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

Subclause 6.1.1

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).
6.1.1	(1/4)		StaticSem	Portion	Lead-in for the following paragraphs. Changed by AI12-0045-1.						
	(2/5)	1	StaticSem			**C611001, *C611B01***	All		Check that Pre can be specified for a non-instance subprogram.		
					Added by AI12-0045-1 (in TC1)	***B611001***	Part	7	Check that Pre can be specified for a generic subprogram.	Still need a C-Test, can be included in some other tests.	
						B611001, B611007	Part	7	Check that Pre can be specified for an entry.	Still need a C-Test, can be included in some other tests.	
				Negative	Added by AI12-0045-1 (in TC1)	***B611001***	All		Check that Pre cannot be specified for an instance that is a subprogram.	Was a change from the original Ada 2012 text.	
				Negative		***B611001***	All		Check that Pre cannot be specified for packages, objects, types, single tasks, or single protected objects.		
				Negative	From 13.1.1(18/4), here to ensure it is tested throughly.	**BD11001** (one example), ***B611002***	All		Check that Pre cannot be specified on a subprogram body that is acting as a completion.		
				Negative	From AI12-0194-1; binding interpretation from Ada 2022.			8	Check that Pre cannot be specified on an entry body.	B-Test.	
		2		Widely Used	This is the Ada 95; any Ada 95 subprogram call implicitly tests it.						
	(3/3)	1	StaticSem		“Primitive” is required by 13.1.1(16/3). A generic subprogram can never be primitive. Nor can an instance of a generic subprogram ever be a primitive operation of a tagged type (the occurrence of the instance freezes the tagged type, making instance be too late for freezing). Thus we don't need a separate instance test here.	**C611001, *C611B02, C611A04***	All		Check that Pre'Class can be specified for a non-instance primitive subprogram of a tagged type.		
				Negative		***B611003***	All		Check that Pre'Class cannot be specified for a generic subprogram.		
				Negative	Confirmed by AI12-0182-1	***B611016***	All		Check that Pre'Class cannot be specified for an entry of a tagged task or protected type.		
				Negative		***B611007***	All		Check that Pre'Class cannot be specified for an entry of an untagged task or protected type.		
				Negative	Confirmed by AI12-0182-1	***B611016***	All		Check that Pre'Class cannot be specified for a protected subprogram of a tagged protected type.		
				Negative		***B611007***	All		Check that Pre'Class cannot be specified for a protected subprogram of an untagged protected type.		
				Negative	“Primitive” is required by 13.1.1(16/3); we test this here because we want to ensure that this rule is tested for this aspect; the general rule just tries one example.	**BD11001** (one example), ***B611003*** (ordinary tagged types, interfaces), ***B611016*** (tagged task and protected types)	All		Check that Pre'Class cannot be specified for a subprogram that is not a primitive subprogram of some tagged type.		
				Negative		***B611003, B611016***	All		Check that Pre'Class cannot be specified for packages, objects, types, single tasks, or single protected objects.		
				Negative	From 13.1.1(18/4), here to ensure it is tested throughly.	**BD11001** (one example), ***B611004***	All		Check that Pre'Class cannot be specified on a subprogram body that is acting as a completion.		
				Negative	From AI12-0194-1; binding interpretation from Ada 2022.			8	Check that Pre'Class cannot be specified on an entry body.	B-Test.	

			Negative	Not allowed by 13.1.1(16/3), even in Ada 2022.			5	Check that Pre'Class cannot be specified on an access-to-subprogram type.	B-Test. OK to test for Ada 2012 but more important for Ada 202x.
	2		Widely Used	This is the Ada 95; any Ada 95 dispatching subprogram call implicitly tests it.					
(4/5)	1	StaticSem			**C611001**, ***C611B01***	All		Check that Post can be specified for a non-instance subprogram.	
				Added by AI12-0045-1 (2015 Corrigendum)	***B611001***	Part	7	Check that Post can be specified for a generic subprogram.	Still need a C-Test, can be included in some other tests.
					B611001, B611007	Part	7	Check that Post can be specified for an entry.	Still need a C-Test, can be included in some other tests.
			Negative	Added by AI12-0045-1 (in TC1)	***B611001***	All		Check that Post cannot be specified for an instance that is a subprogram.	Was a change from the original Ada 2012 text.
			Negative		***B611001***	All		Check that Post cannot be specified for packages, objects, types, single tasks, or single protected objects.	
			Negative	From 13.1.1(18/4), here to ensure it is tested throughly.	**BD11001** (one example), ***B611002***	All		Check that Post cannot be specified on a subprogram body that is acting as a completion.	
			Negative	From AI12-0169-1 and AI12-0194-1; Binding Interpretation in Ada 2022.			8	Check that Post cannot be specified on an entry body.	B-Test.
	2		Widely Used	This is the Ada 95; any Ada 95 subprogram call implicitly tests it.					
(5/3)	1	StaticSem		“Primitive” is required by 13.1.1(16/3).	**C611001**, ***C611B02***, ***C611A04***	All		Check that Post'Class can be specified for a non-instance primitive subprogram of a tagged type.	
			Negative	A generic subprogram can never be primitive. Nor can an instance of a generic subprogram ever be a primitive operation of a tagged type (the occurrence of the instance freezes the tagged type, making instance be too late for freezing). Thus we don't need a separate instance test here.	***B611003***	All		Check that Post'Class cannot be specified for a generic subprogram.	
			Negative	Confirmed by pending AI12-0182-1	***B611016***	All		Check that Post'Class can be specified for an entry of a tagged task or protected type.	
			Negative		***B611007***	All		Check that Post'Class cannot be specified for an entry of an untagged task or protected type.	
			Negative	Confirmed by pending AI12-0182-1	***B611016***	All		Check that Post'Class can be specified for a protected subprogram of a tagged protected type.	
			Negative		***B611007***	All		Check that Post'Class cannot be specified for a protected subprogram of an untagged protected type.	
			Negative	“Primitive” is required by 13.1.1(16/3); we test this here because we want to ensure that this rule is tested for this aspect; the general rule just tries one example.	***B611003*** (ordinary tagged types, interfaces), ***B611016*** (tagged task and protected types)	All		Check that Post'Class cannot be specified for a subprogram that is not a primitive subprogram of some tagged type.	
			Negative		***B611003, B611016***	All		Check that Post'Class cannot be specified for packages, objects, types, single tasks, or single protected objects.	
			Negative	From 13.1.1(18/4), here to ensure it is tested throughly.	**BD11001** (one example), ***B611004***	All		Check that Post'Class cannot be specified on a subprogram body that is acting as a completion.	
			Negative	From AI12-0169-1 and AI12-0194-1; Binding Interpretation in Ada 2022.			7	Check that Post'Class cannot be specified on an entry body.	B-Test.
			Negative	Not allowed by 13.1.1(16/3), even in Ada 2022.			5	Check that Post'Class cannot be specified on an access-to-subprogram type.	B-Test. OK to test for Ada 2012.
	2		Widely Used	This is the Ada 95; any Ada 95 dispatching subprogram call implicitly tests it.					

(6/3)	NameRes		The normal legal case will be checked by any C-Test for the aspect.			5	Check that the expression of aspect Pre can have a boolean type other than Boolean.	C-Test, not very common.
						5	Check that the expression of aspect Pre can be resolved if there is exactly one interpretation for a boolean type.	C-Test, just normal resolution.
		Negative				5	Check that the expression of aspect Pre is illegal if there is not exactly one interpretation for a boolean type.	B-Test, just normal resolution.
			The normal legal case will be checked by any C-Test for the aspect.			4	Check that the expression of aspect Pre'Class can have a boolean type other than Boolean.	C-Test, not very common.
						4	Check that the expression of aspect Pre'Class can be resolved if there is exactly one interpretation for a boolean type.	C-Test, just normal resolution.
		Negative				4	Check that the expression of aspect Pre'Class is illegal if there is not exactly one interpretation for a boolean type.	B-Test, just normal resolution.
			The normal legal case will be checked by any C-Test for the aspect.			5	Check that the expression of aspect Post can have a boolean type other than Boolean.	C-Test, not very common.
						5	Check that the expression of aspect Post can be resolved if there is exactly one interpretation for a boolean type.	C-Test, just normal resolution.
		Negative				5	Check that the expression of aspect Post is illegal if there is not exactly one interpretation for a boolean type.	B-Test, just normal resolution.
			The normal legal case will be checked by any C-Test for the aspect.			4	Check that the expression of aspect Post'Class can have a boolean type other than Boolean.	C-Test, not very common.
						4	Check that the expression of aspect Post'Class can be resolved if there is exactly one interpretation for a boolean type.	C-Test, just normal resolution.
		Negative				4	Check that the expression of aspect Post'Class is illegal if there is not exactly one interpretation for a boolean type.	B-Test, just normal resolution.
(7/5)	NameRes		Essentially replaced by AI12-0113-1 (in TC1). The /5 change (from AI12-0170-1) is essentially a ramification.	**C611001** (abstract operation), ***C611A03*** (concrete operation)	All		Check that, for a primitive operation of a type T, that the class-wide precondition expression can make calls to other primitive operations of type T.	
				B611006	Part	6	Check that, for a primitive operation of a type T, that the class-wide precondition expression can make calls to operations with a parameter of T'Class.	C-Test, might come up in some other context. B-Test includes an example, but we still need to execute one.
			Note: we don't need to worry about F'Result in preconditions; it's not legal there.			7	Check that, for a primitive operation of a type T, that the class-wide precondition expression can convert parameters of type T to T'Class to force redispatching. operations of type T.	C-Test, might come up in some other context.
						5	Check that, for a primitive operation of a type T, that the class-wide precondition expression can call subprograms that do not have a parameter of type T or T'Class, and that global objects of types not related to T can be used.	C-Test, not very likely to be wrong.
		Negative	Made illegal by AI12-0113-1 (but always was nonsense).	***B611006***	All		Check that, for a primitive operation of a type T, that the class-wide precondition expression cannot make calls to nonprimitive operations of type T or functions returning T'Class.	
		Negative	T'Class case made illegal by AI12-0113-1 (but always was nonsense).	***B611006***	All		Check that, for a primitive operation of a type T, that the class-wide precondition expression cannot use a global object of type T or T'Class as a parameter to a primitive operation of type T.	
				C611001 (abstract operation), ***C611A03*** (concrete operations)	All		Check that, for a primitive operation of a type T, that the class-wide postcondition expression can make calls to other primitive operations of type T.	
				B611006	Part	6	Check that, for a primitive operation of a type T, that the class-wide postcondition expression can make calls to operations with a parameter of T'Class.	C-Test, might come up in some other context. The B-Test includes a case, but we'd like to run one.

			Negative	Made illegal by AI12-0113-1 (but always was nonsense).	***B611006***	All		Check that, for a primitive function F with a controlling result of type T, that the class-wide postcondition expression can make calls to other primitive operations of type T using F'Result as a parameter.	C-Test, can be included in some other tests.
							8		
							7	Check that, for a primitive function F with a controlling access result of type T, that the class-wide postcondition expression can make calls to other primitive operations of type T using F'Result as a parameter.	C-Test, can be included in some other tests.
							7	Check that, for a primitive operation of a type T, that the class-wide postcondition expression can convert parameters of type T to T'Class to force redispatching. operations of type T.	C-Test, might come up in some other context.
							5	Check that, for a primitive operation of a type T, that the class-wide postcondition expression can call subprograms that do not have a parameter of type T or T'Class, and that global objects of types not related to T can be used.	C-Test, not very likely to be wrong.
			Negative	Made illegal by AI12-0113-1 (but always was nonsense).	***B611006***	All		Check that, for a primitive operation of a type T, that the class-wide postcondition expression cannot make calls to nonprimitive operations of type T or functions of T'Class.	
			Negative	T'Class case made illegal by AI12-0113-1 (but always was nonsense).	***B611006***	All		Check that, for a primitive operation of a type T, that the class-wide postcondition expression cannot use a global object of type T or T'Class as a parameter to a primitive operation of type T.	
(8/3)		NameRes		The “shall resolve to” case.			7	Check that in a qualified expression used in a postcondition expression, an overloaded prefix of 'Old can be resolved if the prefix alone could be resolved.	C-Test.
				The “expected type” case.			7	Check that in an actual parameter expression used in a postcondition expression, an overloaded prefix of 'Old can be resolved if the prefix alone could be resolved.	C-Test. There are other cases that we could try, but that's probably overkill.
				The “otherwise” case.			7	Check that in a type conversion used in a postcondition expression, an overloaded prefix of 'Old cannot be resolved, even if only one interpretation would be legal.	B-Test.
(9/3)	1	Legality			***B611005***	All		Check that a Pre aspect cannot be specified on an abstract subprogram.	
					B611005	All		Check that a Pre aspect cannot be specified on a null procedure.	
					B611005	All		Check that a Post aspect cannot be specified on an abstract subprogram.	
					B611005	All		Check that a Post aspect cannot be specified on a null procedure.	
	2		Redundant	(The same objectives could have been tested as “Negative” above)	**C611001**	All		Check that a Pre'Class aspect can be specified on an abstract subprogram.	
							7	Check that a Pre'Class aspect can be specified on a null procedure.	C-Test, can be included in some other tests.
					C611001	All		Check that a Post'Class aspect can be specified on an abstract subprogram.	
							7	Check that a Post'Class aspect can be specified on a null procedure.	C-Test, can be included in some other tests.
(10/3)		Legality	Portion	Tested under paragraphs 15 and 16 below.					
(11/3)		Legality	Portion	Tested under paragraphs 15 and 16 below.					
(12/3)		Legality	Portion	Tested under paragraphs 15 and 16 below.					
(13/3)		Legality	Portion	Tested under paragraphs 15 and 16 below.					

(14/3)		Legality	Portion	Just a connecting word.					
(15/3)		Legality			***B611018***	All		Check that for an abstract type T that inherits homographs of a subprogram S from two different ancestors with non-conforming preconditions, the inherited S cannot be called by a statically bound call.	
(16/3)		Legality			***B611018***	All		Check that for a nonabstract type T that inherits homographs of a subprogram S from two different ancestors with non-conforming preconditions, the inherited S is illegal if it is not overridden.	
							6	Check that for a nonabstract type T that inherits homographs of a subprogram S from two different ancestors with non-conforming preconditions, an overriding of the inherited S is allowed.	C-Test. Try just calling the parent routine.
(17/3)		Legality					6	Check that a renaming S1 that overrides an inherited routine S2 is legal if all of the class-wide preconditions fully conform.	C-Test.
					B611019	All		Check that a renaming S1 that overrides an inherited routine S2 is illegal if any of the class-wide preconditions do not fully conform.	
(17.1/4)		Legality		Rule added by AI12-0131-1.	***B611017***	All		Check that Pre'Class cannot be specified for an overriding of a subprogram that does not specify Pre'Class.	
(17.2/4)		Legality		Generic boilerplate.	***B611017***	All		Check that an instance is illegal if the instance contains a subprogram that specifies Pre'Class and overrides a primitive operation of a formal type that does not specify Pre'Class.	
					B611019	All		Check that an instance is illegal if the instance contains a renaming that overrides a primitive operation of a formal type where all of the class-wide preconditions do not fully conform.	
							6	Check that an instance is illegal if it contains for a nonabstract type T that inherits homographs of a subprogram S from two different ancestors (at least one of which is a actual parameter of the instance) with non-conforming preconditions, and the inherited S is not overridden.	B-Test. This seems pretty unlikely in a generic unit.
(18/5)	1	StaticSem		Modified by AI12-0113-1 and AI12-0131-1 from Corrigendum 1, and AI12-0170-1 (a ramification) later.	**C611001** (parent, interface)	Part	6	Check that a class-wide precondition is inherited by a subprogram inherited from an ancestor that has a Pre'Class aspect specified.	C-Test. Should check inheritance subprograms inherited from (interface) progenitors, and from various kinds of types (private, tagged record). May occur as part of other tests.
					C611001 (parent, interface)	Part	6	Check that a class-wide postcondition is inherited by a subprogram inherited from an ancestor that has a Post'Class aspect specified.	C-Test. Should check inheritance subprograms inherited from (interface) progenitors, and from various kinds of types (private, tagged record). May occur as part of other tests.
				Rule added by AI12-0131-1 (in Technical Corrigendum 1 for Ada 2012 [TC1]). The Post'Class part is untestable; anding True has no effect.			9	Check that a class-wide precondition is inherited as True for a subprogram inherited from an ancestor that does not specify Pre'Class.	C-Test. Most cases are undetectable, or illegal by 6.1.1(17.2/4). But the case where an overriding routine does not have Pre'Class, is inherited from two homographs, one with Pre'Class and one without, should end up with a precondition of True, not the inherited Pre'Class.
	2	StaticSem	Portion	Lead-in for the following paragraphs. Added by AI12-0113-1.					

(18.1/4)	StaticSem		Any inherited Pre'Class or Post'Class will implicitly test the basic rule, thus we only test unusual cases. Added by AI12-0113-1.	7	Check that an inherited Pre'Class works properly if the parameter names of an overriding subprogram are different from the ancestor subprogram.	C-Test.
				7	Check that an inherited Post'Class works properly if the parameter names of an overriding subprogram are different from the ancestor subprogram.	C-Test.
				7	Check that an inherited Pre'Class works properly if the original Pre'Class refers to the name of the ancestor subprogram.	C-Test. Probably have to use a recursive call (ugh).
				7	Check that an inherited Post'Class works properly if the original Post'Class refers to the name of the ancestor subprogram.	C-Test. One way to do this is to use F'Result.
(18.2/4)	StaticSem		Added by AI12-0113-1, replaced by AI12-0412-1. The AI12-0113-1 rule was originally tested in B611006 as a third objective; that has been removed.	8	Check that a concrete primitive subprogram is treated as if it is abstract if an inherited Pre'Class or Post'Class could call an abstract subprogram.	B-Test. Try all of a nondispatching call, prefix of 'Access, and generic actual subprogram. See the AI for example cases.
				6	Check that a concrete primitive subprogram can be defined for an abstract type even if Pre'Class or Post'Class could call an abstract subprogram.	C-Test. Try calls on inherited descendants of the subprogram. This is relatively important as a change from Ada 2012 Cor1.
(19/3)	Definition		Defines "enabled". Any test of preconditions or postconditions implicitly tests the basic definition. We check some of the corner cases.	6	Check that a specific precondition expression is not evaluated if it is not enabled.	C-Test.
				6	Check that a class-wide precondition expression is not evaluated if it is not enabled.	C-Test.
				6	Check that a specific postcondition expression is not evaluated if it is not enabled.	C-Test.
				6	Check that a class-wide postcondition expression is not evaluated if it is not enabled.	C-Test.
				7	Check that a specific precondition is evaluated if it is enabled, even if specific preconditions are Ignored at the site of the call.	C-Test. Try overall and individual Assertion_Policies
				7	Check that a class-wide precondition is evaluated if it is enabled, even if preconditions are Ignored at the site of the call.	C-Test. Try overall and individual Assertion_Policies
				7	Check that a specific postcondition is evaluated if it is enabled, even if specific preconditions are Ignored at the site of the call.	C-Test. Try overall and individual Assertion_Policies
				7	Check that a class-wide postcondition is evaluated if it is enabled, even if preconditions are Ignored at the site of the call.	C-Test. Try overall and individual Assertion_Policies
				7	Check that a class-wide precondition expression is evaluated if it is enabled, even if it is inherited by a an overriding subprogram for which the applicable Assertion_Policy is Ignore.	C-Test. From AARM 6.1.1(19.a/3).
				7	Check that a class-wide postcondition expression is evaluated if it is enabled, even if it is inherited by a an overriding subprogram for which the applicable Assertion_Policy is Ignore.	C-Test. From AARM 6.1.1(19.a/3).
(20/3)	Definition	Portion	Defines "potentially unevaluated"; this is a lead-in.			

(21/3)	Definition		We'll make the tests for 6.1.1(27/5) here, as there are a number of cases and they are much easier to enumerate here.	***B611014***	All		Check that an Old attribute reference is illegal if the prefix does not statically denote an object, and the use of Old appears in any part of an if expression other than the first condition.	
(21.1/4)	Definition		We'll make the tests for 6.1.1(27/5) here, as there are a number of cases and they are much easier to enumerate here. Originally added by AI12-0032-1 (in TC1), moved by AI12-0280-2.	***B611014***	All		Check that an Old attribute reference is illegal if the prefix does not statically name an object, and the use of 'Old appears in the predicate of a quantified expression.	
(21.2/5)	Definition		Originally added by AI12-0198-1, an Ada 2022 Binding Interpretation.			8	Check that an Old attribute reference is illegal if the prefix does not statically name an object, and the use of 'Old appears in the expression of an array component association which is null or nonstatic.	B-Test. Make sure to try some legal cases where the prefix statically denotes an object, marked with OK. Try all of subtypes, ranges, and others. There is a tiny example in the question of AI12-0198-1.
(22/3)	Definition			***B611014***	All		Check that an Old attribute reference is illegal if the prefix does not statically denote an object, and the use of Old appears as the dependent expression of a case expression.	
(23/5)	Definition			***B611014***	All		Check that an Old attribute reference is illegal if the prefix does not statically denote an object, and the use of Old appears as the right operand of a short circuit control form.	
(24/3)	Definition			***B611014***	All		Check that an Old attribute reference is illegal if the prefix does not statically denote an object, and the use of Old appears as a membership choice other than the first in a membership operation.	
(25/3)	StaticSem	Portion	Lead-in for the following paragraphs.					
		Negative		***B611015***	All		Check that the prefix of an Old attribute cannot have a limited type.	
(26/4)	StaticSem	Subpart	The effect of location of these implicit constants is fleshed out in 6.1.1(35.1/4); finalization test objectives are there. Modified by AI12-0032-1 and AI12-0280-2.			10	For X'Old given in the postcondition for a subprogram S, check that X is evaluated at the start of the subprogram body for S.	C-Test. Use a prefix with a function call (that uses TcTouch), and ensure that the function is called before any local variables are created. Try in Post and Post'Class (including inherited Post'Class).
						10	For X'Old given in the postcondition for a subprogram S, check that X'Old has the value of X at the start of the subprogram body for S.	C-Test. Try to combine with the above. Try a number of different kinds of types and prefixes (function calls, array indexing, dereferences).
						9	For X'Old given in the postcondition for a task entry E, check that X is evaluated at the start of the accept statement for E.	C-Test. Use a prefix with a function call (that uses TcTouch), and ensure that the function is called before any local variables are created. Try in Post and Post'Class (including inherited Post'Class).
						9	For X'Old given in the postcondition for a task entry E, check that X'Old has the value of X at the start of the accept statement for E.	C-Test. Try to combine with the above. Try a number of different kinds of types and prefixes (function calls, array indexing, dereferences).
						9	For X'Old given in the postcondition for a protected entry E, check that X is evaluated at the start of the entry body for E.	C-Test. Use a prefix with a function call (that uses TcTouch), and ensure that the function is called before any local variables are created. Try in Post and Post'Class (including inherited Post'Class).

(26.1/4)	StaticSem	Portion	Added by AI12-0032-1 (in TC1); lead-in for following paragraphs.
(26.2/4)	StaticSem	Portion	Added by AI12-0032-1 (in TC1); lead-in for following paragraph.
(26.3/4)	StaticSem		Added by AI12-0032-1 (in TC1). This mostly avoids semantic anomalies, not much that is testable. There are probably other cases that could be tested, but it's hard to get interested in dynamic accessibility!
(26.4/4)	StaticSem	Portion	Added by AI12-0032-1 (in TC1); lead-in for following paragraph.

9	For X'Old given in the postcondition for a protected entry E, check that X'Old has the value of X at the start of the entry body for E.	C-Test. Try to combine with the above. Try a number of different kinds of types and prefixes (function calls, array indexing, dereferences).
9	For X'Old given in the postcondition for a protected subprogram S, check that X'Old has the value of X at the start of the subprogram body for S.	C-Test. Try to combine with the above. Try a number of different kinds of types and prefixes (function calls, array indexing, dereferences).
10	For X'Old given in the postcondition for a subprogram S, check that when X is controlled, X'Old is a copy of X initialized at the start of the subprogram body for S.	C-Test. Use a non-limited controlled type, and ensure that Adjust is called appropriately before any local variables are created. (Probably combine with some of the tests for 6.1.1.(35.1/4), which check finalization). Try in Post and Post'Class (including inherited Post'Class).
9	For X'Old given in the postcondition for a task entry E, check that when X is controlled, X'Old is a copy of X initialized at the start of the accept statement for E.	C-Test. Use a non-limited controlled type, and ensure that Adjust is called appropriately before any local variables are created. (Probably combine with some of the tests for 6.1.1.(35.1/4), which check finalization). Try in Post and Post'Class (including inherited Post'Class using interfaces).
9	For X'Old given in the postcondition for a protected entry E, check that when X is controlled, X'Old is a copy of X initialized at the start of the entry body for E.	C-Test. Use a non-limited controlled type, and ensure that Adjust is called appropriately before any local variables are created. (Probably combine with some of the tests for 6.1.1.(35.1/4), which check finalization). Try in Post and Post'Class (including inherited Post'Class using interfaces).
9	For X'Old given in the postcondition for a protected subprogram S, check that when X is controlled, X'Old is a copy of X initialized at the start of the subprogram body for S.	C-Test. Use a non-limited controlled type, and ensure that Adjust is called appropriately before any local variables are created. (Probably combine with some of the tests for 6.1.1.(35.1/4), which check finalization). Try in Post and Post'Class (including inherited Post'Class using interfaces).
6	For X'Old given in the postcondition for a subprogram S, check that X'Old has the accessibility of X when X is an access parameter of S.	C-Test. We can use the accessibility membership to test this, although we'll need to compare both names to named types of the appropriate level.
6	For X'Old given in the postcondition for a subprogram S, check that X'Old has the accessibility of X when X is an access discriminant of a parameter of S.	C-Test. We can use the accessibility membership to test this, although we'll need to compare both names to named types of the appropriate level.
6	For X'Old given in the postcondition for a subprogram S, check that X'Old has the accessibility of X when X is an anonymous access component of a parameter of S.	C-Test. We can use the accessibility membership to test this, although we'll need to compare both names to named types of the appropriate level.

(26.5/4)		StaticSem		Objective matches AARM 6.1.1.(26.a/1). The messy declaration's only semantic effect is on the tag. Added by AI12-0032-1.	***C611B01*** (Post), ***C611B02*** (Post'Class)	All		For X'Old given in the postcondition for a subprogram S, check that X'Old has the same tag as X when X is a parameter P of S, even if the tag of X is different than the nominal subtype of P.
(26.6/4)		StaticSem	Portion	Added by AI12-0032-1 (in TC1); part of the preceding paragraph.				
(26.7/4)		StaticSem	Portion	Added by AI12-0032-1 (in TC1); lead-in for following paragraph.				
(26.8/4)		StaticSem		Added by AI12-0032-1 (in TC1). This mostly allows us to understand corner cases. All of the interesting objectives are elsewhere in this subclause.				
(26.9/4)		StaticSem	Portion	Added by AI12-0032-1 (in TC1); part of the preceding paragraph.				
(26.10/5)	1	StaticSem		Added by AI12-0032-1 (in TC1).	***B611013***	All		For a discrete X, check that the nominal subtype of X'Old is that of X.
	2	NameRes		Added by AI12-0032-1 (in TC1), removed by AI12-0185-1 as it duplicates 6..1.1(8/3) The objectives are under that paragraph.				
	3	NameRes		Added by AI12-0032-1 (in TC1), removed by AI12-0185-1 as it duplicates 6..1.1(8/3) The objectives are under that paragraph.				
(27/5)	1	Legality	Widely Used	Added by AI12-0032-1 (in TC1). Any use of Old in a postcondition will test.				
			Negative	We only try cases associated with a call of some sort; it's hard to imagine what it would mean in any other case (in a package body, for instance).	***B611010***	All		Check that the Old attribute cannot be used inside a subprogram or entry body, or within an accept statement.
					B611011	All		Check that the Old attribute cannot be used inside a precondition expression.
					B611011	All		Check that the Old attribute cannot be used inside of the specification of a generic unit, other than in postconditions.
	2	Legality			***B611012***	All		Check that the prefix of an Old attribute cannot contain a Result attribute.
					B611012	All		Check that the prefix of an Old attribute cannot contain another Old attribute.
					B611012	All		Check that the prefix of an Old attribute cannot contain the loop parameter of an enclosing quantified expression.
	3	Legality	Subpart	The objectives for this are under paragraphs 21-24 above. That's backwards from the usual layout, but it makes it a lot easier to see that all of the cases are tested.				
(28/3)		StaticSem	Widely Used	Any use of Result will test.				
			Negative		***B611008***	All		Check that the prefix of a Result attribute cannot be a procedure or entry.
			Negative		***B611008***	All		Check that the prefix of a Result attribute cannot be an object.
			Negative		***B611008***	All		Check that the prefix of a Result attribute cannot be a type, package, task, or protected type.
(29/5)	1	StaticSem					9	Check that the F'Result attribute denotes the result of the function F within a specific postcondition for F.	C-Test. (Might be combined with another test?)

							9	Check that the F'Result attribute denotes the result of the function F within a class-wide postcondition for F.	C-Test. (Might be combined with another test?)
	2	NameRes	Widely Used	Any use of Result will test. We could try fancy resolution tests, but those would be of low value.					
	3			Removed by AI12-0185-1, as the wording conflicted with 6.1.1(7/4).					
	4			Removed by AI12-0185-1, as the wording conflicted with 6.1.1(7/4).					
(30/3)		Legality			***B611008***	All		Check that F'Result is not allowed in the postcondition expression for some other function.	
					B611009	All		Check that F'Result is not allowed in the body of F, including in a pragma Assert.	
					B611008	All		Check that F'Result is not allowed in a precondition expression for F.	
(31/3)		Dynamic	Portion	Lead-in for the following paragraphs.					
(32/3)		Dynamic			***C611A02***	All		Check that an enabled specific precondition of a subprogram S is evaluated after evaluating the parameters of a call on S and before S is called, and that Assertion_Error is raised if the expression evaluates to False.	
							9	Check that an enabled specific precondition of a task entry E is evaluated after evaluating the parameters of a call on E and before E is called, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.
							9	Check that an enabled specific precondition of a protected entry E is evaluated after evaluating the parameters of a call on E and before E is called, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.
							9	Check that an enabled specific precondition of a protected subprogram S is evaluated after evaluating the parameters of a call on S and before S is called, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.
							9	Check that a specific precondition of a subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test.
							8	Check that a specific precondition of a task entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test.
							8	Check that a specific precondition of a protected entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test.
							8	Check that a specific precondition of a protected subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test.
					C611A01	All		Check that an enabled specific precondition of a subprogram S raises Assertion_Error if it evaluates to False, even if a class-wide precondition for S evaluated to True.	We could have checked a case with two Pre'Class exprs and a Pre, but it doesn't seem worth the extra level of declarations (Jeff's test did that, but it was very unrealistic).
(33/3)		Dynamic			***C611A03*** (normal tagged), ***C611A04*** (interface)	All		Check that an enabled class-wide precondition of a subprogram S is evaluated after evaluating the parameters of a call on S and before S is called, and that Assertion_Error is raised if all such expressions evaluate to False.	

								Check that an enabled class-wide precondition of a task entry E is evaluated after evaluating the parameters of a call on E and before E is called, and that Assertion_Error is raised if all such expressions evaluate to False.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that an enabled class-wide precondition of a protected entry E is evaluated after evaluating the parameters of a call on E and before E is called, and that Assertion_Error is raised if all such expressions evaluate to False.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that an enabled class-wide precondition of a protected subprogram S is evaluated after evaluating the parameters of a call on S and before S is called, and that Assertion_Error is raised if all such expressions evaluate to False.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
						C611A03 (normal tagged), ***C611A04*** (interface)	All	Check that if a subprogram S has multiple applicable class-wide preconditions, that all such expressions evaluate to False before Assertion_Error is raised.	
								Check that if a task entry E has multiple applicable class-wide preconditions, that all such expressions evaluate to False before Assertion_Error is raised.	C-Test. Pre'Class expressions have to be inherited from multiple interfaces, thus the relatively low priority.
								Check that if a protected entry E has multiple applicable class-wide preconditions, that all such expressions evaluate to False before Assertion_Error is raised.	C-Test. Pre'Class expressions have to be inherited from multiple interfaces, thus the relatively low priority.
								Check that if a protected subprogram S has multiple applicable class-wide preconditions, that all such expressions evaluate to False before Assertion_Error is raised.	C-Test. Pre'Class expressions have to be inherited from multiple interfaces, thus the relatively low priority.
								Check that a class-wide precondition of a subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that a class-wide precondition of a task entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that a class-wide precondition of a protected entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that a class-wide precondition of a protected subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Pre'Class. Careful: only one class-wide precondition needs to be evaluated if it is True.
								Check that if all applicable class-wide preconditions evaluated to False, Assertion_Error is raised even if an enabled specific precondition of S evaluates to True.	We could have checked a case with two Pre'Class exprs and a Pre, but it doesn't seem worth the extra level of declarations (Jeff's test did that, but it was very unrealistic).
						C611A01	All		
(34/5)	1	Dynamic	Not Testable	This is non-determinism in the evaluation, which is not testable (but needs to be taken into account in other tests).					
	2	Dynamic	Not Testable						

	3	Dynamic	AI12-0166-1 (not in TC1) makes this deterministic and thus testable. Although I don't know of any easily testable effects of a protected action. Anyway, no test until 2018 at the earliest.					
	4	Dynamic	AI12-0166-1 (not in TC1) originally deleted this sentence, but clearly the task objective is not covered by the previous sentence so that was a mistake.					
(35/3)	1	Dynamic				7	Check that a task entry E evaluates its preconditions before checking that the entry is open; in particular, a precondition check can fail immediately even for a closed entry.	C-Test. Possibly combine with one of the earlier tests.
						7	Check that a protected entry E evaluates its preconditions before checking that the entry is open; in particular, a precondition check can fail immediately even for a closed entry.	C-Test. Possibly combine with one of the earlier tests.
						10	Check that no postcondition check is performed for a subprogram S if S propagates an exception.	C-Test. Try postconditions that would fail if evaluated. Try Post, Post'Class, and inherited Post'Class.
						7	Check that no postcondition check is performed for a task entry E if E propagates an exception.	C-Test. Try postconditions that would fail if evaluated. Try Post, Post'Class, and inherited Post'Class.
						7	Check that no postcondition check is performed for a protected entry E if E propagates an exception.	C-Test. Try postconditions that would fail if evaluated. Try Post, Post'Class, and inherited Post'Class.
						7	Check that no postcondition check is performed for a protected subprogram S if S propagates an exception.	C-Test. Try postconditions that would fail if evaluated. Try Post, Post'Class, and inherited Post'Class.
						9	Check that by-copy in-out and out parameters are not modified if a postcondition check fails for a subprogram call.	C-Test. Try a variety of by-copy parameter types. Try Post, Post'Class, and inherited Post'Class.
						7	Check that by-copy in-out and out parameters are not modified if a postcondition check fails for a task entry call.	C-Test. Try a variety of by-copy parameter types. Try Post, Post'Class, and inherited Post'Class.
						7	Check that by-copy in-out and out parameters are not modified if a postcondition check fails for a protected entry call.	C-Test. Try a variety of by-copy parameter types. Try Post, Post'Class, and inherited Post'Class.
						7	Check that by-copy in-out and out parameters are not modified if a postcondition check fails for a protected subprogram call.	C-Test. Try a variety of by-copy parameter types. Try Post, Post'Class, and inherited Post'Class.
	2, 3	Dynamic	Note that we can't determine precisely when copy-back of parameters occurs (they can't be controlled), so we can only test that the evaluation happens between the end of the body and the continuation of execution.	***C611A02***	All		Check that an enabled specific postcondition of a subprogram S is evaluated after completing the subprogram body but before continuing execution after the call of S, and that Assertion_Error is raised if the expression evaluates to False.	
						8	Check that an enabled specific postcondition of a task entry E is evaluated after completing the subprogram body but before continuing execution after the call of E, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.
						8	Check that an enabled specific postcondition of a protected entry E is evaluated after completing the subprogram body but before continuing execution after the call of E, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.

			Check that an enabled specific postcondition of a protected subprogram S is evaluated after completing the subprogram body but before continuing execution after the call of S, and that Assertion_Error is raised if the expression evaluates to False.	C-Test. A TCTouchy test.
C611A03 (normal tagged), ***C611A04*** (interface)	All		Check that an enabled class-wide postcondition of a subprogram S is evaluated after completing the subprogram body but before continuing execution after the call of S, and that Assertion_Error is raised if any such expression evaluates to False.	
		7	Check that an enabled class-wide postcondition of a task entry E is evaluated after completing the subprogram body but before continuing execution after the call of E, and that Assertion_Error is raised if any such expression evaluates to False.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
		7	Check that an enabled class-wide postcondition of a protected entry E is evaluated after completing the subprogram body but before continuing execution after the call of E, and that Assertion_Error is raised if any such expression evaluates to False.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
		7	Check that an enabled class-wide postcondition of a protected subprogram S is evaluated after completing the subprogram body but before continuing execution after the call of S, and that Assertion_Error is raised if any such expression evaluates to False.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
C611A03 (normal tagged), ***C611A04*** (interface)	All		Check that if multiple enabled class-wide postconditions apply to a subprogram S, check that they are all evaluated if they all evaluate to True.	
		7	Check that if multiple enabled class-wide postconditions apply to a task entry E, check that they are all evaluated if they all evaluate to True.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Try both adding a Post'Class to an overriding routine, or inheriting Post'Class from 1 or more interfaces.
		7	Check that if multiple enabled class-wide postconditions apply to a protected entry E, check that they are all evaluated if they all evaluate to True.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Try both adding a Post'Class to an overriding routine, or inheriting Post'Class from 1 or more interfaces.
		7	Check that if multiple enabled class-wide postconditions apply to a protected subprogram S, check that they are all evaluated if they all evaluate to True.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Try both adding a Post'Class to an overriding routine, or inheriting Post'Class from 1 or more interfaces.
		8	Check that a specific postcondition of a subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test.
		7	Check that a specific postcondition of a task entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class.
		7	Check that a specific postcondition of a protected entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Post'Class.
		7	Check that a specific postcondition of a protected subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Post'Class.

							Check that a class-wide postcondition of a subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test. Careful: only one class-wide postcondition needs to be evaluated if it is False.
						7	Check that a class-wide postcondition of a task entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the task type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
						7	Check that a class-wide postcondition of a protected entry E that is not enabled is not evaluated during a call on E.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
						7	Check that a class-wide postcondition of a protected subprogram S that is not enabled is not evaluated during a call on S.	C-Test. A TCTouchy test. This can happen if the protected type has an interface with Post'Class. Careful: only one class-wide postcondition needs to be evaluated if it is False.
				C611A01	All		Check that if any applicable class-wide postcondition evaluates to False, Assertion_Error is raised even if an enabled specific postcondition of S evaluates to True.	We could have checked a case with two Post'Class exprs and a Post, but it doesn't seem worth the extra declarations.
				C611A01	All		Check that if an enabled specific postcondition valuates to False, Assertion_Error is raised even if all enabled applicable class-wide postconditions of S evaluate to True.	We could have checked a case with two Post'Class exprs and a Post, but it doesn't seem worth the extra declarations.
	4	Dynamic	Not Testable	This is non-determinism in the evaluation, which is not testable (but needs to be taken into account in other tests).				
	5	Dynamic	Not Testable	This is non-determinism in the evaluation, which is not testable (but needs to be taken into account in other tests).				
(35.1/4)	1	Dynamic		Added by AI12-0032-1. AI12-0193-1 confirms that the objective is the intent. The protected action part is untestable; the only effect of not doing this is to introduce race conditions – which are not testable.		8	Check that if a postcondition check fails for a task entry E, Assertion_Error is raised in both the accept_statement and the entry call for E.	C-Test.
	2					10	For X'Old given in the postcondition for a subprogram S, check that when X is controlled, X'Old is finalized last, after any local objects that need finalization and after the postcondition check, for a subprogram call of S.	C-Test; combine with the initialization tests for 6.1.1(26/4).
						9	For X'Old given in the postcondition for a task entry E, check that when X is controlled, X'Old is finalized last, after any local objects that need finalization and after the postcondition check, for an entry call of E.	C-Test; combine with the initialization tests for 6.1.1(26/4).
						9	For X'Old given in the postcondition for a protected entry E, check that when X is controlled, X'Old is finalized last, after any local objects that need finalization and after the postcondition check, for an entry call of S.	C-Test; combine with the initialization tests for 6.1.1(26/4).
						9	For X'Old given in the postcondition for a protected subprogram S, check that when X is controlled, X'Old is finalized last, after any local objects that need finalization and after the postcondition check, for a subprogram call of S.	C-Test; combine with the initialization tests for 6.1.1(26/4).

(36/3)

9	Check that the exception raised by the evaluation or failure of a specific precondition check for a subprogram cannot be handled inside of the subprogram body.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific precondition check for a task entry E cannot be handled inside of the accept statement for E.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific precondition check for a protected entry cannot be handled inside of the entry body.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific precondition check for a protected subprogram cannot be handled inside of the subprogram body.	C-Test.
9	Check that the exception raised by the evaluation or failure of a class-wide precondition check for a subprogram cannot be handled inside of the subprogram body.	C-Test.
6	Check that the exception raised by the evaluation or failure of a class-wide precondition check for a task entry E cannot be handled inside of the accept statement for E.	C-Test. The task must have an interface.
6	Check that the exception raised by the evaluation or failure of a class-wide precondition check for a protected entry cannot be handled inside of the entry body.	C-Test. The protected type must have an interface.
6	Check that the exception raised by the evaluation or failure of a class-wide precondition check for a protected subprogram cannot be handled inside of the subprogram body.	C-Test. The protected type must have an interface.
9	Check that the exception raised by the evaluation or failure of a specific postcondition check for a subprogram cannot be handled inside of the subprogram body.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific postcondition check for a task entry E cannot be handled inside of the accept statement for E.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific postcondition check for a protected entry cannot be handled inside of the entry body.	C-Test.
7	Check that the exception raised by the evaluation or failure of a specific postcondition check for a protected subprogram cannot be handled inside of the subprogram body.	C-Test.
9	Check that the exception raised by the evaluation or failure of a class-wide postcondition check for a subprogram cannot be handled inside of the subprogram body.	C-Test.
6	Check that the exception raised by the evaluation or failure of a class-wide postcondition check for a task entry E cannot be handled inside of the accept statement for E.	C-Test. The task must have an interface.
6	Check that the exception raised by the evaluation or failure of a class-wide postcondition check for a protected entry cannot be handled inside of the entry body.	C-Test. The protected type must have an interface.
6	Check that the exception raised by the evaluation or failure of a class-wide postcondition check for a protected subprogram cannot be handled inside of the subprogram body.	C-Test. The protected type must have an interface.

(37/4)	1	Dynamic	Widely Used	For normal subprogram calls, the expressions evaluated are obvious and tested any time the aspects are used. We don't have to implementation inheritance for task and protected operations, as only interfaces can be inherited for them.	***C611A02***	All		For a dispatching call, check that the specific precondition evaluated is that of the actual body invoked.	
					C611A02	All		For a dispatching call, check that the specific postcondition evaluated is that of the actual body invoked.	
					C611A03	All		For a dispatching call, check that the class-wide postcondition evaluated is that of the actual body invoked.	
					C611A02	All		For a call on a subprogram S whose implementation is inherited from the primitive subprogram A of an ancestor, check that the specific precondition that applies to A is checked for a call on S.	
					C611A02	All		For a call on a subprogram S whose implementation is inherited from the primitive subprogram A of an ancestor, check that the specific postcondition that applies to A is checked for a call on S.	
	2	Dynamic		We'll test the unusual case (the normal case should be previously tested). Note that this case can't happen for task or protected entries or subprograms – implementations can't be inherited.			9	For a subprogram S that is inherited from an ancestor type A of a type T, check that class-wide postconditions inherited from a homograph of S that is primitive for an interface that is a progenitor of T but not of A are checked.	C-Test. We're trying to check that a wrapper is used in this (unusual) case; the original body must NOT check the added Post'Class.
	3, 4	Dynamic		Added by AI12-0113-1 (in TC1).	***C611A03***	All		For a nonabstract tagged type T and a primitive subprogram S of T and that has a class-wide postcondition expression E, check that for a call of S that is statically bound to type T, calls to primitive operations of T within E invoke the bodies appropriate for T, even if the tag of the controlling parameter object is not T.	
					C611A04	Part	6	For an interface type T and a primitive subprogram S of T and that has a class-wide postcondition expression E, check that for a call of S that is statically bound to a nonbastract type NT derived from T, calls to primitive operations of T within E invoke the bodies appropriate for NT, even if the tag of the controlling parameter object is not NT.	C-Test. The tag of the controlling parameter should identify some descendant of T that has overriding bodies for the subprograms mentioned in the postcondition. Still should try task and protected interfaces here.
					C611A03, C611B02	All		For a nonabstract tagged type T and a primitive subprogram S of T and that has a class-wide postcondition expression E, check that for a dynamically tagged dispatching call of S, calls to primitive operations of T within E invoke the bodies appropriate for the controlling tag, even if it is not T.	
					C611A04	Part	6	For an interface type T and a primitive subprogram S of T and that has a class-wide postcondition expression E, check that for a dynamically tagged dispatching call of S, calls to primitive operations of T within E invoke the bodies appropriate for the controlling tag, even if it is not T.	C-Test. The tag of the controlling parameter should identify some descendant of T that has overriding bodies for the subprograms mentioned in the postcondition. Still should try task and protected interfaces here.
(38/4)	1	Dynamic		We treat statically bound calls as “widely used” for this objective; it's hard to imagine what else they would do, and almost any test of class-wide preconditions will try them.	***C611A03*** (normal tagged), ***C611A04*** (interface)	All		Check that the class-wide precondition of a dispatching call is that associated with the denoted subprogram, even if the body of a descendant operation is invoked.	

	2,3	Dynamic	Added by AI12-0113-1 (in TC1).	***C611A03***	All		For a nonabstract tagged type T and a primitive subprogram S of T and that has a class-wide precondition expression E, check that for a call of S that is statically bound to type T, calls to primitive operations of T within E invoke the bodies appropriate for T, even if the tag of the controlling parameter object is not T.	
				C611A04	Part	6	For an interface type T and a primitive subprogram S of T and that has a class-wide precondition expression E, check that for a call of S that is statically bound to a nonabstract type NT derived from T, calls to primitive operations of T within E invoke the bodies appropriate for NT, even if the tag of the controlling parameter object is not NT.	C-Test. The tag of the controlling parameter should identify some descendant of T that has overriding bodies for the subprograms mentioned in the postcondition. Still should try task and protected interfaces here.
				C611A03	All		For a nonabstract tagged type T and a primitive subprogram S of T and that has a class-wide precondition expression E, check that for a dynamically tagged dispatching call of S, calls to primitive operations of T within E invoke the bodies appropriate for the controlling tag, even if it is not T.	
				C611A04	Part	6	For an interface type T and a primitive subprogram S of T and that has a class-wide precondition expression E, check that for a dynamically tagged dispatching call of S, calls to primitive operations of T within E invoke the bodies appropriate for the controlling tag, even if it is not T.	C-Test. The tag of the controlling parameter should identify some descendant of T that has overriding bodies for the subprograms mentioned in the postcondition. Still should try task and protected interfaces here.
(38.1/5)			Added by AI12-0195-1, a Binding Interpretation in Ada 2022.			7	For an inherited subprogram S of a tagged type T, check that both versions of the class-wide precondition expression E are checked if any primitive subprograms used by E are overridden for T.	C-Test. The test will need to arrange that the original E returns False while the new E returns True.
						6	For an inherited subprogram S of a tagged type T, check that both versions of the class-wide postcondition expression E are checked if any primitive subprograms used by E are overridden for T.	C-Test. The test will need to arrange that the original E returns False while the new E returns True.
(39/3)		Dynamic				9	For a call via an access-to-subprogram value created with S'Access, check that the specific precondition of S is checked if it is enabled.	C-Test. Try different subprograms called via a single access type.
						9	For a call via an access-to-subprogram value created with S'Access, check that the specific postcondition of S is checked if it is enabled.	C-Test. Try different subprograms called via a single access type.
						9	For a call via an access-to-subprogram value created with S'Access, check that all enabled class-wide preconditions of S are checked.	C-Test. Try different subprograms called via a single access type.
						9	For a call via an access-to-subprogram value created with S'Access, check that all enabled class-wide postconditions of S are checked.	C-Test. Try different subprograms called via a single access type.
						8	For a call via an anonymous access-to-subprogram parameter value created with S'Access, check that the specific precondition of S is checked if it is enabled.	C-Test. Try different subprograms passed to the same subprogram parameter.
						8	For a call via an anonymous access-to-subprogram parameter value created with S'Access, check that the specific postcondition of S is checked if it is enabled.	C-Test. Try different subprograms passed to the same subprogram parameter.
						8	For a call via an anonymous access-to-subprogram parameter value created with S'Access, check that all enabled class-wide preconditions of S are checked.	C-Test. Try different subprograms passed to the same subprogram parameter.
						8	For a call via an anonymous access-to-subprogram parameter value created with S'Access, check that all enabled class-wide postconditions of S are checked.	C-Test. Try different subprograms passed to the same subprogram parameter.

			6	For a call via an access-to-protected-subprogram value created with S'Access, check that the specific precondition of S is checked if it is enabled.	C-Test. Try different subprograms called via a single access type.
			6	For a call via an access-to-protected-subprogram value created with S'Access, check that the specific postcondition of S is checked if it is enabled.	C-Test. Try different subprograms called via a single access type.
			6	For a call via an access-to-protected-subprogram value created with S'Access, check that all enabled class-wide preconditions of S are checked.	C-Test. Try different subprograms called via a single access type.
			6	For a call via an access-to-protected-subprogram value created with S'Access, check that all enabled class-wide postconditions of S are checked.	C-Test. Try different subprograms called via a single access type.
(41/5)	Dynamic	Based on the paragraph added by AI12-0272-1 (which is usually not relevant to Ada 2012). But this is explicitly redundant paragraph, and we haven't made these tests elsewhere.	7	For a call of a generic formal subprogram, check that the specific precondition of the actual subprogram is checked if it is enabled.	C-Test.
			7	For a call of a generic formal subprogram, check that the specific postcondition of the actual subprogram is checked if it is enabled.	C-Test.
			7	For a call of a generic formal subprogram, check that all enabled class-wide preconditions of the actual subprogram are checked.	C-Test.
			7	For a call of a generic formal subprogram, check that all enabled class-wide postconditions of the actual subprogram are checked.	C-Test.
(40/3)	NonNormative	A note.			

Paragraphs:		Objectives with tests:	Objectives to test:	Total objectives:	Objectives with submitted tests:
1 59		96	162	246	0

Must be tested	Objectives with Priority 10	5
	Objectives with Priority 9	28
Important to test	Objectives with Priority 8	20
	Objectives with Priority 7	60
Valuable to test	Objectives with Priority 6	33
	Objectives with Priority 5	10
Ought to be tested	Objectives with Priority 4	6
	Objectives with Priority 3	0
Worth testing	Objectives with Priority 2	0
Not worth testing	Objectives with Priority 1	0
	Total:	162

Objectives covered by new tests since ACATS 2.6	96
Completely:	84