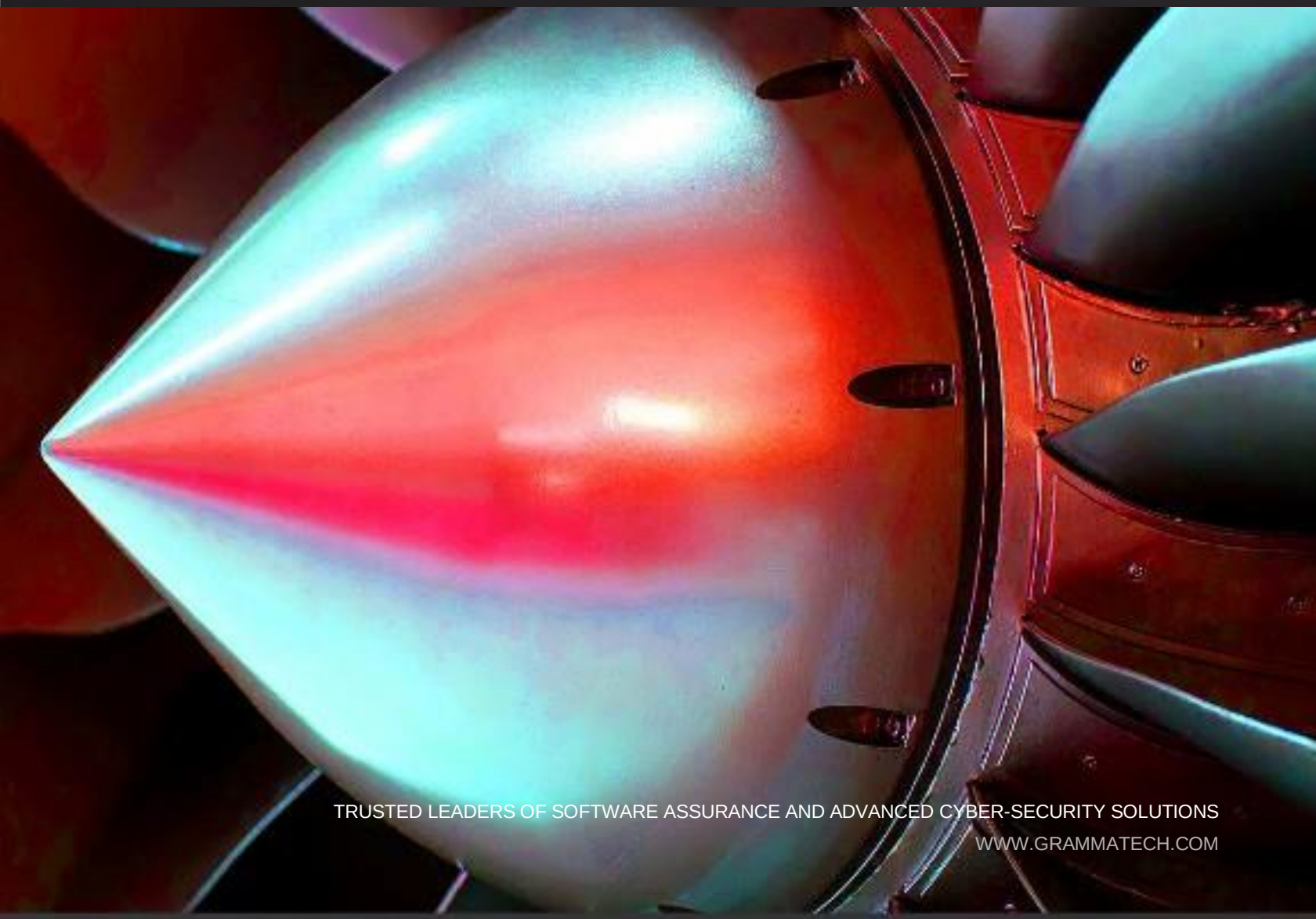
GRAMMATECH

JPL INSTITUTIONAL CODING STANDARD FORTHE C PROGRAMMING LANGUAGE | CLOSE & BROAD MAPPING TO CODESONAR® 7.3



TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

The following table shows the CodeSonar warning classes that are associated with JPL rules.

Note close and broad mappings are identical, thus only one chart below to reflect both close and broad mapping.

Rule	Rule Name	Supported
JPL:1	Do not stray outside the language definition.	Yes
JPL:2	Compile with all warnings enabled; use static source code analyzers.	Yes
JPL:3	Use verifiable loop bounds for all loops meant to be terminating.	Yes
JPL:4	Do not use direct or indirect recursion.	Yes
JPL:5	Do not use dynamic memory allocation after task initialization.	Yes
JPL:6	Use IPC messages for task communication.	No
JPL:7	Do not use task delays for task synchronization.	Yes
JPL:8	Explicitly transfer write-permission (ownership) for shared data objects.	No
JPL:9	Place restrictions on the use of semaphores and locks.	Yes
JPL:10	Use memory protection, safety margins, barrier patterns.	No
JPL:11	Do not use goto, setjmp or longjmp.	Yes
JPL:12	Do not use selective value assignments to elements of an enum list.	Yes
JPL:13	Declare data objects at smallest possible level of scope.	Yes
JPL:14	Check the return value of non-void functions, or explicitly cast to (void).	Yes
JPL:15	Check the validity of values passed to functions.	Yes
JPL:16	Use static and dynamic assertions as sanity checks.	Yes
JPL:17	Use U32, I16, etc instead of predefined C data types such as int, short, etc.	Yes
JPL:18	Make the order of evaluation in compound expressions explicit.	Yes
JPL:19	Do not use expressions with side effects.	Yes
JPL:20	Make only very limited use of the C pre-processor.	Yes
JPL:21	Do not define macros within a function or a block.	Yes
JPL:22	Do not undefine or redefine macros.	Yes
JPL:23	Place #else, #elif, and #endif in the same file as the matching #if or #ifdef.	Yes
JPL:24	Place no more than one statement or declaration per line of text.	Yes
JPL:25	Use short functions with a limited number of parameters.	Yes
JPL:26	Use no more than two levels of indirection per declaration.	Yes
JPL:27	Use no more than two levels of dereferencing per object reference.	Yes
JPL:28	Do not hide dereference operations inside macros or typedefs.	Yes
JPL:29	Do not use non-constant function pointers.	No
JPL:30	Do not cast function pointers into other types.	Yes
JPL:31	Do not place code or declarations before an #include directive.	Yes