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

Clauses 6.5-6.8

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).
6.5	(1/2)		Definitions		Return statement						
	(2/2)		Syntax								
	(2.1/3)		Syntax		AI05-0277-1 gives the declaration it's own production.						
	(2.2/3)		Syntax		AI05-0015-1 adds "constant" (Ada2012).						
	(2.3/2)		Syntax		Paragraph number changed by AI05-0277-1.						
	(3/5)	1	Definitions		Result subtype						
		2	NameRes	Widely used	Basic resolution is tested in any test using a simple return statement.						
			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.		
				Negative		B58003A (normal, Integer), B58003B (generic, Integer)		2	Check that the type of the expression of a simple return statement must match the result type of the function.	B-Test; low priority as this is just normal resolution. We need to try anonymous access result cases, as well as tagged and untagged private types (where we try to return something of the full type).	
		3	NameRes	Widely used	Basic resolution is tested in any test using an extended return statement with an initializing expression. Text (but not meaning) changed by AI12-0173-1).						
			NameRes					7	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.	C-Test. Look at C87B44A for inspiration.	
								2	Check that the type of the initializing expression of an extended return statement must match the return subtype of the return statement.	B-Test; low priority as this is just normal resolution. We need to try anonymous access result cases, as well as tagged and untagged private types (where we try to initialize with something of the full type).	
	(4/2)	1	Legality	Widely used		C58004C, and many others. ***C650003*** (extended return)	All		Check that a return statement is allowed in a subprogram_body.		
						C650002	All		Check that a return statement is allowed in an entry_body.		
						C650002	All		Check that a return statement is allowed in an accept_statement.		
				Negative		***B650004***	All		Check that a simple return statement is illegal if it is not within a callable construct.		
						B650004	All		Check that an extended return statement is illegal if it is not within a callable construct.		

	2	Legality		"Construct to which it applies" can be a procedure, function, entry body, accept statement, or extended return statement.	***B650004***	All	Check that a simple return statement is illegal if it is within a body that is within the construct to which it applies.
					B650004	All	Check that an extended return statement is illegal if it is within a body that is within the construct to which it applies.
(5/5)	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.	**B650002**	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.	**B650002**	All	Check that a simple return statement cannot omit the expression if used in a function body.
			Negative		**B650002**	All	Check that a simple return statement inside of an extended return statement cannot have an expression.
	3	Legality	Subpart	Any extended return test.			
			Negative		**B650002**	All	Check that an extended return statement cannot be used to return from a procedure body, entry body, or accept statement.
					B650002	All	Check that an extended return statement cannot be used to return from an outer extended return statement.
	4	Legality	Subpart	Any extended return test using **constant**. Rule added by AI05-0015, then text modified by AI12-0173-1.			
			Negative		***B650006***	All	Check that an extended return statement containing **constant** cannot omit an expression.
(5.1/5)		Definition	Subpart	Definition of expression of an extended return, added by AI12-0173-1. Widely used in other rules, no semantic change intended. Does change paragraph numbers of below paragraphs.			
(5.2/5)		Legality	Portion	Lead-in for the bullets below.			
(5.3/5)	1	Legality	Subpart	Most extended return tests.			
			Negative		**B650001**	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.
	2	Legality	Subpart	Most extended return tests.			
				"Covered by" is added by AI05-0032-1.	***C650B03*** (nonlimited), ***C650A02*** (two limited cases in auxillary function)	All	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.
			Negative		**B650001**	All	Check that the return_subtype_indication of an extended return statement cannot fail to be covered by the result subtype of the function it appears in if that subtype is given by a subtype_mark.

	3	Legality	Subpart	Many extended return tests. Substantially changed by AI05-0103-1.					
			Negative		**B650001**	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.	This objective is OK, even given the new wording (it is just more limited than necessary).
					B650001	All		Check that if the result subtype of a function is elementary, an extended return statement given within it is illegal if the return_subtype_indication does not statically match the result subtype.	
					B650001	All		Check that an extended return statement is illegal if the return_subtype_indication is not statically compatible with the result subtype.	
	4	Legality					8	Check that if the result subtype of a function is indefinite, the return_subtype_indication of an extended_return_statement given within it can be any definite subtype of the result type.	C-Test. Class-wide cases have their own objective above; this objective covers discriminated records and unconstrained arrays. Combine with the following??
					C650A02 (limited, class-wide)		7	Check that if the result subtype of a function is indefinite, the return_subtype_indication of an extended_return_statement given within it can be indefinite if an expression is given.	C-Tests. (Vaguely covered in B650001.) Still need a test for discriminated records and for unconstrained arrays.
			Negative		**B650001**	All		Check that if the result subtype of a function is indefinite, the return_subtype_indication of an extended_return_statement given within it cannot be indefinite unless an expression is given.	
(5.4/5)	1	Legality	Subpart	Any C-Test with an anon. access return subtype					
			Negative		**B650001**	All		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.	
	2	Legality	Subpart	Any C-Test with an anon. access return subtype					
			Negative		**B650001**	All		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.	
	3	Definition		Accessibility level of extended return statement.					
(5.5/5)		Legality		This paragraph was added by AI05-0032-1.	***B650005***	All		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.	
(5.6/5)		Legality	Portion	Lead-in for the bullets below. [Careful, this paragraph was renumbered by AI05-0032-1]					
(5.7/5)		Redundant		This rule is redundant with 7.5(2.8/2); we'll test it there. [Careful, this paragraph was renumbered by AI05-0032-1]					
(5.8/5)		Legality		6.5(8/2) contains a run-time version of this rule. This paragraph was renumbered by AI05-0032-1.	**B650003**	All		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.	

(5.9/5)		Legality		6.5(21/3) contains a run-time version of this rule. This paragraph was split from the preceding paragraph by AI05-0051-1.			7	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.	B-Test. Good luck figuring out how to test this. ;-) Note: The rule applies to constrained access discriminants as well, but that cannot fail.
(5.10/5)		Legality		Added by AI05-0277-1.			5	If the result subtype of a function is immutably limited, check that the keyword aliased can be used in an extended_return_object_declaration.	C-Test. We include a C-Test here because this is likely to be rare and thus not tested much elsewhere (the only other known test would be in 3.10).
			Negative		***B650007***	All		If the keyword aliased is present in an extended_return_object_declaration, check that the type of the result object cannot be any type that is not immutably limited.	
(5.11/5)	1	StaticSem	Subpart	Defines the nominal subtype, affects other rules. [Careful, this paragraph was renumbered by three AIs]					
	2		Subpart	Added by AI05-0015. Defines the return object as a constant.					
(5.12/5)	1	Dynamic		Modified by AI05-0032-1; renumbered by 3 AIs.			6	Check that the subtype of an extended return statement is elaborated.	C-Test. Check that exceptions are raised if needed, and any functions are called.
			Not Testable	Can't check that an anonymous access type is elaborated: it has no effect.					
	2		Not Testable	No observable effect.					
	3						6	Check that the expression of an extended return is evaluated and converted to the nominal subtype.	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.
	4						6	Check that an extended return statement without an expression causes the return object to be initialized by default.	C-Test. Check that value is correct, and that any functions are called. If Initialize is is called when needed is an objective for 7.6(10/2).
	5						6	Check that an extended return statement with an object 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.			6	Check that Constraint_Error is raised if the return object 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)		Dynamic			C58005A (integer), C58005B (integer), C58005H (access), C58006A, C58006B (integer eval.)		4	Check that the expression of an simple return is 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.
(7/2)		Redundant		Tested in 9.2.					
(8/4)	1	Dynamic			***C650B01***	All		Check that result of a function that returns a specific tagged type has the tag of the tagged result type, even if the return expression has a different tag.	
	2	Dynamic		Changed by AI05-0032-1 and AI12-0097-1.	C390004 (simple returns of a local object), ***C650A02*** (returns of limited expressions), ***C650B02*** (returns of non-limited expressions)	All		Check that the tag of the result of a function that returns a class-wide tagged type with a simple return statement is that of the expression.	

					C650A02 (returns of limited expressions), ***C650B02*** (returns of non-limited expressions)	All		Check that the tag of the result of a function that returns 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.	
					C650B03 (nonlimited), ***C650A02*** (two limited cases in auxillary function)	All		Check that the tag of the result of a function that returns a class-wide tagged type with an extended return statement whose subtype indication has a specific type is the tag of the specific type.	
	3, 4	Dynamic		Changed by AI05-0024-1.	***C650B04*** (nonlimited, simple cases)	Part	5	Check that Program_Error 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.	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. Still need to do incomparable cases like those found in AI05-024 (but hold for resolution of AI12-0016-1). Try to use foundation F650B00 for the basic types.
(8.1/5)	1	Dynamic	Subpart	Any legal extended return statement will do this. The wording was changed by AI05-0058-1, and the paragraph moved by AI12-0343-1, but neither have any impact on testing.					
	2				***C650003***	Part	4	Check that a simple return statement in the handled_sequence_of_statements of an extended return statement completes the extended return statement and causes the function to return.	C-Test. Try this with other kinds of types (arrays, anonymous access, etc.).
	3				C58004C, C58004G			Check that the completion of a simple return statement that applies to a function causes the function to return.	
					C650003	All		Check that reaching the **end return** of an extended return statement that applies to a function causes the function to return.	This test just tries a limited record type. Other kinds of types will be tried with C-Tests for other objectives.
					C650003	All		Check that completing an extended return statement by an exit, goto, or exception propagation does not cause the function that the extended return applies to to return.	
					C58004C, C58004D, C58004G			Check that the completion of a return statement that applies to a procedure causes the procedure to return (and not some enclosing subprogram).	
					C650002	All		Check that the completion of a return statement that applies to an entry body causes the entry to return.	
					C650002	All		Check that the completion of a return statement that applies to an accept statement causes the accept statement to return.	
(8.2/5)		Dynamic		Added by AI05-0073-1. Renumbered by AI12-0343-1.	***C650A01***	All		Check that Constraint_Error 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.	
							7	Check that Constraint_Error 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. Case 2: The value is set in the sequence_of_statements of an extended return.	C-Test. Use C650A01 as the outline for the test. This case is motivated by AI12-0343-1, but the requirement is unchanged.
(9/2)		Deleted							
(10/2)		Deleted							
(11/2)		Deleted							
(12/2)		Deleted							
(13/2)		Deleted							

(14/2)	Deleted					
(15/2)	Deleted					
(16/2)	Deleted					
(17/2)	Deleted					
(18/2)	Deleted					
(19/2)	Deleted					
(20/2)	Deleted					
(21/3)	Dynamic		Rule was substantially modified by AI05-0051-1.	8	If the result subtype of a function has access discriminants, check that Program_Error is raised if the accessibility level of the type of any corresponding access discriminant is deeper than the master of the call.	C-Test. Make sure to only test cases that aren't illegal by 6.5(5.9/5). Be careful that your head does not explode. Include cases where the result is modifed in the sequence_of_statements of an extended return statement.
				6	If any subcomponent of the specific result subtype of a function has access discriminants, check that Program_Error is raised if the accessibility level of the type of any corresponding access discriminant is deeper than the master of the call.	C-Test. Be sure to test cases where the presence of access discriminants is only known at run-time, and cases where they don't actually exist. (See the AARM notes.)
(22/5)	Dynamic		Check was added by AI12-0343-1.	5	Check that a predicate check is made on the result of a function, even if that result is modified by the sequence_of_statements of an extended return statement.	C-Test. As a BI, it can be tested even though it is an Ada 202x AI. Test cases like the one in the AARM note where a component of the result is modified in the sequence of statements. (Such that the predicate is not checked in the statements.)
(23/2)	Dynamic	Subpart	Constantness is defined in 3.3(15-22), and the results of that rule are tested elsewhere.			
(24/3)	Impl-Def	Subpart	Not separately testable, but it needs to be taken into account when creating other tests. Modified by AI05-0050, now a lead-in.			
(24.1/3)	Impl-Def		A permission, added by AI05-0050.			
(24.2/3)	Impl-Def		A permission, added by AI05-0050.			
		Negative		8	Check that if the result subtype of a function is unconstrained and the return object is not known to be constrained, Constraint_Error is not raised before the entire function executes	C-Test. We're checking that the permission is not applied inappropriately. The return object should have discriminants with defaults (the wrong defaults), be default-initialized, and the discriminants should be changed to the correct ones before returning (so that no exception should be raised).
		Negative		7	Check that if the result subtype of a function is an unconstrained elementary type, and the return object in an extended return statement is initialized to be out-of-range for the result object, Constraint_Error is not raised until the entire extended return statement has executed	C-Test. We're checking that the permission is not applied to elementary type functions. Use Integer'Base to get an unconstrained discrete type. Also try float and access types (not null).
(25)	NonNormative		Start of example...			
(26/2)	NonNormative					
(27)	NonNormative					
(28/2)	NonNormative		...end of example.			

6.5.1	(1/5)		General		AI05-0229-1 rewrites the entire section in terms of aspects; AI12-0269-1 extends the aspect to all subprograms.					
	(2/3)		Deleted		Deleted by AI05-0229-1.					
	(3/3)		Deleted		Deleted by AI05-0229-1.					
	(3.1/3)		Definitions	Lead-in						
						B651003	All		Check that aspect No_Return cannot be specified for a function (including an expression function) or generic function.	
				Negative		***B651003***	All		Check that the aspect No_Return cannot be specified for a entry.	
				Negative		***B651003***	All		Check that the aspect No_Return cannot be specified for a non-subprogram.	
	(3.2/3)	1	NameRes					2	Check that the expected type of the expression specified for aspect No_Return is Boolean.	B-Test; low priority as this is just normal resolution and as the expression has to be static, its hard to test any meaningful overloading cases.
		2	Definitions		Defines “nonreturning”. Other tests will test this definition.					
	(3.3/3)	1	Legality	Subpart	Legal tests will check this.					
				Negative		***B651003***	All		Check that the expression specified for aspect No_Return cannot be nonstatic.	
		2	Definitions	Widely Used	Defines that the aspect is not set by default. Any test that doesn't use Non_Returning implicitly is testing this.					
	(3.4/3)		Definitions		This rule should be tested as part of other tests, specifically that of paragraph 9.					
	(4/3)		Legality			***B651003***	All		Check that the aspect No_Return cannot be specified for a null procedure.	
						B651003	All		Check that the aspect No_Return cannot be specified for a generic instance of a procedure.	
	(5/2)		Legality			***B651001*** (pragma), ***B651002*** (aspect)	All		Check that a return statement cannot be used in a nonreturning procedure.	B-Test. Check simple returns, both at the outer level and nested inside of statements and blocks. Check both generic and non-generic subprograms.
	(6/3)		Legality			***B651004*** (aspect)	Parti al	2	Check that a procedure that overrides a dispatching nonreturning procedure must be nonreturning.	B-Test needed for the pragma (but it's obsolescent).
								6	Check that a nonreturning procedure can be dispatching.	C-Test. This is a corollary of this rule.
	(7/3)		Legality		Modified by AI12-0269-1	***B651001*** (pragma), ***B651002*** (aspect)	All		Check that a renames-as-body that completes a nonreturning procedure declaration renames a nonreturning procedure.	
	(8/3)		Deleted		Deleted by AI05-0229-1.					
	(9/2)		Dynamic			***C651001*** (pragma), ***C651002*** (aspect)	All		Check that a nonreturning procedure raises Program_Error if it attempts to return normally.	
						C651001 (pragma), ***C651002*** (aspect)	All		Check that a nonreturning procedure can propagate an exception to "return" to the caller.	
	(10/3)		NonNormative		An example.					
6.6	(1)	1	Definitions		"operator".					
		2	Redundant							

	(2)		Definitions	Widely Used	Any use of user-defined operators tests this equivalence.					
	(3/3)	1	Legality		Modified by AI05-0143-1.	B67001A (normal declarations), B67001B (formal subprograms), B67001D (renaming)			Check that the subprogram declaration defining an operator cannot have more or less parameters than defined by the kind of operator (unary or binary).	
						B67001A, B67001B, B67001C, B67001D, B67001H, B67001I, B67001J, B67001K			Check that non operators (membership, short circuit, assignment) cannot be used in operator symbols.	
						C67002A (normal), C67002B (case differences), C67002C (formal subprograms), C67002E (renames)			Check that a subprogram declaration for an operator symbol can be given if the number of parameters is correct for the kind of operator (unary or binary).	
						B660003	All		Check that parameters of mode **in out** and **out** are not allowed in the declaration of operators.	
		2				B67001C			Check that an instance defining an operator cannot have more or less parameters than defined by the kind of operator (unary or binary).	
						C67002D			Check that a instance can be named by an operator symbol can be given if the number of parameters is correct for the kind of operator (unary or binary).	
						B660003	All		Check that a generic function with a parameter of mode **in out** or **out** cannot be instantiated as an operator.	
	(4)		Legality			B67001A (normal declarations), B67001B (formal subprograms), B67001C (instances), B67001D (renaming)			Check that default expressions are not allowed in the parameters of an operator.	
	(5)		Legality			B660001, B660002			Check that an explicit declaration of "/=" does not have a result of Boolean.	
	(6/3)		StaticSem		Modified by AI05-0128-1.	***C660001***	All		Check that an explicit declaration of "=" whose result is Boolean declares a "/=" as well.	The test tries the tagged case; the untagged case occurs for various language-defined packages including Ada.Strings.Unbounded, so a bug would turn up in virtually any test or use of those packages – a separate test is unnecessary.
						B660002			Check that a declaration of "=" whose result is not Boolean does not declare a "/=".	
						C660001	All		Check that a declaration of "/=" implicitly created by the declaration of "=" with a Boolean result is inherited for a derived type.	
	(7)		NonNormative		A note.					
	(8)		NonNormative		Start of example...					
	(9)		NonNormative		...end of example.					
6.7	(1/2)		General							
	(2/3)		Syntax		AI05-0183-1 adds aspect clauses; these will be tested as part of other rules.					
	(2.1/3)	1	Legality					4	Check that a null procedure can be the completion of a procedure or generic procedure declaration.	C-Test.

				Negative				6	Check that a null procedure cannot complete a function declaration or any kind of subprogram body.	B-Test.
		2	Legality					6	Check that a null procedure that completes a procedure or generic procedure declaration must fully conform to the profile of that declaration.	B-Test. We don't need to check all of the conformance rules here, just a small selection to ensure that the check is made.
	(3/2)	1	Definitions		"null procedure"					
		2	Legality			**B670001**	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.					
	(5/3)		Dynamic	Not Testable	Can't check "no effect", except to ensure that elaboration checks don't fail. Any call to a null procedure will test that.					
				Negative				3	Check that a call to a procedure that is completed by a not yet elaborated null procedure raises Program_Error.	C-Test. Low priority because it's hard to construct such a case, so it's pretty unlikely – and nothing bad will happen even if the check is omitted. Could base on the test from C680001.
	(6/2)		NonNormative		An example.			4	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.
6.8	(1/3)		General		This entire subclause was added by AI05-0177-1.					
	(2/4)		Syntax		Aggregate was added by AI12-0157-1.					
	(3/4)		NameRes		Aggregate was added by AI12-0157-1.			5	Check that a call to an overloaded function as the expression of a expression function can be resolved if only one of the functions matches the result type of the expression function.	C-Test. Not very important as it's just normal resolution.
				Negative		**B680001**	All		Check that the type of the expression of an expression function must match the result type of the expression function.	We could test additional cases (the test only tries two simple cases) but this is unimportant as this is just normal resolution.
	(4/3)	1	Legality			**C680001**	All		Check that an expression function can be the completion of a function or generic function declaration.	
						B680001	All		Check that an expression function cannot complete a procedure declaration, a package declaration, or any kind of body.	Could have tried other kinds of program units (protected, tasks) and additional bodies, but those are just normal homograph rules.
		2	Legality			**B680001**	All		Check that an expression function that completes a function or generic function declaration must fully conform to the profile of that declaration.	Could have tried more cases of conformance (the test only tries 3), but we expect the conformance rules to be throughly tested in subclause 6.3.1.
	(5/4)		Legality		Aggregate was added by AI12-0157-1.			7	If the result subtype of an expression 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.	B-Test. Good luck figuring out how to test this. ;-) [But it's the same as 6.5(5.8/3).] It's not clear that it is testable here, as no local objects are possible.
	(6/4)	1	Definition		“expression function”.					
		2	Definition		“return expression”					
		3	Legality			**B680001**	All		Check that a completion is not allowed for an expression function.	There really is only one way to do this sensibly, other cases usually are normal homograph violations.

(7/4)	Dynamic			**C680001**, ***C760A03*** (limited)	Part	4	Check that a call to an expression function executes as a body containing only a simple return of the expression of the expression function.	C-Test. Try cases that fail the checks described in 6.5 for a simple return (in particular, the various tag checks).
				B732C01, ***C760A03***	All		Check that an aggregate can directly be the return expression of an expression function.	
				C680001	All		Check that an expression function can include a recursive call on itself.	This test ensures that the implementation can deal with expression functions that cannot be inlined.
(8/3)	Dynamic	Not Testable	Can't check "no effect", except to ensure that elaboration checks don't fail. Any call to an expression function will test that.					
		Negative		**C680001**	All		Check that a call to a function that is completed by a not yet elaborated expression function raises Program_Error.	
(9/3)	NonNormative		An example.					

	Paragraphs:		Objectives with tests:	Objectives to test:	Total objectives:	Objectives with submitted tests:
5	86		76	32	101	0

Must be tested	Objectives with Priority 10	0
	Objectives with Priority 9	0
Important to test	Objectives with Priority 8	3
	Objectives with Priority 7	6
Valuable to test	Objectives with Priority 6	9
	Objectives with Priority 5	4
Ought to be tested	Objectives with Priority 4	5
	Objectives with Priority 3	1
Worth testing	Objectives with Priority 2	4
Not worth testing	Objectives with Priority 1	0
	Total:	32

Objectives covered by new tests since ACATS 2.6	61
Completely:	57