CFScript CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
7/26/2016	1.0	Original Release	Derek Lengenfelder

Table of Contents

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line containing declaration of data and used by an assembler or compiler to interpret other elements of the program.

The following table lists the CFScript keywords that denote data declaration lines:

import	include	interface
function	property	var

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile. Cold Fusion Script does not have any compiler directives.

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

Cold Fusion Script comment delimiters are "//" and "/*". A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

- An executable line of code may contain the following program control statements:
  - Selection statements (if, ? operator, switch)
  - Iteration statements (for, while, do-while)
  - Empty statements (one or more ";")
  - Jump statements (return, goto, break, continue, exit function)
  - Expression statements (function calls, assignment statements, operations, etc.)

  - Block statements
- An executable line of code may not contain the following statements:
  - Data declaration (data) lines
  - Whole line comments, including empty comments and banners
  - Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) Lines	2	One per line	Defined in 1.4
Compiler Directives	3	NA	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included	
Embedded	5	Not Included	
Banners	6	Not Included	
Empty Comments	7	Not Included	
Blank Lines	8	Not Included	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	"for", "while", "for each" or "if" statement	1	Count once	"while" is an independent statement.
R02	*do {…} while (…); statement*	2	Count once	Braces {…} and semicolon; used with this statement are not counted.
R03	Statements ending by a semicolon	3	Count once per statement, including empty statement.	Semicolons within "for" statement are not counted. Semicolons used with R01 and R02 are not counted.
R04	Block delimiters, braces {…}	4	Count once per pair of braces {..}, except where closing brace is followed by a semicolon, i.e. };or an opening brace comes after a keyword "else".	Braces used with R01 and R02 are not counted. Function definition is counted once since it is followed by {…}.
R05	Compiler directive	5	NA	

3.Examples

EXECUTABLE LINES

SELECTION Statements

ESS1 – if, else if, else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if (<*boolean expression*>) <statements>;	if (x != 0) WriteOutput (“non-zero ”);	1 1

if (<*boolean expression*>)
  <*statements*>;
else
  <*statements*>;
	if (x > 0)
	  WriteOutput (“positive
	”);
	else
	  WriteOutput (“negative
	”);
		1
		1
		0
		1

if (<*boolean expression*>)
  <*statements*>;
else if (<*boolean expression*>)
  <*statements*>;
…
else
  <*statements*>;
	if (x == 0)
	  WriteOutput (“zero”);
	else if (x > 0)
	  WriteOutput (“positive
	”);
	else {
	  WriteOutput (“negative
	”);
	}
		1
		1
		1
		1
		0
		1
		0

NOTE: complexity is not considered, i.e. multiple “&&” or “||” as part of the expression.	if ((x != 0) && (x > 0)) WriteOutput (“positive ”);	1 1

ESS2 - ? :operator

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Exp1?Exp2:Exp3	x > 0 ? WriteOutput (“+ ”) : WriteOutput (“- ”);	1

ESS3 – switch and nested switch statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

switch (<*expression*>)
{
  case <*constant 1*>:
  <*statements*>;
  break;
  default:
  <*statements*>;
	switch (number)
	{
	  case 1:
	  WriteOutput (“case 1
	”);
	  break;
	  default:
	  WriteOutput (“invalid case
	”);
		1
		0
		0
		1
		1
		0
		1

} 	} 	0 0

ESS4 – try-catch

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

try {} catch() {}	try {
	  inputFileName=arg;
	}
	catch (IOException e) {
	  System.err.println(e);
	  System.exit(1);
	}
		1
		1
		0
		1
		1
		1
		0

ITERATION Statements

EIS1 - for

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

for (*initialization; condition; increment*)
  <*statement*>;
 	
NOTE: "for" statement counts as one, no matter how many optional expressions it contains, i.e.
for (i = 0, j = 0; I < 5, j < 10; i++, ,j++)
	for (i = 0; i < 10; i++)
	  WriteOutput(i & "
	");
		1
		1

EIS2 – empty statements (could be used for time delays)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for (i = 0; i < SOME_VALUE; i++) ; 	for (i = 0; i < 10; i++) ; 	2

EIS3 – while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

while (<*boolean expression*>)
  <*statement*>;
 	
	while (i < 10)
	{
	  WriteOutput (i & "
	");
	  i++;
	}
		1
		0
		1
		1
		0

EIS4 – do-while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

do
{
<*statements*>;
} while (<*boolean expression*>);
	do
	{
	ch = getCharacter();
	} while (ch != '\n');
		0
		0
		1
		1

EIS5 – for-each

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for (String name: moreNames) <*statements*>;	for (String n: Names) WriteOutput (ncharAt(0));	1 1

JUMP Statements
(are counted as they invoke action-pass to the next statement)

EJS1 – return

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return *expression*;	If (i == 0) return;	2

EJS2 – break

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break;	If (i > 10) break;	2

EJS3 – exit function

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
exit *return_code*;	If (x < 0) exit 1;	2

EJS4 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

continue;	while (!done)
	{
	ch = getchar();
	if (char == '\n')
	{
	done = true;
	continue;
	}
	}
		1
		0
		1
		1
		0
		1
		1
		0
		0

EXPRESSION Statements		

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<*function_name*> (<*parameters*>);	read_file (name);	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

<*name*> = <*value*>	x = y;
	char name[6] = “file1”;
	a = 1; b = 2; c = 3;
		1
		1
		3

EES3 - empty statement(is counted as it is considered to be a placeholder for something to call attention)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more “;” in succession	;	1 per each
BLOCK Statements		

EBS1 – block means related statements treated as a unit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

{ /* start of block */
<definitions>
<statement>
} /* end of block */
	/* start of block */
	{
	i = 0;
	System.out.print (“%d”, i);
	} /* end of block */
		0
		0
		1
		1
		0

DECLARATION OR DATA LINES

DDL1 – function prototype variable declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<*function*>	include template="myinclude.cfm" import "tag library location"	1 1

COMPILER DIRECTIVES

CDL1 – directive type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
#define <name> <value> #include <library_name>	#define MAX_SIZE 100 #include <stdio.h>	1 1

4.Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

Math Functions	Trig	Log	Calculations	Conditionals	Logic	Assignment
abs	acos	log	+	if	==	=
arrayavg	asin	log10	-	else if	!=	
arraysum	atn		*	else	>	
ceiling	cos		/	case	<	
decrementvalue	sin		**	for	>=	
exp	tan			while	<=	
fix				try	eq	
incrementvalue				catch	neq	
int					gt	
max					lt	
min					gte	
mod					lte	
pi						
precisionvaluate						

5.Notes on Special Character Processing

1) Quotes:

Start of Quotes: "\""

End of Quotes: "\""

2) File extension recognized for CFScript: ".cfs"