and mortality trees.

The third table type produces volume tables. For every cycle that the TreeList file contains information, a separate table is produced for each volume type (board feet per acre, for example) with the values listed by size class for all species combined, and then for each of the individual species present.

 The user may specify the size of the diameter classes that are to be displayed in the output tables. The specified class size must be a whole number of inches, so no decimal point is allowed. The class size may be specified by selecting Class Size... from the Table menu, or by clicking on the double-arrow button on the toolbar. The class size may also be specified on the command line.

 There are two options for the format of the tables, each of which is described below. The table format may be specified by selecting one of the formats in the Table menu, or by clicking on one of the format buttons on the toolbar. The format may also be specified on the command line.

The first option is a standard tabular format where values are in columns below the attribute name, and table sections are separated by borders. Additional descriptive information is included above the tables.

The second option is a comma delimited format where items are separated by commas. A single header record is printed at the top of the output. This format is typically used to when the values are to be imported into another program, like a spreadsheet or database. Any information that is included is part of the table itself.

Output

In order to produce meaningful output, the Stand and Stock Tables post processor requires a valid TreeList file as input. The TreeList file is produced by FVS only when the TreeList keyword or CutList keyword is included in the simulation. If neither the TreeList keyword nor the CutList keyword is included in the simulation the TreeList file will be empty and the post processor will report an error.

Several sample output tables are shown below, followed by a description of the types of information each contains. The type of table that is produced, as well as the format and diameter class size are program options specified either in the interface to the Stand and Stock Tables post processor, or on the command line used to launch the post processor.

The first sample output table was produced by selecting the first table type option, which creates standard tables for all species combined. The values are displayed by diameter class for each of the attributes. If the second table type option had been selected, this type of table would have again been created, and in addition similar tables would have been created for each individual species present in the stand.

```
FVS Run: test
Stand:   002047.0002
Mgmt Id: NONE
Year:    1997
                                   FOREST VEGETATION SIMULATOR
                                     STAND AND STOCK TABLES
                                Per-acre values are based on total stand area


Species: ALL    Year: 1997    Mgmt Id: NONE    Stand: 002047.0002
------|------------------LIVE TREES-------------------|---------------HARVESTED TREES----------------|---------------MORTALITY TREES---------------
      |                                               |                                               |
DIAM. |   TREES   AVG  BASAL   TOTAL   MERCH    MERCH |   TREES   AVG  BASAL   TOTAL   MERCH    MERCH |   TREES   AVG  BASAL   TOTAL   MERCH    MERCH
CLASS | PER ACRE    HT   AREA  CU FT   CU FT   BD FT | PER ACRE    HT   AREA  CU FT   CU FT   BD FT | PER ACRE    HT   AREA  CU FT   CU FT   BD FT
------|-----------------------------------------------|-----------------------------------------------|-----------------------------------------------
    2 |   6750.0   1.4   4.2    50.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
    4 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
    6 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |     35.1  53.0   6.7   147.5   98.3     0.0
    8 |     36.2  54.9  13.3   272.2  239.2     0.0 |     12.1  56.0   5.0    95.7   86.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   10 |     79.7  66.8  40.0  1027.2  933.5  2608.7 |     59.8  66.8  30.0   770.4  700.1  1956.5 |      0.0   0.0   0.0     0.0    0.0     0.0
   12 |     17.8  74.5  13.3   403.2  383.6  1536.3 |     13.4  74.5  10.0   302.4  287.7  1152.3 |      8.8  60.0   6.7   161.5  149.2   583.7
   14 |     25.2  80.9  26.7   790.2  750.3  3205.9 |     18.9  80.9  20.0   592.6  562.7  2404.4 |      0.0   0.0   0.0     0.0    0.0     0.0
   16 |     19.7  79.9  26.7   734.2  512.2  2341.0 |     14.8  79.9  20.0   550.6  384.1  1755.8 |      0.0   0.0   0.0     0.0    0.0     0.0
   18 |     14.4  89.1  26.7   909.1  872.1  4225.3 |     10.8  89.1  20.0   681.9  654.1  3169.1 |      0.0   0.0   0.0     0.0    0.0     0.0
   20 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   22 |      2.7  87.0   6.7   196.1  188.2   908.1 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   24 |      4.3 101.6  13.3   447.9  432.9  2271.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   26 |      1.8  94.0   6.7   202.6  195.6  1096.9 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   28 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
   30 |      1.3  91.0   6.7   195.8  188.9  1099.6 |      0.0   0.0   0.0     0.0    0.0     0.0 |      0.0   0.0   0.0     0.0    0.0     0.0
------|-----------------------------------------------|-----------------------------------------------|-----------------------------------------------
Total |   6953.1   3.4 184.2  5228.4 4696.6 19293.0 |    129.7  72.0 105.0  2993.6 2674.8 10438.1 |     43.9  54.4  13.3   309.0   247.6   583.7
```

The second table was produced using the third table type, which produces tables for trees per acre, cubic foot volumes, and board foot volume. The values are displayed by diameter class for all species combined, and for each species present in the stand.

```
FVS Run: test
Stand:   002047.0002
Mgmt Id: NONE
Year:    1997
```

FOREST VEGETATION SIMULATOR
STAND AND STOCK TABLES
Per acre values are based on total stand area

TREES PER ACRE Year: 1997 Mgmt Id: NONE Stand: 002047.0002

DIAM. CLASS	LIVE ALL	LIVE DF	LIVE WF	LIVE ES	LIVE AS	HARVESTED ALL	HARVESTED DF	HARVESTED WF	HARVESTED ES	HARVESTED AS	MORTALITY ALL	MORTALITY DF	MORTALITY WF	MORTALITY ES	MORTALITY AS
2	6750.0	1666.7	750.0	4333.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	35.1	0.0	0.0	35.1	0.0
8	36.2	16.1	0.0	20.1	0.0	12.1	12.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	79.7	26.4	13.5	39.8	0.0	59.8	19.8	10.2	29.8	0.0	0.0	0.0	0.0	0.0	0.0
12	17.8	0.0	0.0	7.7	10.1	13.4	0.0	0.0	5.8	7.6	8.8	0.0	8.8	0.0	0.0
14	25.2	13.1	0.0	14.0	0.0	18.9	9.9	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0
16	19.7	14.5	0.0	5.2	0.0	14.8	10.9	0.0	3.9	0.0	0.0	0.0	0.0	0.0	0.0
18	14.4	3.5	3.8	7.1	0.0	10.8	2.6	2.8	5.3	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	2.7	0.0	2.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	4.3	2.1	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	1.8	1.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	1.3	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TOTAL	6953.1	1745.6	254.9	4411.0	10.1	129.7	55.3	0.0	43.2	7.6	43.9	0.0	0.0	35.1	0.0

DESCRIPTION OF THE OUTPUT

FVS Run	Base name of the FVS simulation
Stand	26-character stand identification code
Mgmt Id	4-character management code from the MgmtId keyword record
Year	Year in which the FVS cycle began
Species	2-character FVS species code. The designation ALL represents all species combined.
Diam Class	DBH class (inches) for which the values are reported. The number shown is the midpoint of the diameter class. The example output was created using 2-inch diameter classes. In this case, the class labeled with a 4 includes trees with DBH values at least 3.0 inches, and less than 5.0 inches. The first diameter class is always larger than the others because it includes trees with DBH values from 0.0 inches up to the lower diameter of the second class.

LIVE TREES All values in the LIVE TREES section are representative of the beginning of the FVS cycle. This is before any harvest activity, growth, or mortality.

Trees Per Acre	Total number of live trees per acre in the diameter class at the beginning of the cycle
Avg Ht	Average height (feet) of all the live trees in the diameter class
Basal Area	Total basal area (ft2/acre) of all the live trees in the diameter class
Total Cu Ft	Total cubic foot volume (western U.S.) or total cubic foot volume of the pulpwood and sawlog portions combined (eastern U.S.) for all the live trees in the diameter class
Merch Cu Ft	Total merchantable cubic foot volume (western U.S.) or total sawlog cubic foot volume (eastern U.S.) for all the live trees in the diameter class
Merch Bd Ft	Total merchantable board foot volume (western U.S.) or total sawlog board foot volume (eastern U.S.) for all the live trees in the diameter class

HARVESTED TREES All values in the HARVESTED TREES section are representative only of the harvested material. Harvests are done in FVS at the beginning of the cycle, before growth or mortality.

Trees Per Acre	Total number of harvested trees per acre in the diameter class
Avg Ht	Average height (feet) of all the harvested trees in the diameter class
Basal Area	Total basal area (ft2/acre) of all the harvested trees in the diameter class
Total Cu Ft	Total cubic foot volume (western U.S.) or total cubic foot volume of the pulpwood and sawlog portions Combined (Eastern U.S.) For All The Harvested Trees In The Diameter Class

Merch Cu Ft	Total merchantable cubic foot volume (western U.S.) or total sawlog cubic foot volume (eastern U.S.) for all the harvested trees in the diameter class
Merch Bd Ft	Total merchantable board foot volume (western U.S.) or total sawlog board foot volume (eastern U.S.) for all the harvested trees in the diameter class

MORTALITY TREES All values in the MORTALITY TREES section are representative only of trees that have died during that FVS cycle. Mortality is applied in FVS before growth is applied to any trees.

Trees Per Acre	Total number of live trees per acre in the diameter class that died during that FVS cycle
Avg Ht	Average height (feet) of all the mortality trees in the diameter class
Basal Area	Total basal area (ft2/acre) of all the mortality trees in the diameter class
Total Cu Ft	Total cubic foot volume (western U.S.) or total cubic foot volume of the pulpwood and sawlog portions combined (eastern U.S.) for all the mortality trees in the diameter class
Merch Cu Ft	Total merchantable cubic foot volume (western U.S.) or total sawlog cubic foot volume (eastern U.S.) for all the mortality trees in the diameter class
Merch Bd Ft	Total merchantable board foot volume (western U.S.) or total sawlog board foot volume (eastern U.S.) for all the mortality trees in the diameter class

Command Line Arguments

The Stand and Stock Tables post processor may be run by typing `standtbl` at a DOS or UNIX command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to the post processor, such as input filename and program options. The following command line arguments may be used. Each must be preceded by a dash (-). If an argument is missing, the default value will be used. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

-*TreeListFilename*	The name of the TreeList file that is to be used as input. This file usually has a `.trl` extension. The actual filename is used in place of the designation *TreeListFilename* shown here. This must be the first command line argument.
-*OutputFilename*	The name of the file that will be used to write the output from the post processor. The actual filename is used in place of the designation *OutputFilename* shown here. If used, this must be the second command line argument. If there is more than one command line argument, the second will always be read as the output file name. If this argument is not used, the output file will be given the same base name as the TreeList file with a `.stb` extension.
-C	Comma delimited output. A single header record is printed. All other records contain a series of values separated by commas. This format is primarily used for exporting the data. The default is to print the information in standard tabular format.
-S#	Size of diameter classes, where # is a number representing the size, in inches, of the diameter classes that are to be used in the tables. The size must be an integer value, so no decimal point is allowed.
-T#	Table type, where the # is a number representing the table type. All of the possible table type designations are shown below. The default type is standard tables.

T1 = standard tables for individual species and for all species combined. For every cycle for which the TreeList file contains information, a table is produced that contains all of the attributes by size class for each individual species present, and for all species combined.

T2 = standard tables only for all species combined. For every cycle for which the TreeList file contains information, a table is produced that contains all of the attributes by size class for all species combined.

T3 = volume tables. For every cycle for which the TreeList file contains information, tables are produced for each volume type (board feet per acre, for example) with the values listed by size class for the individual species present.

-X Exit immediately after running the post processor. The interface does not appear. This is primarily only useful in batch processing where it is desirable to have the post processor output produced without having the interface open.

If the -C, -S#, -T# or -X command line arguments are to be used, the FVS output filename argument and output filename argument must also be used. In other words, the -C, -S#, -T# or -X argument may never appear as the first or second command line argument.

Command Line Example:

```
standtbl -example.trl -example.stb -S1 -T2
```

This will run the Stand and Stock Tables post processor using the file `example.trl` as input, using the file `example.stb` for the output, using one inch diameter classes in the tables, and including only the tables with all species combined.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching the post processors. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Stand and Stock Tables post processor is shown below.

```
//start ppif.standtab
name:{Stand Tables and Stock Tables}
command{dos}:{
!fvsbin!\\standtbl.exe -!run!.trl -!run!.stb}
...
//end ppif.standtab
```

The command line that calls the post processor on a Windows system is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\standtbl.exe` will become `C:\Fvsbin\standtbl.exe`. This is the command that actually calls the post processor. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. The first command line argument would then be `-testrun.trl`, which represents the name of the TreeList file used as input for the post processor. This is the default name that FVS will give the TreeList file. The second command line argument would be `-testrun.stb`. It is highly recommended that the `!run!` designation in the filenames not be changed.

If other command line arguments are desired, they should be added after `-!run!.stb` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to create comma delimited output using 5-inch diameter classes, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\standtbl.exe -!run!.trl  -!run!.stb -C -S5
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

Stand Visualization System

Variants: All
Keywords required: **SVS**
Program filename: WinSvs.exe

The Stand Visualization System (SVS) generates images depicting stand conditions represented by a list of individual stand components, e.g., trees and down material. It is a robust program that can use data from a variety of sources, one of which is FVS. It is called a post processor for our purposes because it can be called as a post processor from the *Suppose* interface. The required input files are produced by FVS when the SVS keyword record is included in the simulation. Without the SVS keyword record the required input files are not created.

*An **SVS** keyword must be included in the FVS simulation file.*

When called as a post processor in *Suppose*, there are two options for running SVS. The first option is called "SVS for Windows," and it launches the SVS program with a series of files created by the SVS keyword in FVS. The user has full access to all of the options available in SVS. The second option is called "SVS for Windows - Movies," and it runs the SVS program in the background creating a series of image files. A viewer is then launched to view the images in a timed sequence. The user does not have access to any of the SVS options.

A detailed discussion of the Stand Visualization System is beyond the scope of this document. There is, however, a document dedicated entirely to this model. Please refer to the document Stand Visualization System (SVS) by R.J. McGaughey. That document is available from the Documents section of the Forest Vegetation Simulator web site (which is at www.fs.fed.us/fmsc/fvs at the time of production of this document). SVS also comes packaged with comprehensive online help files.

Program options may be specified through command line arguments. Please refer to the SVS documentation mentioned above for a description of those options. A few selected command line arguments are discussed here.

The *Suppose* interface program uses command line arguments to call the post processors. You can change the default *Suppose* command line arguments that are used to call SVS.

Command Line Arguments

The Stand Visualization System may be run by typing `winsvs` at a DOS command prompt. The program will then open and the user will have to enter all of the desired information. Additional things called command line arguments may be added to the command typed at the command prompt. These arguments convey additional information to SVS, such as input file or viewing angle. The *Suppose* interface program uses specialized *Suppose* command line arguments to launch this post processor.

The list of command line arguments below is only a partial list of those available. Please refer to the documentation referred to above for a description of all of the available command line arguments.

SvsFilename	The name of the SVS file that is to be used as input. Note that this argument is not preceded by a dash (-). If used, this must be the LAST command line argument. The actual filename is used in place of the designation *SvsFilename* shown here. The file name usually has a `.svs` extension. When used with FVS the file is usually an index file, which will have the same base name as the simulation file with `_index.svs` attached. An index file contains a list of the individual SVS treelist files that FVS produces for each FVS cycle for each stand in the simulation.
-A#	Azimuth from the center of the SVS plot to the viewpoint location. The # is a number representing the angle in degrees.

-C*filename*	Captures the image to a file, and then exit SVS. The actual filename is used in place of the designation *filename* shown here. The image is rendered in the background, so the user never sees it. If the file is an index file, each individual SVS treelist file specified in the index file will be loaded into SVS and then captured to a file. The files will be given the same base name as the index file with an extension of i##, where ## is the image number in the sequence of images.
-D#	Distance from the viewpoint tothe center of the stand. The # is a number representing the distance in feet, and must be greater than zero.
-E#	Elevation of the viewpoint above the ground level. The # is a number representing the elevation in feet, and must be greater than zero.
-F	Forces the "Movies" option to use the perspective view. Without this argument, the movies will be created using the view that was displayed the last time SVS was run.
-I	Runs the image viewer program. This is the used to display the "movies" when that option is selected in *Suppose*. The images are usually captured using the -C argument.
-L#	Lens focal length used to generate the perspective view. The # is a number representing the lens focal length in millimeters. A larger value (up to 400) zooms in for a larger image, while a smaller value (down to 20) zooms out for a smaller image.
-S#	Stand focus point elevation for the perspective view. The # is a number representing the height at the near edge of the stand that should be the center of the image. This is similar to tilting a camera up or down to change where the center of the image is focused.

Suppose Command Line Arguments

The *Suppose* interface program uses command line arguments to specify the program options when launching SVS. The command lines are contained in a special file called suppose.prm. The format of the command line may look strange due to the way the *Suppose* program handles filenames, but the general syntax is identical.

A portion of the section of the suppose.prm file that deals with the Stand Visualization System is shown below.

```
//start ppif.22asvs
name:{SVS for Windows (use with new SVS keyword)}
command{dos}:{
!fvsbin!\\winsvs.exe -a315 -e135 -s75 -d445 -l50 !run!_index.svs
}
...
//end ppif.22asvs
```

A portion of the section of the suppose.prm file that deals with the Movies option for the Stand Visualization System is shown below.

```
//start ppif.22asvsmovie
name:{SVS for Windows - "Movies" (use with new SVS keyword)}
command{dos}:{
if exist !run!\\* !fvsbin!\\winsvs.exe -a315 -e135 -s75 -d445 -l50 -i -f -c!run!\\!run! !run!_index
}
...
//end ppif.22asvs
```

The command line that calls the Stand Visualization System is between the curly brackets following `command{dos}:`. The directory in which the FVS software resides is identified as `!fvsbin!\\` by *Suppose*. For example, this directory might be C:\Fvsbin\, in which case `!fvsbin!\\winsvs.exe` will become `C:\Fvsbin\winsvs.exe`. This is the command that actually calls SVS. The remainder of the line represents the command line arguments. Please refer to the section on command line arguments for a description of each of them.

Suppose represents the name of the simulation with `!run!`. For example, if your simulation file is named `testrun.key`, then `!run!` will be `testrun`. `!run!_index` would then be `-testrun_index`, which represents the base name of the SVS index file.

If other command line arguments are desired, they should be added before `-!run!.stb` but before the `}`. A space must precede any additional argument, and all arguments must begin with a dash (-). For example, to run the SVS program in the standard mode viewing the stand from an azimuth of 90 degrees, the command would be written as shown below. The curly bracket would follow immediately.

```
!fvsbin!\\winsvs.exe –a90 -e135 -s75 -d445 -l50 !run!_index.svs
```

CAUTION - Great care must be exercised whenever modifying the suppose.prm file. The *Suppose* program uses this file for nearly everything it does. Even the slightest error in the suppose.prm file can cause major malfunctions throughout *Suppose*.

TOSS

Variants: All
Keywords required: None
Program filename: Toss.exe

The TOSS (Table Output Selection Screen) post processor allows the user to select which tables from the main FVS output file are to be viewed. Only those tables that are selected will be included in the output. The output file is always named toss.log, but the user can save the output to a different file if desired.

For a complete discussion of the TOSS post processor please refer to the document Select Topics for the Forest Vegetation Simulator by D.A. Vandendriesche. That document is available from the Documents section of the Forest Vegetation Simulator web site (which is at www.fs.fed.us/fmsc/fvs at the time of production of this document).

Suppose Parameters File

The *Suppose* program was written to rely heavily on a parameters file, which is usually named `suppose.prm`. This file is typically found in the same directory with the *Suppose* executable file, which is usually named `Fvsbin`. Many of the things done by the *Suppose* program are controlled through this file. For example, the entire set of keywords, along with the instructions needed to draw the keyword window and the commands needed to write the keyword record to the simulation file, are contained in the parameters file. This system allows updates and modifications to the *Suppose* program without having to recompile the computer code. It also allows the user the ability to change things without needing a version of the program compiled especially for them.

The danger in allowing this flexibility is that errors are quite easy to introduce into the system. There is no way for *Suppose* to check the parameters file. Even a very small error in the parameters file can cause major malfunctions throughout the *Suppose* program. It is therefore extremely important that this file remain in perfect working order. There is a very specific syntax that must be followed in the parameters file. For a comprehensive explanation of this syntax refer to the online help files available through the *Suppose* Help menu.

Parameters for the Post Processors

Major sections begin with the designation `//start sectionName` and end with `//end sectionName`, where `sectionName` is replaced with the actual name of the major section. The first major section is the preferences section.

Immediately following the preferences section are the sections dealing with the post processors. The first post processor section in the default `suppose.prm` file is for SVS for Windows. That section begins with the designation `//start ppif.22asvs` and ends with the designation `//end ppif.22asvs`. The entire section is shown below.

```
//start ppif.22asvs

name:{SVS for Windows (use with new SVS keyword)}

command{dos}:{
!fvsbin!\\winsvs.exe -a315 -e135 -s75 -d445 -l50 !run!_index.svs
}

description:{\
The Stand Visualization System (SVS) will provide a 3D drawing of
a stand.  Don't run this and "SVS Movies" in the same run.

You must use the SVS keyword to generate the SVS tree list files
directly from FVS.

SVS for Windows was developed by Robert J. McGaughey of the Pacific
Northwest Research Station, Seattle, WA}

//end   ppif.22asvs
```

The following sections appear in the major section for every post processor.

//start *SectionName*	Signal for the start of the major section. The designation *SectionName* is replaced with the actual code for the post processor. *Suppose* lists the post processors alphabetically according to these codes.

name:{*PostProcName*}	Name of the post processor as it will be displayed in the list of post processors in *Suppose*. The designation *PostProcName* is replaced with the actual name of the post processor.
command{dos}:{*PostProcCommand*}	Command that will be used to call the post processor when running *Suppose* on a PC. The designation *PostProcCommand* is replaced with an actual command line call. The command line is used to call the post processor when the simulation is run. *Suppose* uses the designation `!fvsbin!\\` to represent the directory in which the FVS programs are stored. If this directory is `C:\Fvsbin\` then `!fvsbin!\\winsvs.exe` would become `C:\Fvsbin\winsvs.exe` when translated into an actual command line. Similarly, Suppose used the designation `!run!` to represent the base name of the simulation file. If the simulation file is `testfile.key` then `!run!_index.svs` would become `testfile_index.svs` when translated into an actual command line.
description:{*PostProcDescription*}	Description that is to be printed at the bottom of the *Suppose* post processor window when the post processor is highlighted in the list of available post processors. The designation *PostProcDescription* is replaced with the actual description, which may be multiple lines. It is simply to aid the *Suppose* user in the selection of a post processor. It typically describes the output produced by the post processor, and the keywords required for its use. The entire description is always contained between a pair of curly brackets.
//end *SectionName*	Signal for the end of the major section. The designation *SectionName* is replaced with the name of the post processor section, and must be exactly the same as the *SectionName* used at the start of the major section.

Editing the Parameters File

CAUTION - Since *Suppose* relies so heavily on the parameters file for everything that it does, it is extremely important that this file be maintianed in perfect working order at all times. Even a small error, like the omission of a single curly bracket, can cause malfunctions throughout the *Suppose* program, and at times cause it to crash. *Extreme care must be exercised whenever editing this file to be sure that it conforms exactly to the required syntax.* It is highly recommended that a backup copy of the original file be saved in case errors are introduced during editing.

The parameters file that is packaged with the *Suppose* program is named suppose.prm. It is recommended that this name be maintained, even if the file is edited. When the *Suppose* program is launched, it looks for a file named suppose.prm (unless a special *Suppose* command line argument is included to indicate a different parameters file should be used).

The parameters file is simply an ASCII text file, and may therefore be edited in any text editor. When saving the file it must be saved as an ASCII text file. With regard to the post processors, the most typical change would involve adding, deleting, or changing command line arguments in a command line section. These are the sections that begin with `command{dos}:` or `command{unix}:` and the actual command line is contained between the associated curly brackets. Edits need only be made to the command line for the type of computer system being used.

The command line arguments change the initial options used by the post processor when it is run. Usually, any additional command line arguments that are added to a command line are placed at the end of the command line, just before the closing curly bracket. The command line arguments for each individual post processor are described in the command line arguments section of the documentation for the post processor.

The U.S. Department of Agriculture (USDA) prohibits discrimination in all its programs and activities on the basis of race, color, national origin, sex, religion, age, disability, political beliefs, sexual orientation, or marital or family status. (Not all prohibited bases apply to all programs.) Persons with disabilities who require alternative means for communication of program information (Braille, large print, audiotape, etc.) should contact USDA's TARGET Center at (202) 720-2600 (voice and TDD).

To file a complaint of discrimination, write USDA, Director, Office of Civil Rights, Room 326-W, Whitten Building, 1400 Independence Avenue, SW, Washington, DC 20250-9410 or call (202) 720-5964 (voice or TDD). USDA is an equal opportunity provider and employer.