Coverage for ISO/IEC 8652:2012 and subsequent corrections in ACATS 3.x and 4.x

Subclause B.3.3

A Key to Kinds and subkinds is found on the sheet named Key. Tests new to ACATS 3.0 are shown in **bold**; ACATS 3.1 in ***bold italic***; ACATS 4.0 in **blue bold**; ACATS 4.1 in ***blue bold italic***. ACATS 4.2 in ***green bold italic***.

Clause	Para.	Lines	Kind	Subkind	Notes	Tests	New	Objective's Priority	Objective Text	Objective notes	Submitted tests (will need work).
B.3.3	(1/3)		General								
	(2/3)		Deleted								
	(3/3)		Deleted								
	(3.1/3)		StaticSem	Portion	Lead-in for next paragraph.						
	(3.2/3)	1	StaticSem			***CXB3019, CXB3023*** (pragma), ***CXB3024*** (aspect)	All		Check that aspect Unchecked_Union can be specified for a discriminated record type with a variant part.		
				Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot be specified for a non-record type.		
				Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot be specified for a record type that has no discriminants.		
				Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot be specified for a discriminated record type that does not have a variant part.		
				Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot be specified for a derived record type whose parent type has primitive operations.		
								6	Check that aspect Unchecked_Union can be specified with a static expression of type Boolean, and that expression can have parts that are imported from other units.	C-Test. Could be similar to test CA21002. Also should consider testing with a constant declared after the type (but before the freezing point).	
				Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot specified with an expression of a type other than Boolean.		
		2		Negative		***BXB3004***	All		Check that aspect Unchecked_Union cannot be specified with a non-static Boolean expression.		
		3		Widely Used	Any type not using Unchecked_Union tests this.						
	(4/3)		Deleted								
	(5/3)		Deleted								
	(6/3)	1	Definitions		“unchecked union type”						
		2	Definitions		“unchecked union subtype”						
		3	Definitions		“unchecked union object”						
	(7/2)		Legality		Note: The negative is untestable, since an implementation can define all types to be C-compatible if it wants.	***CXB3019, CXB3023*** (pragma), ***CXB3024*** (aspect)	All		Check that aspect Unchecked_Union can be specified if all of the components are C-Compatible.		
	(8/2)		Legality			***BXB3001, CXB3019***	All		Check that a component of an unchecked union can be another unchecked union that depends on the original union's discriminant.		
				Negative		***BXB3001***	All		Check that an Unchecked_Union is illegal if there is any component that is not an unchecked union and whose constraint depends on a discriminant.		
	(9/3)		Legality			***BXB3001, CXB3019***	All		Check that a discriminant of an unchecked union can be used to control a variant of the record type, or in a discriminant constraint of another unchecked union.		
				Negative		***BXB3002***	All		Check that the name of a discriminant of an unchecked union cannot be used outside of the type declaration.		

		Negative		***BXB3002***	All		Check that the name of a discriminant of an unchecked union cannot be used in a record representation clause for the type.	
(10/3)	Legality					7	Check that a component of a variant of an unchecked union cannot need finalization.	B-Test. But could have a different error if it is impossible to declare a C-compatible type that needs finalization. Allow that as an error.
						5	Check that if the type of a component of a variant of an unchecked union declared in a generic specification is a generic formal type, the actual type corresponding to that formal type cannot need finalization.	B-Test. But could have a different error if it is impossible to declare a C-compatible type that needs finalization. Allow that as an error.
						7	Check that type of a component of a variant of an unchecked union declared in a generic body cannot be a generic formal private type or private extension.	B-Test. Don't forget generic children.
(11/2)	Legality			***BXB3003***	All		Check that the completion of an incomplete type with a discriminant part cannot be an unchecked union type.	
				BXB3003	All		Check that the full type of a private type with a known discriminant part cannot be an unchecked union type.	
(12/2)	Legality					5	Check that an unchecked union type can be the actual for a generic formal private type without discriminants.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
						5	Check that an unchecked union type can be the actual for a generic formal private type with unknown discriminants.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
						5	Check that an unchecked union type can be the actual for a generic formal derived type whose ancestor does not have discriminants.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
						5	Check that an unchecked union type can be the actual for a generic formal derived type whose ancestor is an unchecked union type.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
		Negative				7	Check that an unchecked union type cannot be the actual for a generic formal private type with a known discriminant part.	B-Test.
		Negative				7	Check that an unchecked union type cannot be the actual for a generic formal derived type whose ancestor has discriminants and is not an unchecked union type.	B-Test.
(13/2)	StaticSem			***CXB3019, CXB3023*** (pragma), ***CXB3024*** (aspect)	All		Check that an unchecked union type can have convention C.	Put a usage-oriented test here (it doesn't have a natural objective).
(14/2)	StaticSem	Not Testable	Could check that objects do have the same size, but that would require guessing the circumstances where a compiler would not make that true (it would be true for variants on most compilers anyway).					
(15/2)	StaticSem					4	Check that the size of the discriminants of an unchecked union object is zero.	C-Test. Not very important as a direct test.
(16/2)	StaticSem	Not Testable	Suppression is a permission; in particular, a compiler can make one of these checks if it can do so without reading the discriminant value (for instance, if it can infer the value). Therefore, this can't be tested.					

(17/2)	StaticSem	Not Testable	See above.					
(18/2)	StaticSem	Not Testable	See above.					
(19/2)	StaticSem	Not Testable	See above.					
(20/2)	Definitions		“inferable discriminants”					
(21/2)	Definitions		“expression with inferable discriminants”					
(22/2)	Dynamic	Portion	Lead-in for the following rules.					
(23/2)	Dynamic			***CXB3019, CXB3023*** (pragma), ***CXB3024*** (aspect)	Part	4	Check that Program_Error is raised by the predefined equality operator for an unchecked union type if either operand does not have inferable discriminants.	C-Test. Try examples in an individual membership test.
						5	Check that Program_Error is not raised by an equality operation for an unchecked union type if the type has a user-defined primitive equality operation.	C-Test. Try explicit uses of equality, as well as those in a composed equality and in an individual membership test.
(24/2)	Dynamic			***CXB3020***	All		Check that Program_Error is raised by the predefined equality operator for any type that has a subcomponent of an unchecked union type whose nominal subtype is unconstrained.	
(25/2)	Dynamic			***CXB3021***	All		Check that Program_Error is raised by a membership test if a subtype_mark denotes a constrained unchecked union subtype and the expression lacks inferable discriminants.	
(26/2)	Dynamic			***CXB3022***	All		Check that Program_Error is raised by the conversion from a derived unchecked union type to an unconstrained non-unchecked-union type if the operand of the conversion lacks inferable discriminants.	
(27/2)	Dynamic					7	Check that Program_Error is raised by the default implementation of the Write or Read attribute of an unchecked union type.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
(28/2)	Dynamic					7	Check that Program_Error is raised by the default implementation of the Output or Input attribute of an unchecked union type if the type lacks default discriminant values.	C-Test. All of the components have to either have Convention C, or be declared in one of the C interface packages.
(29/3)	Deleted							
(30/2)	NonNormative		Part of a note.					
(31/3)	NonNormative		Part of a note.					
(32/2)	NonNormative		Part of a note.					

Paragraphs:		Objectives with tests:	Objectives to test:	Total objectives:	Objectives with submitted tests:
1	34	20	15	34	0

Must be tested	Objectives with Priority 10	0
	Objectives with Priority 9	0
Important to test	Objectives with Priority 8	0
	Objectives with Priority 7	6
Valuable to test	Objectives with Priority 6	1
	Objectives with Priority 5	6
Ought to be tested	Objectives with Priority 4	2
	Objectives with Priority 3	0
Worth testing	Objectives with Priority 2	0
Not worth testing	Objectives with Priority 1	0
	Total:	15

Objectives covered by new tests since ACATS 2.6	20
Completely:	19