ACATS Modification List for ACATS 4.0
Version 4.0M
June 29, 2016

What is this?

This is the ACATS modification list for ACATS 4.0. This document includes test corrections, test removals, and new tests for ACATS 4.0. ACATS 4.0 is the current test suite for Ada, covering the third edition of Ada, ISO/IEC 8652:2012 (usually known as "Ada 2012").

ACATS Modification List version 3.1M for ACATS 3.1 is the current list for that test suite. ACATS 3.1 should be used for testing implementations that support Amendment 1 (ISO/IEC 8652:1995/AMD 1:2007, often known as "Ada 2005"), and will continue to be maintained as necessary for the foreseeable future.

ACATS Modification List version 2.6D for ACATS 2.6 is the current list for that test suite. ACATS 2.6 should be used for testing implementations of the second edition of Ada, ISO/IEC 8652:1995 (usually known as "Ada 95"); it also will continue to be maintained as necessary for the foreseeable future.

Changes from the last list:

Tests B38003D, B455002, and B83F02C had incorrect error tags repaired.

Tests B3A1001 and B3A1002 had Pack6 split so that the possible error sets do not intersect.

Test C52101A was corrected to not have a comment in the form of an ERROR: tag. These should not appear in C-Tests.

Tests B62001C and B62001D were changed to make the second error optional, as the first error is illegal syntax and thus has to be detected.

The Pass/Fail criteria for test B740003 was updated so that the test tools don't pick up an error marker.

“Note” was replaced by “Optional Error” in test B95007B.

The error marker for the second error in test BD4003A was moved so that it was on the incorrect construct.

Tests CXB3023 and CXB3024 were corrected to have missing N/A => ERROR markers added.

A number of tests changed status on April 1st, this list reflects those changes.

New tests posted between April 1, 2016 and June 30, 2016 will become required for conformity assessments on January 1, 2017. New tests posted between January 1, 2016 and March 31, 2016 will become required for conformity assessments on October 1, 2016. New tests posted between October 1, 2015 and December 31, 2015 will become required for conformity assessments on July 1, 2016. New tests posted before September 30, 2015 are now required for conformity assessments. All tests posted to modification lists for other versions of the ACATS are included in ACATS 4.0 if appropriate and thus are required for conformity assessment using ACATS 4.0. The effective dates of tests are marked in the lists.

Questions about this document? Contact the ACAA Technical Agent at agent@ada-auth.org.

Technical Corrigendum 1:
Beginning with ACATS Modification List 4.0B, we have included tests for corrections (including new features) in Technical Corrigendum 1, ISO/IEC 8652:2012/Cor 1:2016. All of these changes are corrections to Ada 2012; as such we will assume all such corrections are implemented in Ada 2012 compilers.

ACATS 4.1 Announced

The ACAA has announced its intention to create a new version of the ACATS under the baselining policy. Such a new ACATS version will include new and modified test files that are effective by the official use date of the new ACATS (in this case, July 1, 2016).

ACATS 4.1 will include optional Test Grading tools

ACATS 4.1 will introduce new test grading tools. The new version of Report included in this ACATS Modification List is the first piece of those tools.

When the ACATS was designed (as the ACVC in the early 1980s), the intention always was that running it would give a simple and clear Pass or Fail result. However, grading of tests (particularly of B and L Tests) is somewhat subjective and very time-consuming. (Test grading typically involved poring over compiler listings of the entire ACATS with a large highlighter.) The subjectivity was acceptable in the days of common formal testing, as the formal testers would enforce a common grading standard (even if no one could describe what that was), but in an age of self-testing, it reduces the value of the ACATS.

As such, we're introducing a long-overdue set of automated grading tools. These tools are intended to:

- Greatly reduce the effort to grade an ACATS test run (and, hopefully to reduce the cost of formal testing to make it a more practical option);
- Ensure that all users of the ACATS grade tests in the same way, thus making it more valid to compare ACATS results between different implementations and different testers;
- Give implementers certainty as to whether a particular test result is passing (whether the location of an error is “close enough” to the canonical position);
- Be able to detect process errors (especially errors of omission) – that is, tests improperly run (for instance, files compiled in the wrong order where the ACATS requires an order), or not run at all (if one is checking off lines in a listing, it's hard to notice that some test or portion of a test is missing).

The tools also will optionally give feedback to implementers about quality of error message positioning and of error cascades (does one error generate multiple messages?). (Since “goodness of an implementation” is not something that the ACATS tests for, this is not an official goal for the tools.)

This is an ambitious set of goals. It's unlikely that they'll be completely met by the initial versions of the tools. As such, their use will be completely optional (but encouraged!) for ACATS 4.1. The tools will be updated periodically by the normal process for updating the ACATS – an ACATS Modification List like this one.

It's probably obvious to most readers that these grading tools will require the compiler and binder output to be put into a common form (called an “event trace” in ACATS 4.1). We've settled on a .csv file format, as it is simple to write in an Ada program, easy to check in any spreadsheet program, and relatively easy to read in Ada code. To see an example, look at the new version of the Report package. The 60-line Put_Event_Trace routine writes a single event to the trace file. Most of the code is related to ensuring that the message doesn't contain any double quote characters, and to truncating the message to a reasonable length.

There are two ways to put event trace construction into an existing Ada implementation. One is to add compiler and binder switches to turn on event trace writing. The compiler and binder would then write events in their error handlers and other strategic locations. Combined with the event writing in Report (by
changing the Generate_Event_Trace_File constant to True), a complete event trace can easily be created for any portion of a test run.

Alternatively, one could create a program to analyze a compiler/binder/run listing and convert the events found in it to appropriate event trace events. With such a tool, all of the events would be extracted from the listing, and the event writing in Report would not be used. This approach would be more appropriate if the ACATS user cannot change the compiler and/or binder.

Much more on this topic, and related topics, will be found in the ACATS 4.1 documentation. As always, if you have questions or would like to volunteer to help test the tools, contact the ACAA Technical Agent at agent@ada-auth.org.

Notice:

The ACAA has created and maintains the Ada Conformity Assessment Test Suite for the purpose of conformity assessments conducted in accordance with the International Standard ISO/IEC 18009 - Ada: Conformity assessment of a language processor. This test suite should not be used to make claims of conformance unless used in accordance with ISO/IEC 18009 and any applicable ACAA procedures.

List explanation:

Test numbers given are for the basic test. Categories are “Withdrawn” (the test should not be processed for Ada conformity assessments); “Allowed Modification” (either the original test or the modified one should be processed for Ada conformity assessments); “Modified” (the modified test must be processed for Ada conformity assessments); “Pending New” (the test does not need to be processed for Ada conformity assessments, but will be added in the future); “New” (the test is processed for Ada conformity assessments).

ACATS version labels are provided with all new and modified tests. Use this label to retrieve the test from the ACATS VCS. See “How to get new and modified tests”.

In citations, line numbers are given relative to the start of a test file in the format of '@<line#,...>; if a test program comprises more than one file, then the particular file will be indicated by "fn", where "n" is the test-program name's file-designator numeral. E.g., "B38103E : @f1-31, ... " denotes test file b38103e1.ada. In cases where there are two or more rationales for different errors in a test program, these different rationales with their respective line citations are listed on separate lines. References to the Ada standards are given in the format:

@section|chapter.<clause>[<subclause>]:<paragraph>

These references are to the Ada standard as available at http://www.adaic.org/resources/add-contentstandards/12rm/html/RM-TOC.html.

Main list:

====== Core Language ======

Name    Category  Version label  (effective date; old version label)  Short description.

C324001 [Modified] VCS Label: A4_0E
     Eliminated overlong lines.
C324002 [Modified] VCS Label: A4_0F
   [VCS Label A4_0E] Eliminated overlong lines.
   [VCS Label A4_0F] Replaced last test case (qualified expressions), as that check is required by the Corrigendum.

C324003 [New] VCS Label: A4_0H
   [VCS Label A4_0E] Check that predicates are checked in the correct order and that the specified exception is raised.
   [VCS Label A4_0H] Missing function Name is defined.

C324004 [New] VCS Label: A4_0F
   Check predicate checks are not performed for static and dynamic subtype predicates when the assertion policy is Ignore, but memberships, the Valid attribute, and for loops continue to use the predicates -- case A: global assertion policy pragmas

C324005 [New] VCS Label: A4_0F
   Check predicate checks are not performed for static and dynamic subtype predicates when the assertion policy is Ignore, but memberships, the Valid attribute, and for loops continue to use the predicates -- case B: specific assertion policy pragmas

B38003D [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
   Required after 2016-10-01
   Corrected error tags @70,76,82,88 so “optional” and “error” are on the same line, as required by the test standards.

C390012 [New] VCS Label: A4_0H
   Check the operation of Ada.Tags.Generic_Dispatching_Constructor.

C391003 [New] VCS Label: A4_0E
   [VCS Label A4_0D] Check that a type extension can be declared in a more nested scope than its parent type, and that overridden subprograms of that extension can be dispatched to.
   [VCS Label A4_0E] Eliminate an overlong line.

B393011 [New] VCS Label: A4_0D
   Check that an abstract routine of an abstract partial view overridden by a non-abstract routine in the private part requires overriding when it is inherited if the private part is not visible where it is inherited.

C3A0030 [Pending New] New VCS Label: A4_OK
   Required after 2016-10-01
   Check that an access discriminant is only null excluding if it includes a null exclusion.

B3A1001 [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
   Required after 2016-10-01
   Split Pack6 so possible error sets don't intersect; fixed overlong lines.

B3A1002 [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
   Required after 2016-10-01
   Split Pack6 so possible error sets don't intersect; fixed overlong lines.

C3A1003 [New] VCS Label: A4_0E
   [VCS Label A4_0D] Check that a dereference of an access-to-tagged-incomplete type can be passed as a parameter.
   [VCS Label A4_0E] Eliminated overlong lines.
Check that a parameter of a tagged incomplete type can be passed as a parameter.
Eliminated overlong lines.

Check that a dereference of an access-to-incomplete type cannot be used as a prefix.
Eliminated overlong lines.

Check that a dereference of an access-to-incomplete view cannot be used as a prefix. Check that a parameter of an incomplete view cannot be used as a prefix.
Eliminated overlong lines.

Check that a dereference of an access-to-formal incomplete type cannot be used as a prefix. Check that a parameter of a tagged formal incomplete type cannot be used as a prefix.

Check that the type of the prefix of 'Access is used to resolve the expected type of the use of the attribute.

Check P'Access is not allowed in a generic body if P is declared in the generic (including in the parent of a child or in a formal package) and the access type is a formal type or outside of the generic.
Eliminated overlong lines.

Check various rules associated with the specification of aspect Implicit_Dereference.

Expression @155-6 violates 6.4.1(6.17/3).

Check that the object denoted by a generalized reference cannot be used as a variable if the discriminant has an access-to-constant type.

Check that a generalized indexing is illegal in a variable context if no Variable_Indexing is specified and Constant_Indexing specifies a function returning an ordinary object.

When both Constant_Indexing and Variable_Indexing are specified, check that a generalized indexing is illegal if it is called in variable contexts when the prefix is a constant and Constant_Indexing specifies a function returning an ordinary object.

Check that the name specified for a Constant_Indexing or Variable_Indexing aspect can refer to a set of overloaded functions, with two or more parameters, and that all of those
functions can be used in a generalized indexing.

C416A02 [Pending New] New VCS Label: A4_0J
  Required after 2016-07-01
  [VCS Label A4_0F] Check that the overloaded routines can be added to an inherited indexing aspect and the new routines can be used in a generalized indexing.
  [VCS Label A4_0H] Correct package C416A02_A to be a child of the foundation.
  [VCS Label A4_0J] Remove .K from line @211 so test reflects expected trace.

C431002 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Check that a component association of others => <> in a record or extension aggregate may have any number of associated components, including none. (Case 2: Null records and extensions.)
  [VCS Label A4_0E] Eliminated overlong line.

B431005 [New] VCS Label: A4_0B
  Check that a component association with an expression in a record or extension aggregate that have more than one associated component is illegal if the expression is illegal for any one of the associated components.

B431006 [New] VCS Label: A4_0C
  Check that if a variant part is not nested in an unselected variant, the value of the governing discriminant of a variant in a record aggregate cannot be non-static.

B43102A [Withdrawn] VCS Label: A4_0C
  Replaced by B431006.

C433002 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Check that no array components can be outside of the index subtype for an array aggregate, even when the components are specified by <>.
  [VCS Label A4_0E] Eliminated overlong lines.

B433003 [New] VCS Label: A4_0F
  Check that an others choice is not allowed in an array aggregate that is used as the return expression of an expression function or the expression of a return statement if the result subtype of the function is unconstrained.

C433003 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Check that for each association with a box in an array aggregate, the component is initialized by default (scalar types with Default_Values).
  [VCS Label A4_0E] Eliminated overlong lines.

C433004 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Check that for each association with a box in an array aggregate, the component is initialized by default (array types with Default_Component_Values).
  Object O6 @136 expects the wrong result.
  Eliminated overlong lines.

C433005 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Check that the constraint of the constrained array subtype of a function return is used to determine the
bounds of an array aggregate with an others choice in the expression of a return statement.

[VCS Label A4_0E] Eliminated overlong lines.

C433006 [New] VCS Label: A4_0F
Check that the constraint of the constrained array subtype of a function return is used to determine the bounds of an array aggregate with an others choice in the return expression of an expression function.

B455002 [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
Required after 2016-10-01
Corrected casing of error tags and fixed overlong lines.

B457002 [New] VCS Label: A4_0D
Check that 'others' must be last and alone in a case expression.

B457003 [New] VCS Label: A4_0D
Check that 'else' cannot be omitted for an if expression if the type of the expression is not a boolean type.

B457004 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that case choice coverage and staticness are enforced for case expressions.
[VCS Label A4_0E] Eliminated overlong lines.

B457005 [New] VCS Label: A4_0D
Check that all or none of the dependent expressions of a conditional expression can be dynamically tagged if the type of the expression is a specific tagged type.

C457005 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
When the selecting expression of a case expression is a name with a static nominal subtype and has a static predicate, the static predicate is disabled, and the value of the selecting expression does not satisfy the predicate, check that the others choice is returned, or, if there is no others choice, that Constraint_Error is raised.

B457006 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that an if expression has to be surrounded by parentheses if it is not already surrounded by them.

C457006 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that overloaded functions can be resolved when they appear as dependent expressions in a conditional expression. Part 1: Enumeration literals.

B457007 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that if the type of a conditional expression is not determined by the resolution rules, it is illegal.

C457007 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that literals can be resolved when they appear as dependent expressions in a conditional expression.
Check that an initialized allocator for a class-wide type does an accessibility check on the tag of the initializing object.

Check that an initialized allocator for an access type with a class-wide designated type does not create an object whose type is more local than the access type.

Eliminated overlong lines and tab characters.

Check that an uninitialized allocator for an access type with a class-wide designated type does not create an object whose type is more local than the access type.

Eliminated overlong lines and tab characters.

Corrected test objective and test description; the test code is unmodified.

Added error location codes.

Corrected comment @49 not to have the form of an error tag.

Eliminated overlong lines.

Corrected comment @49 not to have the form of an error tag.

When the selecting expression of a case statement is a name with a static nominal subtype and has a static predicate, the static predicate is disabled, and the value of the selecting expression does not satisfy the predicate, check that the others choice is returned, or, if there is no others choice, that Constraint_Error is raised.

Check that for loops iterate properly over subtypes with static predicates.

Eliminated overlong lines.

Check legality rules for aspect Default_Iterator.

Check legality rules for aspect Iterator_Element.

Check legality rules for array component iterators from Technical Corrigendum 1.

Check correct operation of array component iterators.

Check correct operation of array component iterators where the array has convention Fortran.
Eliminated violations of ACATS lexical rules.

Check legality rules for container element iterators (most of the checked rules are from Technical Corrigendum 1).

Check legality rules for generalized iterators from Technical Corrigendum 1.

Required after 2016-10-01

Check that Pre and Post are only allowed on subprograms and entries (and not on subprogram instances).

Added error location codes.

Required after 2016-10-01

Check that Pre'Class and Post'Class are only allowed on primitive subprograms of tagged types. (Normal tagged types and interfaces.)

Added error location codes.

Required after 2016-10-01

Check that Pre and Post are not allowed on subprograms acting as completions.

Added error location codes.

Required after 2016-10-01

Check that Pre'Class and Post'Class are not allowed on subprograms acting as completions.

Added error location codes.

Required after 2016-10-01

Check that Pre and Post are not allowed on abstract subprograms and null procedures.

Added error location codes.

Required after 2016-10-01

Check that the restrictions on Pre'Class and Post'Class expressions added by AI12-0113-1 are enforced.

Proc3 @88 and Proc4 @91 need to be abstract in order to avoid failing 6.1.1(18.1/4).

Required after 2016-10-01

Check that Pre'Class and Post'Class are not allowed on entries and protected subprograms of untagged task and protected types.

Required after 2016-10-01

Check various rules about the use of the Result attribute.
Check that F'Result is not allowed in the body of F.

Check that the Old attribute is not allowed in a subprogram or entry body, or in an accept statement.

Check the Old attribute cannot be used within a precondition expression nor in a generic specification outside of a postcondition expression.

Check that the prefix of an Old attribute cannot contain the Result attribute, another Old attribute, nor the index of a quantified expression.

For a discrete X, check that X'Old has the nominal subtype of X.

Check that preconditions and postconditions where both specific and class-wide contracts apply are checked properly.

Check that the correct expressions are evaluated for specific preconditions and postconditions, and that they are evaluated at the correct point.

Check that the correct expressions are evaluated for class-wide preconditions and postconditions, and that they are evaluated at the correct point.

Check that X'Old has the tag of X and not the tag of the nominal subtype of X (for Post).

Check that X'Old has the tag of X and not the tag of the nominal subtype of X (for Post'Class).

Changed error @35 to be optional, as the syntax error @33 is required (and the comment that only one error need be detected is bogus).

Changed error @41 to be optional, as the syntax error @33 is
required (and the comment that only one error need be detected is
bogus).

B640001 [New] VCS Label: A4_0E
[VCS Label A4_0C] Check that abstract nondispatching operations
do not participate in resolution.
[VCS Label A4_0E] Eliminated overlong lines and tab characters.

C640002 [New] VCS Label: A4_0E
[VCS Label A4_0C] Check that an abstract nondispatching operator
provides a way to eliminate an unwanted predefined operator.
[VCS Label A4_0E] Eliminate overlong lines.

B650005 [New] VCS Label: A4_0C
Check that the accessibility level of the return subtype
indication of an extended return statement cannot be statically
deeper that that of the function result if that result is
class-wide.

B650006 [New] VCS Label: A4_0C
Check that an extended return statement with a constant return
object cannot omit the initializing expression.

C650A02 [New] VCS Label: A4_0H
Check that the result of a function that returns a class-wide
tagged type has the tag of the return expression, if the
return_subtype_indication (if any) is class-wide.
Case A: limited types.

C650B01 [New] VCS Label: A4_0H
Check that the 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.

C650B02 [New] VCS Label: A4_0H
Check that the result of a function that returns a class-wide
tagged type has the tag of the return expression, if the
return_subtype_indication (if any) is class-wide.
Case B: non-limited types.

C650B03 [New] VCS Label: A4_0H
Check that the result of a function that returns a class-wide
tagged type has the tag of the return_subtype_indication if it is
a specific tagged type.

C650B04 [Pending New] New VCS Label: A4_OK
Required after 2016-10-01
Check that the result of a function that returns a class-wide
tagged type does an accessibility check on the tag of the result
object when the result expression is class-wide.

B730009 [Modified] VCS Label: A4_0E
Completion of Priv12 @131 violates 7.3(10.1/3).

B730010 [New] VCS Label: A4_0E
Check 7.3(10.1/3).

B732001 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check various resolution and legality rules for
aspect Type_Invariant.
[VCS Label A4_0E] Length @80 should have return type Natural.
Changed Is_Valid to Is_Clean @95, 97, 146 so that the inherited routine doesn't come into play.

C732001 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that assertion checks are made after assignments to and calls with actuals of appropriate view conversions.
[VCS Label A4_0E] Eliminated overlong lines.

C732002 [New] VCS Label: A4_0F
Check some differences between specific and class-wide type invariants.

C732A01 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that the failure of a Type_Invariant check raises Assertion_Error (constrained array).
[VCS Label A4_0E] Eliminated overlong lines.

C732A02 [New] VCS Label: A4_0F
Check that the failure of a Type_Invariant check raises Assertion_Error (part of unconstrained array).

C732B01 [New] VCS Label: A4_0F
Check that the failure of a Type_Invariant check raises Assertion_Error (discriminated record).

C732B02 [New] VCS Label: A4_0F
Check that the failure of a Type_Invariant check raises Assertion_Error (access-to-record).

B732C01 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that the aspect Type_Invariant can only be specified on private types, private extensions, and full types that complete one of the others.
[VCS Label A4_0E] Eliminate overlong lines.

C732C01 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that the failure of a Type_Invariant check raises Assertion_Error (tagged record).
[VCS Label A4_0E] Eliminated overlong lines.

B732C02 [New] VCS Label: A4_0E
[VCS Label A4_0D] Check that the aspect Type_Invariant'Class can only be specified on private types, private extensions, and interfaces.
[VCS Label A4_0E] Eliminate overlong lines.

B740003 [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
Required after 2016-10-01
Modified criteria @75 so that it doesn't look like an error tag to the grading tools; also fixed an overlong line.

B750A01 [Modified] VCS Label: A4_0H
The qualification @129 should be for Lim_Ext, not Lim_Rec.

B750A02 [Modified] VCS Label: A4_0H
The qualification @131 should be for Lim_Ext, not Lim_Rec.

B750A04 [New] VCS Label: A4_0H
Check that only legal expressions are allowed in the qualified expression of an allocator for a limited object.
Check that only legal expressions are allowed in the initialization expression of an extended return statement for a limited object.

Check that only legal expressions are allowed in the expression of a simple return statement for a function that returns a limited type.

Required after 2016-10-01

Corrected objective to mention “expression functions”.

Added error location codes.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the initialization expression of an object declaration of a limited type.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the default expression of a component declaration of a limited type.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the qualified expression of an initialized allocator for a limited object.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the initializing expression of an extended return statement that returns a limited type.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the expression of a simple return statement that returns a limited type.

Check that only legal expressions are allowed as the dependent expressions of a conditional expression used as the return expression of an expression function that returns a limited type.

Check that Ada.Finalization is declared pure.

Check that a controlled type can be declared within a subprogram.

Eliminate overlong lines.

Required after 2016-07-01
 Corrected one of the potential results for Subtest2(4) to be “Iafcf”, not “IafcF”.

C831001 [New] VCS Label: A4_0F
Check that overriding indicators do not change the dynamic effect of calls.

B831004 [New] VCS Label: A4_0F
Check that overriding indicators are properly enforced for types derived from a generic formal type. (Nested generic specification cases.)

B831005 [New] VCS Label: A4_0F
Check that overriding indicators are properly enforced for types derived from a generic formal type. (Library unit and body cases.)

B83F02C [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
Updated many optional error markers to match ACATS standards.

B840002 [New] VCS Label: A4_0F
Check a package use clause cannot denote a limited view or a non-package.

B840003 [New] VCS Label: A4_0F
Check a use all type clause does not make non-primitives or class-wide operations that aren't directly declare where a primitive operation would be visible.

B860001 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that the selecting expression of a case statement or case expression cannot be resolved if information from the case choices is needed to resolve it.

B95007B [Allowed Modification] New VCS Label: A4_0M Org VCS Label: A4_0
Required after 2016-10-01
Fixed error markers @62, 66, 70 to match ACATS standards.

BA12017 [New] VCS Label: A4_0E
Check that a limited with clause is not allowed on a descendant of the named package.
[VCS Label A4_0C] Eliminated overlong line.

BA15003 [New] VCS Label: A4_0C
Check that a library unit pragma must apply to a library unit.

CA21002 [New] VCS Label: A4_0E
Check that Pure and Preelaborate aspects can be specified with an object declared in another package.

CB30001 [New] VCS Label: A4_0B
Check that the optional message parameter is supported in a raise statement, and that the message can be retrieved with Exception_Message.

CB30002 [New] VCS Label: A4_0E
Check that the optional message parameter is supported in a raise expression, and that the message can be retrieved with Exception_Message.
[VCS Label A4_0E] Eliminated overlong lines and tab characters.
Check that the arguments of a pragma Assert have to have the correct types.

Added error location codes.

Check the usual operation of pragma Assert, including the optional message.

Check that the arguments of a pragma Assert cannot be reordered even if named notation is used.

Added error location codes.

Check that if the assertion policy is Ignore, none of the expressions of a pragma Assert are evaluated and no exception is raised by the pragma.

Check that a pragma Assertion_Policy is illegal if the policy is not recognized.

Added error location codes.

Check that procedure Ada.Assertions.Assert evaluates its arguments and raises Assertion_Error regardless of the assertion policy.

Eliminated overlong lines.

Eliminated overlong lines and tab characters.

Check that a generic unit with a tagged incomplete formal type can be instantiated with an incomplete view of a type from a limited withed package, and that operations in the instance can be used normally when the full view of the type is visible.

Added constructor function to generic, as Elem'Unchecked_Access is illegal when Elem is incomplete. Moved body of CC51010_2 so it compiles after another package on which it depends.

Test rules associated with formal subprogram defaults of null.

Check that calls on a formal abstract subprogram are dispatching.

Test rules associated with the controlling type of a formal
abstract subprogram.

BC60003 [New] VCS Label: A4_0G
Check that the actual subprogram for a formal abstract subprogram must be a dispatching operation of the controlling type.

BC60004 [New] VCS Label: A4_0G
Check that a call on a formal abstract subprogram must be dispatching.

CD11001 [Pending New] New VCS Label: A4_0L
Required after 2016-10-01
[VCS Label A4_0J] Check that appropriate entities are visible in an aspect specification.
[VCS Label A4_0L] Value 12 @121 should have been 52.

BD11002 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Check that entities in the private part of a package are not visible in aspect specifications given in the visible part.

CD30006 [Modified] VCS Label: A4_0E
Eliminated overlong lines and tab characters.

CD30007 [Modified] VCS Label: A4_0E
Eliminated overlong lines.

CD30008 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Renaming of previous test CD33001 (there is no subclause 13.3.3).
The test is unchanged other than renaming the units involved and fixing an overlong line.

CD30009 [Pending New] New VCS Label: A4_0J
Required after 2016-07-01
Renaming of previous test CD33002 (there is no subclause 13.3.3).
The test is unchanged other than renaming the units involved and fixing an overlong line.

CD33001 [Withdrawn] VCS Label: A4_0J
Renamed to CD30008, as there is no subclause 13.3.3.

CD33002 [Withdrawn] VCS Label: A4_0J
Renamed to CD30009, as there is no subclause 13.3.3.

Required after 2016-10-01
Moved error marker from @66 to @67 so it is on the incorrect construct.

BDB3A01 [New] VCS Label: A4_0E
[VCS Label A4_0B] Test resolution and legality rules for pragma and aspect Default_Storage_Pool.
[VCS Label A4_0E] Declaration of Pool @53 is missing the TC_Id discriminant.

CDB3A01 [New] VCS Label: A4_0E
[VCS Label A4_0B] Test that aspect Default_Storage_Pool changes the default storage pool for a generic instantiation.
[VCS Label A4_0E] Eliminated overlong line.
BDD2004 [Modified] VCS Label: A4_0E  
[VCS Label A4_0B] Error @229 is legal by AI12-0121-1.  
[VCS Label A4_0E] Eliminated overlong lines.  

BDD2005 [Modified] VCS Label: A4_0E  
[VCS Label A4_0B] Replaced test cases waiting on AI12-0106-1 (now approved) and made changes required by AI12-0121-1.  
[VCS Label A4_0E] Eliminated overlong lines.  

CXA4037 [New] VCS Label: A4_0F  

CXA5016 [New] VCS Label: A4_0E  
[VCS Label A4_0C] Check operation of attribute Machine_Rounding.  
[VCS Label A4_0E] Eliminated overlong lines.  

CXA021 [New] VCS Label: A4_0C  
Check file operations of Text_IO.Bounded_IO.  

CXA022 [New] VCS Label: A4_0C  
Check file operations of Text_IO.Unbounded_IO.  

CXAB002 [New] VCS Label: A4_0C  
Check file operations of Wide_Text_IO.Wide_Bounded_IO.  

CXAB003 [New] VCS Label: A4_0C  
Check file operations of Wide_Wide_Text_IO.Wide_Bounded_IO.  

CXAB004 [New] VCS Label: A4_0C  
Check file operations of Wide_Text_IO.Wide_Unbounded_IO.  

CXAB005 [New] VCS Label: A4_0C  
Check file operations of Wide_Wide_Text_IO.Wide_Wide_Unbounded_IO.  

CXAC008 [New] VCS Label: A4_0H  
[VCS Label A4_0C] Check that Stream_IO files opened with mode Out_File are not truncated by Close or Reset.  
[VCS Label A4_0D] Corrected required file size in subtest 8, and various failure messages.  
[VCS Label A4_0E] Eliminated overlong lines.  
[VCS Label A4_0H] Corrected a failure message.  

CXAG001 [New] VCS Label: A4_0E  
Check that Ada.Directories can be used to create and delete directories.  

CXAG002 [New] VCS Label: A4_0E  
Check that Ada.Directories.Hierarchical_File_Names exists and works as expected.  

CXAH001 [New] VCS Label: A4_0E  
[VCS Label A4_0D] Test that package Environment_Variables can be used to read environment variables.  
[VCS Label A4_0E] Added missing Applicability Criteria.  

CXAH002 [New] VCS Label: A4_0E  
[VCS Label A4_0D] Test that package Environment_Variables can be used to modify environment variables.  
[VCS Label A4_0E] Added missing Applicability Criteria.
CXAH003 [New] VCS Label: A4_0E
   [VCS Label A4_0D] Test the two parameter Value in package Environment_Variables.
   [VCS Label A4_0E] Added missing Applicability Criteria.

BXAI001 [New] VCS Label: A4_0B
   Test that package Hashed_Maps does not allow reverse iterators.

BXAI002 [New] VCS Label: A4_0B
   Test that package Hashed_Sets does not allow reverse iterators.

BXAI003 [New] VCS Label: A4_0B
   Test that package Bounded_Hashed_Maps does not allow reverse iterators.

BXAI004 [New] VCS Label: A4_0B
   Test that package Bounded_Hashed_Sets does not allow reverse iterators.

BXAI005 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Hashed_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

BXAI006 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Ordered_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

BXAI007 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Bounded_Hashed_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

BXAI008 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Bounded_Ordered_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

BXAI009 [New] VCS Label: A4_0B
   Test that package Indefinite_Hashed_Maps does not allow reverse iterators.

BXAI010 [New] VCS Label: A4_0B
   Test that package Indefinite_Hashed_Set does not allow reverse iterators.

BXAI011 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Indefinite_Hashed_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

BXAI012 [New] VCS Label: A4_0E
   [VCS Label A4_0B] Test that package Indefinite_Ordered_Sets does not allow direct writing of set elements.
   [VCS Label A4_0E] Eliminated overlong lines.

CXAI027 [Modified] VCS Label: A4_0B
   Corrected the name of the package that is reported as being tested.
Check B.3.3(8/2).

Check B.3.3(9/3).

Check B.3.3(11/2).

Check that rules for the application of aspect Unchecked_Union are enforced.

Check B.3.3(23/2).

Check B.3.3(24/2).

Check B.3.3(25/2).

Check B.3.3(26/2).

Check that a value of an unchecked union type (defined with pragma Unchecked_Union) can be passed to a C function.

[None.]

[None.]

[None.]

[None.]

[None.]

[None.]

[None.]
[None.]

===== Support =====

F416A00 [New] VCS Label: A4_0F
Foundation for tests B416A01, C416A01, and C416A02.

F552A00 [Modified] VCS Label: A4_0H
The Constant_Reference routine @237 and @388 needs to return a Bingo_Call value rather than a reference, (and then was renamed) in order to prevent an accessibility failure.

F611A00 [Pending New] New VCS Label: A4_0L
Required after 2016-10-01
Foundation for tests C611A01, C611A02, and C611A03.

F611B00 [Pending New] New VCS Label: A4_0L
Required after 2016-10-01
Foundation for tests C611B01 and C611B02.

F650B00 [New] VCS Label: A4_0H
Foundation for tests C650B01, C650B02, and C650B03.

F732A00 [New] VCS Label: A4_0F
  [VCS Label A4_0D] Foundation for test C732A01.
  [VCS Label A4_0E] Eliminated overlong lines.
  [VCS Label A4_0F] Added types and operations to support test C732A02. They are only needed for that test.

F732B00 [New] VCS Label: A4_0F
Foundation for tests C732B01 and C732B02.

F732C00 [New] VCS Label: A4_0E
  [VCS Label A4_0D] Foundation for tests B732C01, B732C02, and C732C01.
  [VCS Label A4_0E] Eliminated overlong lines.

FDB3A00 [New] VCS Label: A4_0E
  [VCS Label A4_0B] Foundation for test BDB3A01 and CDB3A01.
  [VCS Label A4_0E] Eliminated overlong line.

IMPDEF [Modified] VCS Label: A4_0J
  [VCS Label A4_0D] Added constants for the names of environment variables used by tests CXAH001, CXAH002, and CXAH003. Also removed obsolete .AW file constants (these are not used as test source has been provided in UTF-8 format since the release of ACATS 3.0). The change is only needed for those tests, it has no effect on any other tests.
  [VCS Label A4_0E] Added a constant and function used by tests CXAG001 and CXAG002. The change is only needed for those tests, it has no effect on any other tests.
  [VCS Label A4_0J] Added constants for the external names for the imported routines in tests CXB3023 and CXB3024. The change is only needed for those tests, it has no effect on any other test.

REPORT [Allowed Modification] New VCS Label: A4_0L Org VCS Label: A4_0
Required after 2016-07-01
  [VCS Label A4_0L] Revised to modern format, added optional event trace creation. (More discussion in introduction.) Note that the support file has been renamed to a modern name REPORT.A; the
older files REPSPEC.ADA and REPBODY.ADA are replaced. This changed file mainly is intended to support new capabilities in ACATS 4.1.

---

**How to get new and modified tests:**

The official home of the ACATS test suite is the ACAA web site. The site can be accessed at www.ada-auth.org.

The complete set of official new and modified tests are available in the Mod_4_0M.Zip and Tar files on the ACAA web site.

Individual new and modified tests can be retrieved from the ACATS VCS. Simply navigate to the appropriate directory (folder), and select the test you want. Select the revision you want (usually the most recent) and it will be retrieved for you. Tests are stored in directories that correspond to the test’s prefix. For instance, the test CD92001A will be found in the CD directory. You can also browse the differences between versions of a test. [In case of a conflict, the files in the ACATS VCS should be considered the 'official' versions.]

If you cannot get the files from the official site or you cannot get the files in a format appropriate for your processor, please contact your ACAL.

The baseline ACATS 4.0 is available in various formats on the ACAA web site.

**Summary lists:**

**Withdrawn Tests**

(Refer to the main list for details)

B43102A  
CD33001  
CD33002

---

**Allowed Modification Tests**

(Refer to the main list for details)

B38003D  2016-10-01  
B3A1001  2016-10-01  
B3A1002  2016-10-01  
B455002  2016-10-01  
B480003  2016-07-01 [VCS Label A4_0L]  
B740003  2016-10-01  
B83F02C  2016-10-01  
B95007B  2016-10-01  
BD4003A  2016-10-01  
C52101A  2016-10-01  
C760015  2016-07-01 [VCS Label A4_0L]  
REPORT  2016-07-01 [Only needed for ACATS 4.1]
Modified Tests

(Refer to the main list for details)

B730009
B750A01
B750A02
BDD2004
BDD2005
C324001
C324002
C415001
CC30003
CC30004
CD30005
CD30006
CD30007
CXAI027
F552A00
IMPDEF [VCS_Label_A4_0J]

New Tests

(Refer to the main list for details)

B393011
B3A1008
B3A1009
B3A1010
B3A2017
B415001
B415002
B416002
B416A01
B431005
B431006
B433003
B457002
B457003
B457004
B457005
B480002
B480003 [VCS_Label_A4_0E]
B551001
B551002
B552A04
B552A05
B640001
B650005
B650006
B730010
B732001
B732C01
B732C02
B750A04
B750A05
B750A06
B750A08
Pending New Tests

(Refer to the main list for details)

B457006  2016-07-01
B457007  2016-07-01
B611001  2016-10-01
B611002  2016-10-01
B611003  2016-10-01
B611004  2016-10-01
B611005  2016-10-01
B611006  2016-10-01
B611007  2016-10-01
B611008  2016-10-01
B611009  2016-10-01
B611010  2016-10-01
B611011  2016-10-01
B611012  2016-10-01
B611013  2016-10-01
B750A07  2016-10-01
B860001  2016-07-01
BB42001  2016-10-01
BB42002  2016-10-01
<table>
<thead>
<tr>
<th>Code</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB42003</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>BD11002</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C3A0030</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C416A02</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C457005</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C457006</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C457007</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C480001</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C540003</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>C611A01</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C611A02</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C611A03</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C611B01</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C611B02</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>C650B04</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>CB42001</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>CB42002</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>CB42003</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>CD11001</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>CD30008</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>CD30009</td>
<td>2016-07-01</td>
</tr>
<tr>
<td>CXB3023</td>
<td>2017-01-01</td>
</tr>
<tr>
<td>CXB3024</td>
<td>2017-01-01</td>
</tr>
<tr>
<td>F611A00</td>
<td>2016-10-01</td>
</tr>
<tr>
<td>F611B00</td>
<td>2016-10-01</td>
</tr>
</tbody>
</table>