

**Coverage for ISO/IEC 8652:2007 Ed. 3 in ACATS 3.x**  
**Clauses 6.5-6.7**

A Key to Kinds and subkinds is found on the sheet named **Key**. Tests new to ACATS 3.x are shown in **bold**.

Objective's

Clause	Para.	Lines	Kind	Subkind	Notes	Tests	New	Priority	Objective Text	Objective notes	Submitted tests (will need work).
6.5	(1/2)		Definitions		Return statement						
	(2/2)		Syntax								
	(2.1/3)		Syntax		AI05-0015 adds "constant";						
	(2.2/2)		Syntax		AI05-0053 will likely delete						
	(3/2)	1	Definitions		Result subtype						
		2	NameRes			C87B44A			Check that a call to an overloaded function as the expression of a simple return statement can be resolved if only one of the functions matches the type of the function containing the return statement. Check that a call to an overloaded function as the expression of an extended return statement can be resolved if only one of the functions matches the type of the function containing the return statement.		
		3	NameRes						7 the function containing the return statement.	C-Test. Look at C87B44A for inspiration. We need an extended return case for this, but any other C-Test will try it.	
	(4/2)	1	Legality	Widely used		C58004C, and many others.			Check that a return statement is allowed in a subprogram_body. Check that a return statement is allowed in an entry_body. Check that a return statement is allowed in an accept_statement.	C-Test. C-Test. There is definitely no existing test.	
				Negative		B58001A (package)			Check that a simple return statement is illegal if it is not within a callable construct. Check that an extended return statement is illegal if it is not within a callable construct.	B-Test. Try returning from a task body. B-Test. Try returning from a package body and task body.	
		2	Legality						Check that a simple return statement is illegal if it is within a body that is within the construct to which it applies. Check that an extended return statement is illegal if it is within a body that is within the construct to which it applies.	B-Test. Try returning from package and task bodies nested in blocks or the declarative part of subprograms. This was Ada 83 T1, but the only test created didn't follow the recommendation. B-Test. Try returning from package and task bodies nested in blocks or the declarative part of functions.	
	(5/3)	1	Legality	Widely used	Any legal function.						
				Negative		B65002A, B65002B			Check that a function is illegal if it does not contain a return statement.		
		2	Legality	Widely used	Any legal simple return statement.						
				Negative	B58002A, B58002B, B58002C were replaced; there was no entry body test.	<b>B650002</b>	All		Check that a simple return statement cannot have an expression if used in a procedure body, entry body, or accept statement.		

		Negative	B58002A and B58002B were replaced.	<b>B650002</b>	All	Check that a simple return statement cannot omit the expression if used in a function body.	
3	Legality	Negative Subpart	Any extended return test.	<b>B650002</b>	All	Check that a simple return statement inside of an extended return statement cannot have an expression.	
		Negative		<b>B650002</b>	All	Check that an extended return statement cannot be used to return from a procedure body, entry body, or accept statement.	
4	Legality	Subpart	Any extended return test using <b>constant</b> . Rule added by AI05-0015.	<b>B650002</b>	All	Check that an extended return statement cannot be used to return from an outer extended return statement.	
(5.1/2) (5.2/3)	Legality	Negative Portion	Lead-in for the bullets below.			Check that an extended return statement containing <b>7 constant</b> cannot omit an expression.	B-Test.
1	Legality	Subpart	Most extended return tests.				
2	Legality	Negative Subpart	Most extended return tests.	<b>B650001</b>	All	Check that the return_subtype_indication of an extended return statement cannot be an access_definition if the result subtype of the function it appears in is given by a subtype_mark.	
			"Covered by" is added by AI05-0032-1.			Check that if the result subtype of a function is class-wide, the return_subtype_indication of an extended_return_statement given within it can be any definite specific subtype that is covered by the class-wide result type.	C-Test. Best to wait until WG 9 approves this AI.
3	Legality	Negative Subpart	Many extended return tests.	<b>B650001</b>	All	Check that the return_subtype_indication of an extended return statement cannot have a different type than the result subtype of the function it appears in if that subtype is given by a subtype_mark.	
		Negative		<b>B650001</b>	All	Check that if the result subtype of a function is constrained, an extended return statement given within it cannot have an unconstrained return_subtype_indication.	
				<b>B650001</b>	All	Check that if the result subtype of a function is constrained, an extended return statement given within it is illegal if the return_subtype_indication does not statically match the result subtype.	
4	Legality					Check that if the result subtype of a function is unconstrained, the return_subtype_indication of an extended_return_statement given within it can be any definite subtype of the result type.	C-Test. (May be able to combine with other tests.)
						10	

			Negative		<b>B650001</b>	All	<p>Check that if the result subtype of a function is unconstrained, the return_subtype_indication of an extended_return_statement given within it can be unconstrained if an expression is given.</p> <p>Check that if the result subtype of a function is unconstrained, the return_subtype_indication of an extended_return_statement given within it cannot be indefinite unless an expression is given.</p>	C-Test. (Vaguely covered in B650001.)
(5.3/2)	1	Legality	Subpart	Any C-Test with an anon. access return subtype				
	2	Legality	Subpart	Any C-Test with an anon. access return subtype	<b>B650001</b>	All	<p>Check that the return_subtype_indication of an extended return statement cannot be a subtype_indication if the result subtype of the function it appears in is given by an access_definition.</p>	
	3	Definition		Accessibility level of extended return statement.	<b>B650001</b>	All	<p>Check that the subtype defined by the access_definition in the return_subtype_indication of an extended_return_statement is illegal if it does not statically match the return subtype of the function that it applies to.</p>	
(5.4/3)		Legality		This paragraph was added by AI05-0032-1. Lead-in for the bullets below. [Careful, this paragraph was renumbered by AI05-0032-1]			<p>If the result subtype of a function is class-wide, check that the accessibility level of the type of the return_subtype_indication of an extended return statement cannot be statically deeper than the master that elaborated the function.</p>	B-Test. Make sure to try cases without an expression. Perhaps wait until WG 9 approves AI05-0032 to issue a test.
(5.5/2)		Legality	Portion	This rule is redundant with 7.5(2.8/2) even though it isn't marked as such; we'll test it there. [Careful, this paragraph was renumbered by AI05-0032-1]				
(5.6/2)		Legality		6.5(8/2) contains a run-time version of this rule. This paragraph was renumbered by AI05-0032-1]			<p>If the result subtype of a function is class-wide, check that the accessibility level of the type of the return expression cannot be statically deeper than the master that elaborated the function.</p>	
(5.7/3)	1	Legality		AI05-0032-1.	<b>B650003</b>	All	<p>If the result subtype of a function has unconstrained access discriminants, the accessibility level of the type of each discriminant cannot be statically deeper than the master that elaborated the function.</p>	
	2			6.5(21/2) contains a run-time version of this rule. Defines the nominal subtype, affects other rules. [Careful, this paragraph was renumbered by AI05-0032-1]			<p>7</p>	B-Test. Good luck figuring out how to test this. ;-)
(5.8/3)	1	StaticSem	Subpart					

	2	Subpart	Added by AI05-0015. Defines the return object as a constant.			
(5.9/3)	1	Dynamic	Modified and renumbered by AI05-0032-1. Can't check that an anonymous access type is elaborated: it has no effect.		Check that the subtype of an extended return statement 6 is elaborated.	C-Test. Check that exceptions are raised if needed, and any functions are called.
	2	Not Testable Not Testable	No observable effect.			C-Test. Check that exceptions are raised for necessary, and any functions are called, and Adjust is called if needed. Priority is higher than usual for this sort of objective because the statement is new.
	3				Check that the expression of an extended return is 6 evaluated and converted to the nominal subtype.	C-Test. Check that value is correct, and that any functions are called. If Initialize is called when needed is an objective for 7.6(10/2).
	4				Check that an extended return statement without an 6 expression causes the return object to be initialized by default.	
	5				Check that an extended return statement with an object 6 of an indefinite subtype is constrained by its initial value.	C-Test. Try to change the bounds/discriminants.
	6, 7		Added by AI05-0032-1.		Check that Constraint_Error is raised if the return object 6 is not in the return subtype.	C-Test. This is thought to be only possible for class-wide return subtypes that have a constraint.
(6/2) (7/2)		Dynamic Redundant		C58005A (integer), C58005B (integer), C58005H (access), C58006A, C58006B (integer eval.)	Check that the expression of a simple return is 4 evaluated and converted to the result subtype of the function.	C-Test. Check constraints of array and record types. Check class-wide expressions for functions returning specific tagged types.
			Tested in 9.2.			C-Test. Check view conversions, class-wide expressions, and dereferences of access-to-classwide. Not tested in ACATS 2.x.
(8/3)	1	Dynamic			Check that the tag of the result of a function that returns 6 a specific tagged type is that of the function, even if the return expression has a different tag.	
	2	Dynamic	Changed by AI05-0032-1.	C390004 (simple returns)	Check that the tag of the result of a function that returns 5 a class-wide tagged type with a simple return statement is that of the expression.	C-Test.
					Check that the tag of the result of a function that returns 7 a class-wide tagged type with an extended return statement whose subtype indication has a class-wide type is the tag of the initializing expression.	C-Test.
					Check that the tag of the result of a function that returns 7 a class-wide tagged type with an extended return statement whose subtype indication has a specific type is the tag of the specific type.	C-Test.

3, 4	Dynamic		Changed by AI05-0024.	<p>Check that <code>Program_Error</code> is raised if the tag identified by the result object for a function returning a class-wide type has a master that does not include the elaboration of the master that elaborated the function body.</p>	<p>C-Test. Make sure to only test cases that aren't illegal by 6.5(5.6/2). Don't forget to test extended returns. Some ideas of how to do this can be found in AI05-024. Make sure to test an incomparable case as from AI05-0024.</p>
(8.1/3) (9/2) (10/2) (11/2) (12/2) (13/2) (14/2) (15/2) (16/2) (17/2) (18/2) (19/2) (20/2)	Dynamic Deleted Deleted Deleted Deleted Deleted Deleted Deleted Deleted Deleted Deleted Deleted		Added by AI05-0073-1.	<p>Check that <code>Constraint_Error</code> is raised if the result subtype of the function is an anonymous access type designating a specific tagged type and the result value is not null and designates some other specific type.</p>	<p>C-Test. Important because it was left out of the Standard, and we need to make sure that users aren't depending on this capability.</p>
(21/2)	Dynamic		Any legal extended return statement will do this. The wording was changed by AI05-0058-1, but it has no impact on testing.	<p>If the result subtype of a function has unconstrained access discriminants, check that <code>Program_Error</code> is raised if the accessibility level of the type of any access discriminant is deeper than the master that elaborated the function.</p>	<p>C-Test. Make sure to only test cases that aren't illegal by 6.5(5.6/2). Be careful that your head does not explode.</p>
(22/2)	1 2 3	Dynamic Subpart		<p>Check that a simple return statement in the <code>handled_sequence_of_statements</code> of an extended return statement completes the extended return statement and causes the function to return.</p>	<p>C-Test.</p>
			C58004C, C58004G	<p>Check that the completion of a simple return statement that applies to a function causes the function to return. Check that the completion of an extended return statement that applies to a function causes the function to return.</p>	<p>C-Test.</p>
			C58004C, C58004D, C58004G	<p>Check that the completion of a return statement that applies to a procedure causes the procedure to return (and not some enclosing subprogram). Check that the completion of a return statement that applies to an entry body causes the entry to return.</p>	<p>C-Test. (Can't find any such tests in ACATS 2.x).</p>

Check that the completion of a return statement that applies to an accept statement causes the accept statement to return.

C-Test. (Can't find any such tests in ACATS 2.x). RRS had a bug report on this, and it was the #2 item to add to the Ada 95 ACATS.

(23/2) Dynamic Subpart Constantness is defined in 3.3(15-22), and the results of that rule are tested elsewhere.

(24/2) Impl-Def NonNormative Not separately testable, but it needs to be taken into account when creating other tests. Also see AI05-0050, which extends and limits the permission. Start of example...

(25) NonNormative

(26/2) NonNormative

(27) NonNormative

(28/2) NonNormative ...end of example.

6.5.1

(1/2) General

(2/2) Syntax

(3/2) Syntax

(4/2) 1 Legality Subpart Legal tests will check this.

Negative

2

(5/2) Legality

(6/2) Legality

(7/2) Legality

(8/2) StaticSem Subpart An instance is non-returning if the generic is, as are non-returning procedures declared in a generic.

(9/2) Dynamic

Check that the name given in a pragma No\_Return cannot be a function.

Check that the name given in a pragma No\_Return cannot be an entry.

Check that the name given in a pragma No\_Return cannot be a non-subprogram.

Check that the name given in a pragma No\_Return cannot be a null procedure.

Check that the name given in a pragma No\_Return cannot be an instance of a generic procedure.

Check that a return statement cannot be used in a non-returning procedure.

Check that a procedure that overrides a dispatching non-returning procedure, the procedure must be non-returning.

Check that a renames-as-body that completes a non-returning procedure declaration renames a non-returning procedure.

B-Test.

B-Test.

B-Test. Try objects, exceptions, packages, tasks, protected objects, types.

B-Test.

B-Test.

B-Test. Check both simple and extended returns.

B-Test.

B-Test. Try renaming instances and procedures in instances.

Check that a non-returning procedure raises Program\_Error if it attempts to return normally.

C-Test. This is a procedure that falls off the end of the code. Try this in instances and in routines declared in generics.

Check that a non-returning procedure can propagate an  
 8 exception to "return" to the caller. C-Test.  
 Check that a non-returning procedure can be  
 6 dispatching. C-Test.

(10/2)		NonNormative	An example.		
6.6	(1)	1	Definitions	"operator".	
		2	Redundant		
	(2)		Definitions	Widely Used	Any use of user-defined operators tests this equivalence.
	(3)	1	Legality		B67001A (normal declarations), B67001B (formal subprograms), B67001D (renaming) B67001A, B67001B, B67001C, B67001D, B67001H, B67001I, B67001J, B67001K C67002A (normal), C67002B (case differences), C67002C (formal subprograms), C67002E (renames)
		2			B67001C
					C67002D B67001A (normal declarations), B67001B (formal subprograms), B67001C (instances), B67001D (renaming)
	(4)		Legality		Check that default expressions are not allowed in the parameters of an operator.
	(5)		Legality		Check that an explicit declaration of "/=" does not have a result of Boolean.
	(6)		StaticSem		Check that a declaration of "=" whose result is Boolean 3 declares a "/=" as well.
	(7)		NonNormative		Check that a declaration of "=" whose result is not Boolean does not declare a "/=".
	(8)		NonNormative	A note.	
	(9)		NonNormative	Start of example... ...end of example.	
6.7	(1/2)		General		
	(2/2)		Syntax		
	(3/2)	1	Definitions	"null procedure"	

C-Test or B-Test. No existing test found, but it seems likely that one exists somewhere, thus the low priority.

2	Legality		<b>B670001</b>	All	Check that a completion is not allowed for a null procedure.
(4/2)	Dynamic	Not Testable	Can't check "no effect", as we'd have to guess what effect the implementation would mistakenly have.		
(5/2)	Dynamic	Not Testable	Can't check "no effect", except to ensure that elaboration checks never fail.		
(6/2)	NonNormative		An example.		Check that a null procedure can be called when the body of the package it is contained in has not yet been elaborated. C-Test.

<b>Paragraphs:</b>		<b>Objectives with tests:</b>	<b>Objectives to test:</b>	<b>Total objectives:</b>	<b>Objectives with submitted tests:</b>
4	65	30	43	70	0
	Must be tested	Objectives with Priority 10	1		
		Objectives with Priority 9	6		
	Important to test	Objectives with Priority 8	7		
		Objectives with Priority 7	10		
	Valuable to test	Objectives with Priority 6	8		
		Objectives with Priority 5	7		
	Ought to be tested	Objectives with Priority 4	2		
		Objectives with Priority 3	1		
	Worth testing	Objectives with Priority 2	1		
	Not worth testing	Objectives with Priority 1	0		
		Total:	43		
		Objectives covered by new tests in ACATS 3.0	14		
		Completely:	14		