JPL Mapping to CodeSonar®

Category ID	Category Name	CodeSonar Class Mnemonic	CodeSonar Class Name	Relationship Type (category to class)
JPL:2	Compile with all warnings enabled; use static source code analyzers.	BUILD.WALL	Not All Warnings Are Enabled	closely mapped
JPL:2	Compile with all warnings enabled; use static source code analyzers.	BUILD.WERROR	Warnings Not Treated As Errors	closely mapped
JPL:3	Use verifiable loop bounds for all loops meant to be terminating.	LANG.STRUCT.LOOP.UB	Potential Unbounded Loop	closely mapped
JPL:4	Do not use direct or indirect recursion.	LANG.FUNCS.RECURSION	Recursion	closely mapped
JPL:5	Do not use dynamic memory allocation after task initialization.	ALLOC.POSTINIT	Dynamic Allocation After Initialization	closely mapped
JPL:7	Do not use task delays for task synchronization.	CONCURRENCY.BADFUNC.DELAY	Task Delay Function	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.ORDER	Conflicting Lock Order	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.MISMATCH	Lock/Unlock Mismatch	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.TWICE	Locked Twice	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.NOLOCK	Missing Lock Acquisition	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.NOUNLOCK	Missing Lock Release	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.NESTED	Nested Locks	closely mapped
JPL:9	Place restrictions on the use of semaphores and locks.	CONCURRENCY.LOCK.UNKNOWN	Unknown Lock	closely mapped
JPL:11	Do not use goto, setjmp or longjmp.	LANG.STRUCT.GOTO	Goto Statement	closely mapped
JPL:11	Do not use goto, setjmp or longjmp.	BADFUNC.LONGJMP	Use of longjmp	closely mapped
JPL:11	Do not use goto, setjmp or longjmp.	BADFUNC.SETJMP	Use of setjmp	closely mapped
JPL:12	Do not use selective value assignments to elements of an enum list.	LANG.STRUCT.INIT.ENUM	Inconsistent Enumerator Initialization	closely mapped
JPL:13	Declare data objects at smallest possible level of scope.	LANG.STRUCT.DECL.MGT	Global Variable Declared with Different Types	closely mapped
JPL:13	Declare data objects at smallest possible level of scope.	LANG.STRUCT.DECL.MG	Multiple Declarations of a Global	closely mapped

JPL Mapping to CodeSonar®

JPL:13	Declare data objects at smallest possible level of scope.	LANG.STRUCT.SCOPE.FILE	Scope Could Be File Static	closely mapped
JPL:13	Declare data objects at smallest possible level of scope.	LANG.STRUCT.SCOPE.LOCAL	Scope Could Be Local Static	closely mapped
JPL:14	Declare data objects at smallest possible level of scope.	LANG.FUNCS.IRV	Ignored Return Value	closely mapped
JPL:15	Check the validity of values passed to functions.	LANG.STRUCT.UPD	Unchecked Parameter Dereference	closely mapped
JPL:16	Use static and dynamic assertions as sanity checks.	LANG.FUNCS.ASSERTS	Not Enough Assertions	closely mapped
JPL:17	Use U32, I16, etc instead of predefined C data types such as int, short, etc.	LANG.TYPE.BASIC	Basic Numerical Type Used	closely mapped
JPL:19	Do not use expressions with side effects.	LANG.STRUCT.SE.COND	Condition Contains Side Effects	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.COND	Conditional Compilation	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.MACROEND	Macro Does Not End With } or)	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.MACROSTART	Macro Does Not Start With { or (	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.PASTE	Macro Uses ## Operator	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.RECURSIVE	Recursive Macro	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.UNBALANCED	Unbalanced Parenthesis	closely mapped
JPL:20	Make only very limited use of the C pre-processor.	LANG.PREPROC.VARIADIC	Variadic Macro	closely mapped
JPL:21	Do not define macros within a function or a block.	LANG.PREPROC.DEFINFN	Macro Defined in Function Body	closely mapped
JPL:21	Do not define macros within a function or a block.	LANG.PREPROC.UNDEFINFN	Macro Undefined in Function Body	closely mapped
JPL:22	Do not undefine or redefine macros.	LANG.PREPROC.RUNDEF	Macro Undefinition of Reserved Name	closely mapped
JPL:22	Do not undefine or redefine macros.	LANG.PREPROC.UNDEF	Use of #undef	closely mapped
JPL:23	Place #else, #elif, and #endif in the same file as the matching #if or #ifdef.	LANG.PREPROC.NOENDIF	No Matching #endif	closely mapped

JPL Mapping to CodeSonar®

JPL:23	Place #else, #elif, and #endif in the same file as the matching #if or #ifdef.	LANG.PREPROC.NOIF	No Matching #if	closely mapped
JPL:24	Place no more than one statement or declaration per line of text.	LANG.STRUCT.DECL.ML	Multiple Declarations On Line	closely mapped
JPL:24	Place no more than one statement or declaration per line of text.	LANG.STRUCT.MULTISTMT	Multiple Statements On Line	closely mapped
JPL:25	Use short functions with a limited number of parameters.	LANG.FUNCS.TOOLONG	Function Too Long	closely mapped
JPL:25	Use short functions with a limited number of parameters.	LANG.FUNCS.TMFP	Too Many Parameters	closely mapped
JPL:26	Use no more than two levels of indirection per declaration.	LANG.STRUCT.TMID	Too Much Indirection in Declaration	closely mapped
JPL:27	Use no more than two levels of dereferencing per object reference.	LANG.STRUCT.TMD	Too Many Dereferences	closely mapped
JPL:28	Do not hide dereference operations inside macros or typedefs.	LANG.PREPROC.ARROW	Macro Uses -> Operator	closely mapped
JPL:28	Do not hide dereference operations inside macros or typedefs.	LANG.PREPROC.STAR	Macro Uses Unary * Operator	closely mapped
JPL:28	Do not hide dereference operations inside macros or typedefs.	LANG.PREPROC.BRACES	Macro Uses [] Operator	closely mapped
JPL:28	Do not hide dereference operations inside macros or typedefs.	LANG.STRUCT.PIT	Pointer Type Inside Typedef	closely mapped
JPL:30	Do not cast function pointers into other types.	LANG.STRUCT.FUNCPTR.CONVERT	Function Pointer Conversion	closely mapped
JPL:31	Do not place code or declarations before an #include directive.	LANG.PREPROC.CBI	Code Before #include	closely mapped