ColdFusion CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
5/11/2010	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 that contains declaration of data and used by a ColdFusion server to determine all cold fusion variables declared in the program.
The following lists the Cold Fusion keywords that denote data declaration lines:

cfapplication	cfapplet	cfargument	cfcomponent	cffunction
cfimport	cfinclude	cfinterface	cfproperty	cfset

Cold Fusion Data Keywords

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile. ColdFusion does not have 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.
ColdFusion 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, switch)
  - Iteration statements (foreach, loop)
  - Empty statements (pass)
  - Jump statements (return, goto, break, exit function)

  - Expression statements (function calls, assignment statements, operations, etc.)
  - Block statements
  - Database 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	All ColdFusion tags beginning with “cf” with no nesting like <cfform>, <cfinput>, etc.	1	Count once	All occurrences of such tags until corresponding end tags </> is assumed to be one logical statement
R02	<cfcase>, <cfloop>, <cfswitch>, either all tags having multiple steps of execution statement	2	Count once	Logically different tags on the same line are to be counted independently
R03	Comment delimiter	3	Count once per combination of start tag and end tag statement, including empty statement.	Comments in ColdFusion are similar to HTML comments<!--- this is a comment --->
R04	Compiler directive	4	NA	NA

3. Examples

EXECUTABLE LINES

SELECTION Statements

ESS1 - cfif, cfelseif, cfelse and nested cfif statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfif *expression*>	<cfif name EQ "USC">	1
statements	some logic	0
</cfif>	</cfif>	0
<cfif *expression*>	<cfif password EQ "name">	1
statements	some code	0
<cfelse>	<cfelse>	1
statements	statements	0
</cfif>	</cfif>	0
<cfif *expression*>	<cfif num GT 0>	1
statements	some code	0
<cfelseif *expression*>	<cfelseif expression>	1
statements	statements	0
<cfelse>	<cfelse>	1
statements	code	0
</cfif>	</cfif>	0
NOTE: complexity is not considered, i.e. multiple “and” or “or” as part of the expression.		

ESS2 - cfswitch, cfcase, cfdefaultcase

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfswitch *expression* = *“expression”*>	<cfswitch expression = “#State#”>	1
<cfcase *value* =*”value”*>	<cfcase value=”CA”>	1
HTML or CFML code	California	0
</cfcase>	</cfcase>	0
<cfdefaultcase>	<cfdefaultcase>	1
HTML or CFML code	one of the other 49 states	0
</cfdefaultcase>	</cfdefaultcase>	0
</cfswitch>	</cfswitch>	0

ESS3 - cftry-cfcatch

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cftry >	<cftry>	1
do something	try: 1/0	0
</cftry>	some code	0
<cfcatch[Error]>	</cftry>	0

cleanup </cfcatch>	<cfcatch ZeroError> some code </cfcatch>	1 0 0

ITERATION Statements

EIS1 - cfloop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfloop *expression*> *statements* </cfloop>	<cfloop index = "LoopCount" from = 1 to = 5> The loop index is <cfoutput>#LoopCount#</cfoutput>. </cfloop> <cfloop condition =”Expression”> <cfloop> <cfloop query =”Query name” startRow =”Start Row value” endRow = “ End Row value” </cfloop>	1 0 0 1 0 0 1 0 1 0 0 0

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

EJS1 – cfreturn

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfreturn *expression*>	<cfreturn true>	1

EJS2 – cfbreak

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfbreak>	<cfloop expression> <cfbreak> </cfloop>	1 1 0

EJS3 - cfexit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfexit>	<cfloop expression> <cfexit> </cfloop>	1 1 0

EJS4 - cfcontinue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfcontinue>	<cfloop expression> <cfcontinue>	1 1

	</cfloop>	0

EXPRESSION Statements

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cffunction >	<cffunction >	1
name = “function name” description	name = "function name"	0
“ function description” return Type=”Data	description "function description"	0
to be returned	return Type="Data to be	0
</cffuntion>	returned”	0
	</cffuntion>	0

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfset *variable_name=”value”*>	<cfset age=”22”>	1

EES3 - CFScript

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfscript>	<cfscript>	1
CFScript code	for (i=1 ; i LE 4; i = i+1) {	0 (3 script)
</cfscript>	if(find("key",strings[i],1))	0
	break;	0
	}	0
	</cfscript>	0

EES4 – database query

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<cfquery datasource=”DB name” name=	<cfquery datasource=“ABCD”	1
“Some name”>	name=”Student Records”>	0
query or stored procedure call	SELECT * FROM Students	0 (1 sql)
</cfquery>	</cfquery>	0

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	+	cfif	eq	=
arrayavg	asin	log10	-	cfelseif	neq	
arraysum	atn		*	cfelse	gt	
ceiling	cos		/	cfmatch	gte	
decrementvalue	sin		++	cfcase	lt	
exp	tan		--	cfloop	lte	
fix			mod		&&	
incrementvalue			%		||	
int			^		not	
max					and	
min					or	
mod					xor	
pi					equiv	
precisionvaluate					imp	
rand					is	
randomize						
randrange						
round						
sgn						
sqr						