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

Clause 3.2.4

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).
3.2.4	(1/3)	1	Definitions		“predicate aspect”						
		2	Definitions		“predicate specification”						
		3	General		Nothing to test here.						
	(2/3)		NameRes			**B324001**	All		Check that a predicate expression must have a boolean type.		
						B324001	Part	4	Check that a predicate expression that makes a call on an overloaded function will resolve if exactly one such function has a boolean type.	B324001 tries this, but we should try an executable C-Test. Low priority because this is normal resolution and should be similar to conditions in an if statement (and many other cases).	
								5	Check that a predicate expression can have a boolean type other than Boolean.	C-Test (or subtest in a B-Test).	
	(3/3)	1	Legality	Subpart	Any legal predicate specification will test.						
								6	Check that a dynamic predicate can be specified for a private type.	C-Test. (Not completely clear that we intended this to work, and it appears that it could also be specified for the full definition, which is weird – sent an e-mail to the ARG).	
				Negative		**B324001**	All		Check that a predicate cannot be specified on entities other than subtype or type declarations.		
		2	Definitions	Lead-in	“applies”						
	(4/4)		StaticSem		Modified by AI12-0071-1 and AI12-0099-1. We test here as this gets complicated otherwise.			7	Check that the predicates that apply to the parent subtype and progenitors, if any, are tested as part of the predicate test for a derived type.	C-Test. Use a membership to trigger evaluation of the predicate test to avoid issues with disabled predicates.	
								6	Check that the predicates that apply to the progenitors, if any, are tested as part of the predicate test for a task type.	C-Test. Use a membership to trigger evaluation of the predicate test to avoid issues with disabled predicates. Be sure to try a single task object.	
								6	Check that the predicates that apply to the progenitors, if any, are tested as part of the predicate test for a protected type.	C-Test. Use a membership to trigger evaluation of the predicate test to avoid issues with disabled predicates. Be sure to try a single protected object.	
	(5/3)					***C324003***	All		Check that the predicates that apply to the subtype_mark of a subtype_declaration, if any, are tested as part of the predicate test for a subtype.		
	(6/4)		Deleted		Deleted by AI12-0071-1.						
	(7/3)		Definitions	Lead-in	“enabled” and “disabled” predicates.						
	(8/3)		StaticSem	Lead-in							
								6	Check that a static predicate specified directly for a type or subtype is checked when it is enabled (the applicable assertion policy is Check).	C-Test. Still need to test types. We test this here as the default assertion policy is implementation-defined, so we can't assume that checks are commonly used – and that also makes this critical, thus the priority. This objective is about enabling the check, so we only need to check subtype conversions for it.	
	(9/3)		StaticSem			**C324001, *C324006***	Part				

(10/3)	StaticSem		C324001, *C324006*	All		Check that a dynamic predicate specified directly for a type or subtype is checked when it is enabled (the applicable assertion policy is Check).	C324006 tests specific forms of the pragma with a type. We could try more cases (global form of pragma with a type, specific with a subtype, but we've reached dimishing returns for that).
(11/3)	StaticSem		*C324004, C324005*	Part	5	Check that a static predicate specified directly for a type or subtype is not checked when it is disabled (the applicable assertion policy is Ignore).	C-Test. Still need to test that the policy at the point of the check is irrelevant (that is, try it with "Check"). Less important since most users disable checks globally. Combine with below??
			C324004, C324005	Part	5	Check that a dynamic predicate specified directly for a type or subtype is not checked when it is disabled (the applicable assertion policy is Ignore).	C-Test. Still need to test that the policy at the point of the check is irrelevant (that is, try it with "Check"). Less important since most users disable checks globally. Combine with below??
					3	Check that a static predicate inherited for a subtype is checked even when it is disabled if the subtype has its own predicate that is enabled.	C-Test. Check using both the global assertion policy and using the specific form of the pragma. This is low priority because assertion policy is usually applied globally (compiler switch) or on a package basis, so it's unlikely that the policy would be different in practice.
					3	Check that a dynamic predicate inherited for a subtype is checked even when it is disabled if the subtype has its own predicate that is enabled.	C-Test. Check using both the global assertion policy and using the specific form of the pragma. See the previous item for priority discussion.
					3	Check that a static predicate inherited for a subtype is not checked even when it is enabled if the subtype has its own predicate that is disabled.	C-Test. Check using both the global assertion policy and using the specific form of the pragma. See the previous item for priority discussion.
					3	Check that a dynamic predicate inherited for a subtype is not checked even when it is enabled if the subtype has its own predicate that is disabled.	C-Test. Check using both the global assertion policy and using the specific form of the pragma. See the previous item for priority discussion.
			C324004, C324005	All		Check that a static predicate specified directly for a type or subtype still is evaluated for a membership even if it is disabled (the applicable assertion policy is Ignore).	
			C324004, C324005	All		Check that a dynamic predicate specified directly for a type or subtype still is evaluated for a membership even if it is disabled (the applicable assertion policy is Ignore).	
			C324004, C324005	All		Check that a static predicate specified directly for a type or subtype still is evaluated for the Valid attribute even if it is disabled (the applicable assertion policy is Ignore).	
			C324004, C324005	All		Check that a dynamic predicate specified directly for a type or subtype still is evaluated for the Valid attribute even if it is disabled (the applicable assertion policy is Ignore).	
			C324004, C324005	All		Check that a static predicate specified directly for a type or subtype still determines the items iterated by a for loop on the type or subtype even if it is disabled (the applicable assertion policy is Ignore).	
(12/4)	StaticSem	AI12-0099-1 clarifies the wording to ensure it covers all kinds of types.			6	Check that a static predicate specified for the parent or progenitor of a derived type is checked for the derived type when predicate of the parent or progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. These inherited cases are considerably less important.
					6	Check that a dynamic predicate specified for the parent or progenitor of a derived type is checked for the derived type when the predicate of the parent or progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.

			5	Check that a static predicate specified for the progenitor of a task or protected type is checked for the type when the predicate of the progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Don't forget to try a single task and single protected object.
			5	Check that a dynamic predicate specified for the progenitor of a task or protected type is checked for the type when the predicate of the progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Don't forget to try a single task and single protected object.
			3	Check that a static predicate specified for the parent or progenitor of a derived type is checked for the derived type even when it is disabled if the predicate of some other parent or progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Low priority because policy is usually applied globally or on a package basis, so having different policies is unlikely.
			3	Check that a dynamic predicate specified for the parent or progenitor of a derived type is checked for the derived type even when the predicate is disabled if the predicate of some other parent or progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Low priority because policy is usually applied globally or on a package basis, so having different policies is unlikely.
			3	Check that a static predicate specified for the progenitor of a task or protected type is checked for the type even when the predicate is disabled if the predicate of some other progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Low priority because policy is usually applied globally or on a package basis, so having different policies is unlikely.
			3	Check that a dynamic predicate specified for the progenitor of a task or protected type is checked for the type even when the predicate is disabled if the predicate of some other progenitor is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma. Low priority because policy is usually applied globally or on a package basis, so having different policies is unlikely.
(13/3)	StaticSem		7	Check that a static predicate specified for subtype S is checked for a subtype directly of S that does have any predicates of its own when the predicate of S is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			7	Check that a dynamic predicate specified for subtype S is checked for a subtype directly of S that does have any predicates of its own when the predicate of S is enabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
(14/3)	StaticSem		5	Check that a static predicate specified for the parent or progenitor of a derived type is not checked for the derived type when the predicates of the parent and all progenitors are disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			5	Check that a dynamic predicate specified for the parent or progenitor of a derived type is not checked for the derived type when the predicates of the parent and all progenitors are disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			4	Check that a static predicate specified for the progenitor of a task or protected type is not checked for the type when the predicate of all of the progenitors are disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			4	Check that a dynamic predicate specified for the progenitor of a task or protected type is not checked for the type when the predicate of all of the progenitors are disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			6	Check that a static predicate specified for subtype S is not checked for a subtype directly of S that does have any predicates of its own when the predicate of S is disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.
			6	Check that a dynamic predicate specified for subtype S is not checked for a subtype directly of S that does have any predicates of its own when the predicate of S is disabled.	C-Test. Check using both the global assertion policy and the specific form of the pragma.

(14.1/4)	Legality	Subpart	Added by AI12-054-2. Any legal predicate failure aspect will test.					
		Negative		***B324004***	All		Check that the Predicate_Failure aspect cannot be specified on a type or subtype that does not also have a Static_Predicate or Dynamic_Predicate aspect specified.	
				B324004	All		Check that a Predicate_Failure aspect cannot be given on an entity other than a type or subtype.	
(14.2/4)	StaticSem	Portion	Added by AI12-054-2. Will be tested by the 14.3/3 and 31.1/4 rules.					
(14.3/4)	NameRes		Added by AI12-054-2.	***B324004***	All		Check that the expression of a Predicate_Failure aspect must have type String.	
						5	Check that a predicate expression that makes a call on an overloaded function will resolve if exactly one such function has type String.	C-Test. This is normal resolution.
(15/3)	Definitions		“predicate static”					
(16/3)	Legality					6	Check that the expression of a Static_Predicate can be a static expression.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot contain a non-static function call.	
		Negative	Some other cases are allowed by other bullets.	**B324001**	All		Check that the expression of a Static_Predicate cannot contain a non-static call to a arithmetic operator, including predefined operators operating on the current instance.	
(17/3)	Legality					6	Check that the expression of a Static_Predicate can be a predicate static membership.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a non-static membership whose tested expression is not the current instance.	
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a membership where one or more choices is non-static even when the tested expression is the current instance.	
(18/3)	Legality					6	Check that the expression of a Static_Predicate can be a predicate static case expression.	C-Test. Use a membership test to check the value of the expression.
		Negative				7	Check that the expression of a Static_Predicate cannot be a non-static case expression whose tested expression is not the current instance.	B-Test.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a case expression where one or more dependent_expressions is non-static even when the selecting expression is the current instance.	
(19/3)	Legality					6	Check that the expression of a Static_Predicate can be a predicate static ordering or equality expression.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a non-static call to a predefined ordering or equality operator if neither operand is the current instance.	
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a call to a predefined ordering or equality operator where one operand is non-static even when the other operand is the current instance.	
(20/3)	Legality		AI12-0099-1 changes this to enumerate the operators as “boolean logical operator” doesn't include **not**.			6	Check that the expression of a Static_Predicate can be a predicate static expression using a boolean operator and, or, xor, or not.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a call to a predefined boolean operator and, or, xor, or not if either operand is not predicate static.	

(21/3)	Legality					6	Check that the expression of a Static_Predicate can be a short circuit control form with predicate static operands.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a short circuit control form if either operand is not predicate static.	
(22/3)	Legality					6	Check that the expression of a Static_Predicate can be a parenthesized expression with a predicate static operand.	C-Test. Use a membership test to check the value of the expression.
		Negative		**B324001**	All		Check that the expression of a Static_Predicate cannot be a parenthesized expression if the operand is not predicate static.	
(23/3)	Legality			**B324001**	Part	5	Check that a predicate cannot be specified for an incomplete subtype.	B-Test. Still need to check for a subtype of a limited view.
(24/3)	Legality					6	Check that predicate cannot mention any subtype to which the predicate applies.	B-Test. Try the example in the AARM, at a minimum.
(25/3)	Legality			**B324002, B324003**	All		Check that an index subtype of an unconstrained array declaration cannot denote a subtype with a predicate.	
				B324002	All		Check that the discrete range of a slice or index constraint cannot denote a subtype with a predicate.	Note: It does not appear possible to create a generic to test this in an instance, as we cannot get the type of the array correct without causing some other error.
				B324002, B324003	All		Check that the discrete subtype definition of a constrained array declaration, entry declaration, or entry index specification cannot denote a subtype with a predicate.	
(26/3)	Legality			**B324002, B324003**	All		Check that the prefix of First, Last, and Range cannot be a subtype with a predicate.	
(27/3)	Legality			**B324002, B324003**	All		Check that the subtype name given in a for loop cannot have a dynamic predicate.	
				B324002, B324003	All		Check that the subtype name given in a for loop cannot be nonstatic and have a static predicate.	
(28/3)	Legality			**B324002**	All		Check that the discrete choice of an array aggregate cannot name a non-static subtype that has a static predicate.	Note: It does not appear possible to create a generic to test this in an instance, as we cannot get the type of the array correct without causing some other error.
				B324002	All		Check that the discrete choice of an array aggregate cannot name a subtype that has a dynamic predicate.	Note: It does not appear possible to create a generic to test this in an instance, as we cannot get the type of the array correct without causing some other error.
(29/3)	Legality	Subpart	Generic boilerplate; test in previous objectives.					
(29.1/4)	Dynamic		Rule moved here from 3.2.4(33/3) by AI12-0071-1.			7	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the prefix of First, Last, or Range.	C-Test.
						7	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the index subtype of an unconstrained array declaration.	C-Test.
						7	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the discrete range of a slice or index constraint.	C-Test.

								6	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the discrete subtype definition of a constrained array declaration, entry declaration, or entry index specification.
								7	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the subtype name given in a for loop.
								7	Check that Program_Error is raised if the actual for a formal discrete or integer type F has a predicate, and the body of the generic unit uses F as the discrete choice of an array aggregate.
(29.2/4)		Dynamic	Lead-in	Rules clarified by AI12-0071-1.					
(29.3/4)		Dynamic		Rules new from AI12-0071-1.				9	Check that for a subtype conversion to a subtype with predicates enabled, if a constraint or null exclusion fails, no predicates are evaluated.
								9	Check that for a membership test, if a constraint or null exclusion fails, no predicates are evaluated.
(29.4/4)		Dynamic	Lead-in	Rules new from AI12-0071-1.					
(29.5/4)		Dynamic		Rules new from AI12-0071-1.				7	Check that for a type conversion to a derived type with predicates enabled, if a predicate of the parent or progenitor fails, no predicates of the type are evaluated.
								6	Check that for a type conversion to a task or protected type with predicates enabled, if a predicate of a progenitor fails, no predicates of the type are evaluated.
(29.6/4)		Dynamic		Rules new from AI12-0071-1.				9	Check that for a subtype conversion to a subtype S with predicates enabled, if a predicate of the parent subtype fails, no predicates directly specified for S are evaluated.
					C324003	Part		7	Check that for a membership test against a subtype S, if a predicate of the parent subtype fails, no predicates directly specified for S are evaluated.
(29.7/4)		Dynamic	Widely Used	Any predicate check will test this.					
(30/3)		Dynamic	Lead-in						
(31/4)	1	Dynamic		Revised by AI12-0071-1. This is really redundant and arguably belongs to 4.6, but it would be weird not to test this here.	C324001, C324002	All	Check that enabled predicate checks are performed on every subtype conversion.		
	2	Redundant		Part of above objective.					
	3	Dynamic			C324002	All	Check that enabled predicate checks are performed upon return for by-reference in out and out parameters.		

C-Test.	
C-Test.	
C-Test.	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3). We'll only test this for subtype conversions and memberships, hopefully the same check code is used for all.	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3).	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3). We'll only test this for subtype conversions, hopefully the same check code is used for all.	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3). We'll only test this for subtype conversions, hopefully the same check code is used for all.	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3). We'll only test this for subtype conversions and memberships, hopefully the same check code is used for all.	
C-Test. This can only be usefully tested for dynamic predicates. Care is needed to avoid triggering 11.4.2(27/3). Jeff's test tries only one example.	

	4	Dynamic	Modified by AI12-0301-1.	**C324001**	All		Check that for an uninitialized object or allocator with any default component expressions, a predicate check is made.	
(31.1/4)	1	Dynamic	Added by AI12-0054-2.	**C324001**	All		Check that a failed predicate check for a type or subtype with a predicate but no Predicate_Failure aspect raises Assertion_Error.	Effectively checked by any “normal” predicate check failure test.
	2	Dynamic		**C324003**	Part	4	Check that a failed predicate check for a type or subtype with a predicate and a Predicate_Failure aspect that raises an exception propagates that exception.	C-Test. Test covers subtypes, ought to try types at some point.
				C324006	All		Check that a failed predicate check for a type or subtype with a predicate and a Predicate_Failure aspect that does not raise an exception propagates Assertion_Error with the string value resulting from the evaluation of the Predicate_Failure aspect.	
				C324003	Part	4	Check that a failed predicate check for a type or subtype with a predicate and a Predicate_Failure aspect that inherits additional predicates from another type or subtype that also has a Predicate_Failure aspect evaluated the Predicate_Failure aspect belonging to the predicate that failed.	C-Test. Test covers subtypes, ought to try types at some point.
	3	Impl-Def	Untestable, the string could be anything.					
(32/4)		Deleted	Deleted by AI12-0071-1.					
(33/4)		Deleted	Deleted by AI12-0071-1.					
(34/3)		NonNormative	A note.					
(35/3)		NonNormative	Another note.					
(36/4)		NonNormative	Another note.					
(37/4)		NonNormative	Another note.					
(38/4)		NonNormative	Another note.					
(39/4)		NonNormative	An example.					
(40/4)		NonNormative	Another example.					
(41/4)		NonNormative	Lead-in	Another example.				
(42/4)		NonNormative		Part of the above example.				
(43/4)		NonNormative		Part of the above example.				
(44/4)		NonNormative		Part of the above example.				
(45/4)		NonNormative		Part of the above example.				
(46/4)		NonNormative		Part of the above example.				
(47/4)		NonNormative		Part of the above example.				
(48/4)		NonNormative		Part of the above example.				
(49/4)		NonNormative		Part of the above example.				
(50/4)		NonNormative		Part of the above example.				
(51/4)		NonNormative		Part of the above example.				

Paragraphs:		Objectives with tests:	Objectives to test:	Total objectives:	Objectives with submitted tests:
1 62		43	54	89	0

Must be tested	Objectives with Priority 10	0
	Objectives with Priority 9	3
Important to test	Objectives with Priority 8	0
	Objectives with Priority 7	11
Valuable to test	Objectives with Priority 6	18
	Objectives with Priority 5	9
Ought to be tested	Objectives with Priority 4	5
	Objectives with Priority 3	8
Worth testing	Objectives with Priority 2	0
Not worth testing	Objectives with Priority 1	0
	Total:	54

Objectives covered by new tests since ACATS 2.6	43

Completely:	35