Resource Standard Metrics for C, C++ and Java

Version 6.01

Resource Standard Metrics provides a standardized approach to measure the quantity and quality of source code within a module or a project. This program can analyze a C, C++ or Java source for metrics and semantic quality problems that most compilers do not identify.

~ Source Code Metrics and Analysis ~

RSM measures source code for quantity and quality metrics. Size metrics are described by counting "lines of code". Source code quality analysis is measured by semantic analysis of the code beyond the syntax rules of the language.

(LOC) A line of code is defined as a line within a source file that is not a comment or blank line. Lines that contain both source code and comments are counted as an instance of each.

(eLOC) Resource Standard Metrics defines an effective line of code as a LOC which is not a stand-alone braces {} or parenthesis (). This metric more accurately defines the quantity of work performed in a source code module and is an invention of M Squared Technologies. We have found that eLOC is the metric we intuitively estimate as experienced software engineers.

(lLOC) Logical Lines of code are defined as code statements or those lines which end in a semicolon. The "for" loop structure accounts for one lLOC.

Source code line	LOC	eLOC	lLOC	Comment	Blank
if (x<10) // test range	x	x		x	
{	x				
// update y coordinate				x	
					x
y = x + 1;	x	x	x		
}	x				

Logical lines within a source file can exceed the physical lines within a file code and comments occur on the same physical line. The sum of code, blanks and comments equates to the logical lines. Metrics programs that show code, comments and blank lines equal to the physical lines are not accounting for a second instance of code or comments.

Various key words and statements are provided for code analysis. Readability and code quality can easily be determined by analyzing the quality notices, comment percentage, white space content and key words used within the source code.

~ Program Operation ~

Source files may be processed via wild cards, recursively down a file tree or from a specified list of files or directories. The evaluation copy of this program will only process 10 files at one execution. Files are normally processed in the order they are discovered by the input mode. The option -Rn will sort the input list alphabetically for sorted processing. The -F option processes a specific list of files or directories.

RSM operates through a series of switches and options. RSM processes files via several input modes and creates reports via several output modes. This user manual details the input modes, output formats, modes and runtime switches.

~ Operation Step by Step ~

RSM is designed to operate according to the following process. Each step of the process is covered in detail in this manual.

Step 1: Determine the output format, -H -A or no switch for text.

"no option"	ASCII UNIX Text Format
-A	CSV format for spreadsheet import
-H	HTML format with hyperlinks

Step 2: Determine the output file relative to the format.

-O report.htm -H	WWW Browser Report
-O report.csv -A	Spreadsheet Report
-O report.txt	Text Report

An option to using the -O option is the use of the shell redirection to a file. This is useful if a DOS text file is desired versus the RSM -O UNIX text format.

```
rsm -s *.cpp > report.txt
```

Step 3: Specify the report type using the various report switches individually or by aggregation.

-f -c -o	Functions, Complexity, Objects
-Es -Ec	Extract strings and comments
-hs	Help Syntax

Step 4: Specify the file input mode using file names, wild cards, file lists or recursive descent.

.\inc\anyfile.h	Ordinary file names.
*.h *.cpp	Wild cards, command line expansion
-F somefilelist.txt	File list of names and directories
-r h,c,cpp,java .	Recursive descent where . specifies
-r h,cpp,java /project	the current working directory.

Step 5: Creating the RSM command line.

```
rsm anyfile.h anyfile.cpp
rsm -O analysis.txt -n c:\proj\*.h c:\proj\*.cpp
rsm -O metrics.txt -Ta -Tl -TN -F project1_list.txt
rsm -H -O metrics.htm -f -c -o -r h,cpp,java .
rsm -A -O metricsdiff.csv -wx baseline1.dat baseline2.dat
```

~ Program License File ~

RSM uses an encrypted license file to provide each user with a unique license tailored to each user's runtime requirements. RSM will look for the license file rsm.lic at the location specified by the environment variable RSMHOME.

If the license file is not found at the RSMHOME location, RSM will search the system PATH for the rsm.lic file. If the license file cannot be located, RSM will warn the users and default to the SHAREWARE mode. If the rsm.lic file detects that the file contents have been modified, RSM will default to the SHAREWARE mode. Hacking the RSM license file will void the end user license agreement.

~ Program Configuration ~

RSM is configured from a configuration file, rsm.cfg. If the configuration file is not found, RSM creates a configuration file. You can edit the configuration file with an ASCII editor to change the operation parameters of RSM. The configuration file is heavily commented to assist in this customization. You may use the -hc option to examine the rsm.cfg file.

RSM looks for the rsm.cfg file at the location where the license file was found. The rsm.cfg file, like the license file should be placed in a directory is located on your system path. The file name has to be rsm.cfg.

~ Program Syntax ~

```
Syntax: rsm <-option -option parameter> filename or wild cards
        -A Automated CSV output mode
           -A -f   Functions and Files
           -A -i   Inheritance Tree
           -A -s   File Summary
           -A -Ta  All T reports
           -A -Tf  Total Functions
           -A -To  Total Structs, Classes, Interfaces, Namespaces
           -A -Tq  Total Quality Notices
           -A -Ti  Total Inheritance Tree
           -A -wW  Work file differentials
           -A -wx  workfile.old workfile.new  Work file extraction
        -a Allocation/de-allocation memory function counts
        -b Benchmark processing performance
        -C Comment documentation relative to function or class
        -c Complexity metrics, Cyclomatic and Interface
        -D De-character source, Warning: Modifies Source Code
           -Dt  Remove tabs \t
           -Du  Remove carriage returns \r, make UNIX files
           -Dd  Add carriage returns \r, create DOS files
        -d Determine source code against LOC analysis algorithm
        -E Extract literal strings or comments from source code
           -Es  Extract strings
           -Ec  Extract comments
        -e Estimate labor rate analysis based on LOC and complexity
        -f Function Identification and LOC metrics
```

```
-F "FileList.txt"  Input mode from a file list
-H HTML output mode for use with a WWW browser
-h Help and licensing information
   -hc  Configuration Settings
   -hs  Syntax only
   -hf  Full Help Text Document
   -hl  License Information
   -hn  Quality Notice Types
   -hp  Abbreviated Syntax with Pause Prompt
   -hu  User License Agreement
-i Inheritance tree generation
-k Key for sorting reports
   -k0  No report sorting (default)
   -k1  Sort by Name
   -k2  Sort by LOC
   -k3  Sort by eLOC
   -k4  Sort by lLOC
   -k5  Sort by Cyclomatic Complexity
   -k6  Sort by Work File Difference State
-l List function names declared in each file
-m Macro identification
-n Notices for code quality analysis & common errors
-N Notice output format
   -Ns  Single line summary format
   -Nv  Visual Studio and KAWA interactive format
-o Object metrics classes and structs
-O <filename> output mode to a file instead of screen
-p Printable format for source code
   -B 1          Begin Top Margin (defaults shown)
   -M 5          Left Margin
   -P 55         Page Length
   -L 80         Line Length
   -S "string" Header String
-r "ext,ext,..." "directory"  Recurse directory input mode
-R Read File Input Processing Options
   -Rd  Process directory (-r No Recursive Descent)
   -Rl  List files to be processed and exit
   -Rn  Files input are sorted by name
-s Summary LOC metrics per file
-T Totals only mode, Sorted
   -Ta  All Total reports
   -Tf  Functions, Classes Namespaces and Notices
   -Ti  Inheritance Tree Only
   -To  Objects Structs, Classes, Interfaces, Namespaces
   -Tp  Quality Profile by notice type
   -Tq  Quality Notice summary list by file
   -Tn  Quality Notices emitted by file and line number
   -Ts  Total summary LOC metrics
   -Tl  Summaries only, no listing of functions, classes etc.
   -TN  No file names displayed
   -Tw  Work file metrics differentials only
-v Verbose metrics for files, LOC, keywords and constructs
-u User Log Management
   -us  Show the current user log
   -ua  (Network Lic) Archive log/restart
   -ur  (Network Lic) Remove user from current log
-w Work files for metrics differentials between baselines
```

```
        -wc  Create files, workfile.dat and rm112220.00 date format
        -w  "File myworkfile.dat"  User specified work file
        -wW  Work differential between last (workfile.dat) and now
        -wx  Extract work differential between two work files
     -y Display configuration parameters, export rsm.cfg
```

~ Source File Requirements ~

RSM is designed to process only source code from the C, ANSI C, C++ ANSI C++ and Java 2.0 languages. RSM requires that each source file be compilable. RSM reads no preprocessor directives so that it may analyze all the source within a file. RSM will always process header files prior to source files unless overridden by wild card ordering.

There may be macros used within the source code of which RSM may not be aware. Windows defines macros and keywords that are outside the language specification. These macros must be defined to RSM in the rsm_macro.cfg file. This configuration file is located with the rsm.cfg configuration file.

~ Switches and Options ~

RSM uses runtime switches to configure its operation. There are two types of switches. The plain switch takes no parameter and the complex switch requires a modification parameter. Switches like f, o, v, c are plain switches that may be specified by using a dash for each switch or by running them as a single string.

```
i.e.  Plain Switches:  rsm -focmnv *.cpp
                       or
                       rsm -f -o -c -m -n -v *.cpp
```

Complex switches must be separated by a dash(-) for each switch. Each complex switch requires an additional modifier. If -r is used for recursive mode, it must be the last switch.

```
i.e.  Complex Switches: rsm -focmnv -k4 -r h,cpp c:\src
                         where k and r are complex switches
```

~ Files/Directories Names and Wild Cards ~

The Windows operating system requires DOS type paths and filenames. The UNIX OS requires UNIX paths and filenames. Under Windows, filenames are case insensitive where in UNIX they are case sensitive. The rsm.cfg configuration file has a setting that must reflect the case sensitive aspect of the (OS) operating system or incomplete identification of files may result.

Windows files or directories with spaces within their names must have these names delimited with quotation marks, "c:\Program Files".

```
i.e.  normal with wild cards: rsm -f -c *.c *.cc
      recursive descent:      rsm -f -o -r c,cc,h,hh /proj
      read a file list:       rsm -f -o -c -F filelist.txt
(Windows)                     rsm -f -c c:\src\*.h c:\src\*.c
```

(UNIX) rsm.sol -f -c /src/*.h /src/*.c

~ Source Code File Input Modes ~

RSM processes files in one of four modes. Modes cannot be mixed.

1. Direct Files
   rsm -f -o /proj/src/filename.cpp

2. Wild Cards
   rsm -f -o /proj/inc/*.h /proj/src/*.cpp
   Note: The order of header files must come before cpp files.

3. Recursive descent of a directory tree
   rsm -f -o -r h,cpp /proj/src

4. Reading from a list of files or directories
   rsm -f -o -F /proj/filelist.lst

~ Program Output Modes ~

RSM can output its reports in three modes. Modes cannot be mixed.

1. Direct output to the screen
   rsm /proj/src/filename.cpp

2. Direct output to a file
   rsm -O report.txt /proj/src/*.cpp

3. Redirection to a file
   rsm -r h,cpp /proj/src > report.txt

~ Program Output File Formats ~

RSM can create reports in three formats, text, HTML and CSV.

1. Text or default mode
   rsm -O report.txt *.h *.cpp

2. HTML Format
   rsm -O report.htm -H *.h *.cpp

3. Comma Separated Variables (spreadsheet import)
   rsm -O report.csv -A -f *.h *.cpp

~ Run Time Switch Options ~

The options shown with an asterisk '*' can be aggregated together to form compound reports. Some experimentation will be required to find the combination of options that yield the desired result. For a complete cross correlation of all switches and options, please reference the on-line manual.

Automated CSV File Output Mode

-A<mode>

The CSV mode creates a Comma Separated Variables output for import of metrics directly into spreadsheet programs. This output can be redirected to a file then directly opened by MS Excel.

```
-A -f   Functions and Files
-A -i   Inheritance Tree
-A -s   File Summary
-A -Ta  Sorted all T reports
-A -Tf  Sorted Total Functions
-A -Tq  Quality Notice by File
-A -To  Sorted Total Structs, Classes, Interfaces, Namespaces
-A -Ti  Class inheritance tree
-A -wW  Work file differentials
-A -wx  Work file extraction
```

```
i.e.  rsm -A -f *.cpp *.c > metrics.csv
      rsm -A -Ta -k5 *.h *.cpp > metrics.csv
      rsm -A -w *.h *.cpp > diff.csv
      rsm -A -x oldfile.dat newfile.dat > diff.csv
      rsm -A -Ti -O proj_inherit.csv -F projfiles.txt
```

* Allocation/De-allocation of Memory Mode

-a

Produces metrics on memory creation through the counting of instances of malloc, calloc, realloc and new. Memory de-allocation is measured by counting free and delete. This metric is useful for focusing on functions which use dynamic memory. Most program bugs can be attributed to problems in dynamic memory management.

```
i.e.  rsm -a *.c
```

* Benchmark Mode

-b

This mode tracks the CPU, User, Wait and elapsed time for the running of RSM. This mode is intended to create a timing metric for RSM so that the time required to execute metrics is known. It may also serve as a benchmark for comparing RSM to other like products, although no other products are known which have as many features as RSM.

```
i.e.  rsm -b *.c
```

* Comment Documentation Mode

-C

This mode extracts comments relative to functions and classes for the

specified files by either wild cards or file specification options. This mode should not be used with other metrics options.

i.e. rsm -C *.c

* Complexity Mode

-c

This mode measures both cyclomatic complexity and function interface complexity. Function cyclomatic and interface complexity are combined to form function complexity.

Cyclomatic complexity is consistent with McCabe's definition where each decision point or predicate node is counted plus 1 for the function body. Cyclomatic complexity describes the number of logical pathways through a function and is typically used to asses whether a function can be de-composed into several functions. V(g) is the symbol used to describe this metric. This metric can also indicate the number of test cases required in unit test to fully test a method.

Functional interface complexity is calculated by summing the number of function input parameters and the number of return states. This metric can be useful when analyzing the complexity of a function's interface. A function's estimated size can be related to a function's interface complexity using the -e option.

Function complexity is calculated by the sum of cyclomatic complexity and the interface complexity. The maximum and average complexity is reported for the file and for all files processed or the entire project.

i.e. rsm -c *.java

De-Character Source Files

-D<mode>

De-character source code to remove tabs and convert source code from DOS to UNIX and UNIX to DOS format. This option operates in a standalone mode and modifies the original source. You must have the source code "checked out" and have write permissions to use this options.

-Dt Remove tabs where the tabs/space setting is in the rsm.cfg file.
-Dd Create DOS formatted files where each line terminates in a carriage return and new line.
-Du Create UNIX formatted files where each line terminates in a new line.

i.e. rsm -Dt -Du *.h *.cpp
rsm -Dt -Dd -r h,c,cpp,java c:\proj\game

Deterministic Mode

-d

This mode will show the user how a line of code is interpreted by RSM Many metrics tools yield metrics that create metrics of LOC, Comments and blanks lines. When these metrics are added, the number equals the number of physical lines. Therefore, if comments are on the same line as the code, what metrics has the tool not accounted for.

RSM counts all LOC, comments and blanks and sums these metrics into a logical line count that usually is greater then the physical line count. RSM measures effective lines of code by accounting for the lines of code that are single "{", "}", "};", "(" or ")" characters.

i.e. rsm -d source.c

This mode produces the validation of the given source against the LOC algorithm of Resource Standard Metrics. Output will echo each line of code and a state of the LOC determination.

Example Output:

Line 1: /* functions using pointers to functions */
LOC Type(s): <C Comment>

Line 2:
LOC Type(s): <Blank>

Line 3: void calculate(int x) /*standard includes*/
LOC Type(s): <Code> <C Comment>

Line 4: {
LOC Type(s): <Non-Effective LOC>

Extract Strings or Comments from Source Code

-E<mode>

Extract strings or comments from a source file. This feature allows for the creation of a file which contains text. The file can be processed by spell checker such as MSWord or GNU ispell. Each string and/or comment is output with the line number where it is located within the source file. This option should be used without any other RSM formatting options.

-Ec Extract comments
-Es Extract literal strings

i.e. rsm -O spellcheck.txt -Es -Ec *.h *.cpp
rsm -Es -r h,c,cpp,java c:\proj\game > spellcheck.txt

* Estimation Analysis Based on LOC and Complexity

-e

Estimation analysis presents LOC, eLOC, lLOC per: function input parameter; function return state; interface complexity (parameters + returns); cyclomatic complexity and total functional complexity (Interface + Cyclo) complexity.

i.e. rsm -e *.c

* File List Mode

-F <FileList.txt>

This mode will process files and directories from a text list file. The list must be a text file compliant to the specified format. The -RL mode can be helpful in creating a file list.

File Format:

Files must be the complete path and file name.

i.e. c:\project\src\vertex.java
/proj/source/geom.cpp

Directory Format:

Directories must the full path. The following parameters to the directory are required for processing the directory.

-r	Recursive descent of the directory
-n	No Recursive descent
c,h,cpp	Types string tells RSM which kind of files
directory	Starting point for processing

i.e. -r java /proj/java/source
-n h,cpp /proj/cpp/source

Example File
filelist.txt

```
c:\src\control.c
c:\src\display.c
-r h,c,cpp,java d:\myprojects
-n java d:\javasrc\src
```

Note that no wild cards are valid in this mode. Blank lines are ignored. Lines leading with a # character are not processed and considered comments.

* Function Analysis Mode

-f

Functional analysis mode determines the functions in a source file and if quality notices are turned on, it will perform quality checking upon the functions. Modes -f and -n are often used together.

i.e. rsm -f -n *.c

```
* HTML Output Mode
-------------------------------------------------------------------------
-H

This mode creates HTML output with color, fonts and hyperlinks to the report
files.  Such output is meant to be processed by a WWW browser like Netscape
Navigator or MS Explorer.  You may set the background color of the HTML
reports by setting the associated parameter in the rsm.cfg file.  If the
background is not set, then the browser default background color will be
used.  The HTML option is designed for use with all RSM reports and options
designated with an asterisk '*'.

    i.e.  rsm -H -O metrics.htm -fcn *.c
          rsm -Hx rm122819.97 rm122919.97 > metrics.html

The HTML text, vlink and link color must be specified together to
override the standard browser default.  You may redefine the specified
HTML colors by setting the configuration item associated with each
color.  HTML colors may be specified by the #000000 hex value or by
the text word.

HTML hyperlinks are by default the literal file name.  If you
desire relative hyperlinks you can set the associated rsm.cfg file
setting, Relative HTML Links: Yes.

Help Mode
-------------------------------------------------------------------------
-h<mode>

Produces the help output.  Such output can be printed under UNIX
by piping the output to a printer (lpr) or redirecting it to a file.

     -hc  Configuration Settings
     -hs  Syntax only
     -hf  Full Help Text Document
     -hl  License Information
     -hn  Quality Notice Types
     -hp  Small Syntax and Pause Prompt
     -hu  EULA, End User License Agreement

     UNIX:
     i.e.  rsm -hs | lpr
           rsm -hf > help.txt
           rsm -O help.txt -hf

     DOS/W95/NT/2000:
     i.e.  rsm -hs > prn
           rsm -hf > help.txt


* Inheritance Tree Generation
-------------------------------------------------------------------------
-i

This mode will generate an inheritance tree of all classes in the
```

current file set. It is assumed that the user creates a file set that has some meaning to a design. When using wild cards, one should process header files before the implementation files.

```
i.e.  rsm -i *.h *.cpp
```

* Key Project Sort Method

-k<mode>

This mode will sort the project summary data as specified by the key numeric identifier. This mode requires a modifier as specified below.

```
-k0   No sorting methods (default & fastest operation)
-k1   Sort by namespace/package, class or function name
-k2   Sort by LOC
-k3   Sort by eLOC
-k4   Sort by lLOC
-k5   Sort by Cyclomatic Complexity
-k6   Sort by Work File Difference State

i.e.  rsm -f -o -k3 *.h *.cpp
```

* List Function Name Mode

-l

This mode produces a list of functions that can be cut and pasted from the output into the comment header of the file which was processed. It will create a report where all function names are correlated with their parent file. This is useful when documenting the system.

```
i.e.  rsm -l *.cc *.c
```

Example Output:

```
  Function: Show
  Function: Drawable::Drawable
  Function: Drawable::Show
```

* Macro Mode

-m

Macro mode is similar to function mode except it reports the names of the macros within the source. Macros are a common cause of maintenance and portability headaches, so locating their existence is quite valuable.

In Windows code there may be an occurrence of placing a macro between the function name and the function open parenthesis or between the class name and the opening brace. You must tell RSM about the macros by either processing the macro prior to its use or you may added the macro by name to a special file called rsm_macro.cfg. Names in this file are read at startup so that RSM knows about your macros defined in other

portions of the code that may not be processed by RSM. The rsm_macro.cfg file is located with your rsm.lic license file.

```
i.e.  rsm -O metrics.txt -f -m *.c
```

* Quality Notice Format Mode

-N<mode>

Quality notices by default are displayed in a verbose format. This option displays a single line summary of the quality notice.

```
-Ns  Single line summary format
-Nv  Visual Studio and KAWA interactive format
```

Quality notices output in Visual Studio format allows the user, inside Visual Studio environment, to double click on a quality notice and jump directly to the file and specified line number. For specific information on setting up RSM as an integrated tool to Visual Studio visit the RSM web site, http://mSquaredTechnologies.com

```
i.e.  rsm -f -o -c -Nv *.h *.cpp
i.e.  rsm -foc -Ns *.h *.cpp
```

* Quality Notice Mode

-n

Turns on quality checking on the source for approximately 50 common problems which create difficult maintenance, porting or semantic bugs the compiler simply misses. This mode will return the number of notices to the operating system upon RSM normal exit. This will enable scripting languages to pick up the non-zero exit state and flag the end user.

```
i.e.  rsm -O metrics.txt -f -n *.cc *.c *.cpp
```

* Output Report to a file

-O <filename>

This mode routes the RSM report from the stdout or screen to a file. If the file cannot be opened, RSM will output the report to the screen.

```
i.e.  rsm -H -o -f -O rsm_report.htm *.cc *.cpp
      rsm -A -Ta -O rsm_report.csv -F project_files.txt
      rsm -O proj.txt -r h,c /proj
```

* Object Class/Struct Metrics

-o

This mode details metrics for each class specification identified

within the source code. Metrics include LOC, lines of code, eLOC, comments, blanks, the number of public, protected, private attributes and methods. If a class implements inheritance, it is shown with the respective base classes.

```
i.e.  rsm -of *.cc *.cpp
```

Print Format, Code Listing Mode

-p

This mode generates a code listing for the files specified by name, wild card, file list or recursive mode. Files are created with a user specified header, file name, page number, file date and line numbers for each line of code. Line numbers correspond with the line numbers generated from other RSM modes. This output is suitable for publication and peer reviews.

```
i.e.  rsm -p *.cc *.c *.cpp
      rsm -p -F filelist.txt
      rsm -p -r cpp,h,hh ./
```

Page format switches that apply to print code-listing mode.

```
-B 1          Top Margin,  Default 0
-M 10         Left Margin, Default 5
-P 66         Page Length, Default 55
-L 72         Line Length, Default 80
-S "string"   Header String

i.e..  rsm -O lst.txt -p -B5 -M5 -P60 -L78 -S"My Code" .\src\*.c
```

* Read File Processing Options

-R<mode>

RSM processes files from 1 of four modes.

1. Direct Files
   rsm -f -o /proj/src/file
2. Wild Cards
   rsm -f -o /proj/src/*.cpp
3. Recursive descent of a directory tree
   rsm -f -o -r h,cpp /proj/src
4. Reading from a list of files or directories
   rsm -f -o -F /proj/filelist.lst

The following options modify the way RSM processes the files to be analyzed.

-Rl Generate a list of all files which will be processed in RSM operation. This list may be cut from the report to provide a project file inventory or used at a stating point for creating a file list to process. This option exits upon listing the files.

-Rn Sort the file names found by the file read mode.

Normally, RSM does not sort alphabetically, but this option will process files in a sorted order instead of the order they were listed or found in the directory tree.

-Rd This mode turn off recursive decent for a directory. This option does not apply to the recursive flag in the file list. See the file list section for more information.

i.e. rsm -Rn *.h *.cpp
rsm -Rn -r c,cpp .\proj
rsm -Rd -r h,cpp "c:\Program Files\proj"

* Recursive Descent Mode

-r <type> <dir>

Recursive descent mode will process all the specified files from a given point in the directory tree. A file spec must be specified as a list of file extensions separated by commas. The starting point must be a directory.

i.e. rsm "options" -r "ext,ext,..." "directory"
rsm -fac -r c,cpp ./proj
rsm -f -a -c -r cc,c,h,hh .\mysrc
rsm -v -RN -r h,c c:\srcs

* Summary Mode

-s

Summary mode summarizes the LOC, eLOC comments and blanks per file. This mode presents the metrics in a summarized format.

i.e. rsm -H -O summary.htm -s *.c
rsm *.c "default if no option specified"

* Totals Only Mode

-T<mode>

This mode only displays project totals for all files processed. In this mode, other reporting data is suppressed. This mode makes it easy to determine grand totals for your project. This mode requires a modifier as specified below. You may use the -k option for sorting the report by different keys.

-Ta All Totals, Functions, Classes, Interfaces, Namespaces Quality Notices and LOC Summary
-Tf Total Functions
-Ti Inheritance Tree Only
-To Total Structs, Classes, Interfaces, Namespaces
-Tq Total Quality Notice Summary List
-Tp Total Quality Notice Profile

```
     -Tn   Quality Notices by file and line number
     -Ts   Total LOC Summary
     -Tl   Summaries Only, No Lists of functions, classes etc.
     -TN   No Filename output for total report

     -Tw   Totals for work file metrics differentials
     i.e.  rsm -Ta -k3 *.c *.cpp
           rsm -Ta -TN -veW -r c,h c:\src
           rsm -Tn -TN -v -r c ./project
```

User Log Mode Management

```
-u<mode>

Administration mode for the RSM User log.  The user log is called
rsm.log and is located in the rsm directory.

     -us   Show User Log

     Network License Options
     -ua   Archive the user log
     -ur   Remove user from the user log

     i.e.  rsm -us
           rsm -ua
           rsm -ur
```

* Verbose Mode

```
-v

Verbose output shows many metrics for each file and a grand
total for all files.  This mode provides many different numbers
which the user can interpret.

     i.e.  rsm -f -o -a -c -m -n -v *.cpp *.c
```

Work Analysis Based on LOC Metric Differentials

```
-w<mode>

Work analysis is the capability to track LOC metrics on a specific
code tree over time.  This facility will create a differential metrics
file for use with the -wW or -wx mode.  The file workfile.dat is the
basis for the -W option and -wx mode can process any two work files.

     -wc  Creates the work files named workfile.dat and a date
          formatted file named rm112220.00 for the date 11-22-2001.

     -w "File usernamedfile.dat"
          This option allows the user to specify the name of the
          generated work file.  The date formated file is replaced
          by the user specifed file.
```

i.e. rsm -w "File mywork.dat" -r h,cpp,java .

-wW Work Analysis from Last Work Analysis File
This mode uses the last work analysis file (workfile.dat) and the current work analysis metrics based upon the specified files. The specified files must match the files specified which originally created the (workfile.dat) file. A work file differential analysis report will be generated.

i.e. rsm -wW -r c,cc,h,hh /project1

-wx Work Analysis Extraction from Work Analysis Files
This mode extracts the work analysis differential between two work analysis files. These work analysis files must be based on the same code tree. The reported metrics will show LOC information based upon the time differential between these two files.

i.e. rsm -wx <ancestor file> <current file>
rsm -H -O diff.htm -wx baseline1.dat baseline2.dat
rsm -A -O diff.csv -wx rm111019.97 rm111419.97

RSM Configuration Parameters

-y or -hc

This option shows the RSM configuration parameters. These settings can be made by specifying their values in the rsm.cfg file.

The emitted format is compliant with the rsm.cfg syntax. You can easily create the rsm.cfg file by redirecting the output to a file called rsm.cfg.

i.e. rsm -y > c:\rsm\rsm.cfg

Please note the rsm.cfg and rsm.lic file must be found in a directory in the system path or at the location specified by the environment variable RSMHOME

~ Support, Sales and Licenses ~

Please contact M Squared Technologies for all RSM licenses. All licenses turn on the wild card support and summations for all files and functions, yielding project metrics.

Paid Licenses include the latest revision, pre-compiled for Window 2000/NT/9x or UNIX. Single user or Network licenses available. UNIX licenses include un-modifiable source code for compiling on your exact machine and UNIX operating system. The source is compilable on any ANSI UNIX C compiler.

If the information you desire is not located in this help listing, please refer to the on-line manual and/or the frequently asked questions FAQ on our website.

Resource Standard Metrics and M Squared Technologies are Trademarks belonging

to M Squared Technologies.

Email - sales@mSquaredTechnologies.com
Web Site - http://mSquaredTechnologies.com
http://ResourceStandardMetrics.com