# Table of Contents

Table of Contents ............................................................................................................i  

Section 1: Introduction ..................................................................................................5  
  1.1 ACATS Purpose .................................................................................................5  
  1.2 ACATS Coverage of Ada ..................................................................................6  

Section 2: Changes for ACATS 4.0 ...............................................................................9  

Section 3: Test Objectives and Coverage .....................................................................11  
  3.1 Test Objectives and Rules ................................................................................11  
  3.2 Coverage of the Ada Standard ...........................................................................11  
      3.2.1 Coverage Documents ................................................................................11  
      3.2.2 General Coverage Guidelines .................................................................12  
      3.2.3 Coverage Guidelines for Specific Rule Categories ..................................13  

Section 4: Configuration Information ..........................................................................17  
  4.1 Structure ..........................................................................................................17  
      4.1.1 Physical Organization ..............................................................................17  
      4.1.2 Logical Organization ...............................................................................18  
      4.1.3 Legacy Tests ............................................................................................18  
      4.1.4 Test Foundation Code .............................................................................19  
      4.1.5 Special Core Tests ..................................................................................19  
      4.1.6 Foreign Language Code .........................................................................19  
  4.2 Test Classes .......................................................................................................20  
      4.2.1 Class A ......................................................................................................20  
      4.2.2 Class B ......................................................................................................20  
      4.2.3 Class C ......................................................................................................20  
      4.2.4 Class D ......................................................................................................21  
      4.2.5 Class E ......................................................................................................21  
      4.2.6 Class L ......................................................................................................21  
      4.2.7 Foundation Code .....................................................................................22  
      4.2.8 Specialized Needs Annex Tests ...............................................................22  
  4.3 Naming Convention ............................................................................................22  
      4.3.1 Legacy Naming ........................................................................................22  
      4.3.2 Modern Naming .......................................................................................24  
      4.3.3 Multiple File Tests ...................................................................................26  
  4.4 Test Program Format ..........................................................................................26  
  4.5 General Standards .............................................................................................27  
  4.6 Test Structure ....................................................................................................28  
  4.7 Delivery Directory Structure .............................................................................29  
  4.8 File Format ........................................................................................................30  

Section 5: Using the ACATS .....................................................................................31  
  5.1 Installation of the ACATS Test Suite ................................................................32  
      5.1.1 Contents of the ACATS Delivery ............................................................32  
      5.1.2 Guide to Decompressing Files .................................................................33  
      5.1.3 Files With Non-Graphic Characters ......................................................34  
  5.2 Tailoring the ACATS Test Suite ........................................................................36  
      5.2.1 ImpDef Customization ............................................................................36
# Table of Contents

## 5.2.2 Macro Defs Customization ................................................................. 37
## 5.2.3 Packages SPPRT13 and FCNDECL .................................................. 37
## 5.2.4 Modification of Package REPORT .................................................... 38
## 5.2.5 Allowed Test Modifications ............................................................... 38
## 5.3 Processing the Support Files ............................................................... 39
### 5.3.1 Support Files .................................................................................. 39
### 5.3.2 "CZ" Acceptance Tests ................................................................. 39
## 5.4 Establishing Command Scripts ............................................................ 40
### 5.4.1 Command Scripts .......................................................................... 40
### 5.4.2 Dependencies ............................................................................... 40
## 5.5 Processing ACATS Tests ..................................................................... 40
### 5.5.1 Required Tests ............................................................................ 41
### 5.5.2 Test Partitions ............................................................................... 41
### 5.5.3 Bundling Test Programs ............................................................... 41
### 5.5.4 Processing that may be Omitted .................................................... 41
### 5.5.5 Tests with Special Processing Requirements ................................. 42
### 5.5.6 Focus on Specific Areas ................................................................. 47
## 5.6 Grading Test Results .......................................................................... 47
### 5.6.1 Expected Results for Executable Tests .......................................... 47
### 5.6.2 Expected Results for Class B Tests .............................................. 48
### 5.6.3 Expected Results for Class L Tests .............................................. 49
### 5.6.4 Inapplicable Tests ........................................................................ 49
### 5.6.5 Withdrawn Tests .......................................................................... 49
## 5.7 Addressing Problems or Issues ........................................................... 50
### 5.7.1 Typical Issues ............................................................................... 50
### 5.7.2 Deviation from Expected Results - Petition & Review .................. 50
## 5.8 Reprocessing and Regrading ............................................................... 51

## Annex A: Version Description ................................................................. 53
### A.1 Core Test Files ................................................................................ 53
### A.2 Specialized Needs Annex Test Files ............................................... 75
### A.3 Foundation Code Files ..................................................................... 77
### A.4 Documentation Files ........................................................................ 77
### A.5 Other Files ..................................................................................... 78
#### A.5.1 List of ACATS 4.0 Files ............................................................... 78
#### A.5.2 Support Units Referenced by Many Tests ................................. 78
#### A.5.3 Preprocessing Tools and Data .................................................... 78
#### A.5.4 Tests for Reporting Code .......................................................... 78
### A.6 Tests With Special Requirements .................................................... 79
### A.7 Test Files Added Since ACATS 3.1 ............................................... 79
### A.8 Documentation Files Added Since ACATS 3.1 ............................... 80
### A.9 Test Files Modified Since ACATS 3.1 .......................................... 80
### A.10 Support Files Modified Since ACATS 3.1 ................................. 81
### A.11 Documentation Files Modified Since ACATS 3.1 ....................... 81
### A.12 Test Files Deleted Since ACATS 3.1 .......................................... 82
### A.13 Documentation Files Deleted Since ACATS 3.1 ......................... 82

## Annex B: Parameterization Files ............................................................. 83
### B.1 Macro Substitution File ................................................................. 83
### B.2 Macro Substitution Tests ............................................................... 84
### B.3 Package ImpDef and Its Children .................................................. 84
### Annex C: Results of CZ Tests

- C.1 Sample Output From CZ0004 ................................................................. 87
- C.2 Sample Output From CZ1101A ................................................................. 88
- C.3 Sample Output From CZ1102A ................................................................. 89
- C.4 Sample Output From CZ1103A ................................................................. 89
  - C.4.1 Output When External Files Are Supported ........................................ 89
  - C.4.2 Output When External Files Are Not Supported ................................... 90

### Annex D: Test Applicability Criteria

- D.1 Compile-Time Inapplicability ................................................................. 91
  - D.1.1 Type Short_Integer ............................................................................ 91
  - D.1.2 Type Long_Integer ............................................................................ 91
  - D.1.3 Other Predefined Integer Types ......................................................... 92
  - D.1.4 Fixed Point Restrictions ..................................................................... 92
  - D.1.5 Non-binary Values of 'Small ............................................................... 92
  - D.1.6 Compiler Rejection of Supposedly Static Expression ......................... 92
  - D.1.7 Machine Code Insertions .................................................................... 92
  - D.1.8 Illegal External File Names ............................................................... 92
  - D.1.9 Decimal Types .................................................................................... 93
  - D.1.10 Instantiation of Sequential_IO with indefinite types ...................... 93
  - D.1.11 Special Handling Tests ...................................................................... 93
- D.2 Reported Inapplicability ......................................................................... 93
  - D.2.1 Value of Machine_Overflows is False ............................................... 93
  - D.2.2 System.Max_Digits ............................................................................ 93
  - D.2.3 Floating Point Overflow ..................................................................... 93
  - D.2.4 Type Duration .................................................................................... 94
  - D.2.5 Text Files (Non-supported Features) ................................................. 94
  - D.2.6 Text Files (Supported Features) ......................................................... 97
  - D.2.7 Sequential Files (Non-supported Features) ......................................... 97
  - D.2.8 Sequential Files (Supported Features) ................................................. 98
  - D.2.9 Direct Files (Non-supported Features) .............................................. 99
  - D.2.10 Direct Files (Supported Features) ...................................................... 100
  - D.2.11 Stream Files (Non-supported Features) ............................................ 101
  - D.2.12 Wide Text Files (Non-supported Features) ....................................... 101
  - D.2.13 File I/O Tests .................................................................................. 101
  - D.2.14 Memory for Allocated Objects ....................................................... 103
  - D.2.15 Task Attributes .............................................................................. 103
  - D.2.16 Reserved Interrupts .......................................................................... 103
  - D.2.17 Multiprocessor Systems .................................................................... 103
  - D.2.18 Non-binary Machine Radix .............................................................. 104

### Annex E: Guidelines for Test Development ............................................ 105

### Annex F: Definitions ............................................................................... 109

### References ............................................................................................. 113

### Index ....................................................................................................... 115
Section 1: Introduction

The Ada Conformity Assessment Test Suite (ACATS) provides the official tests used to check conformity of an Ada implementation with the Ada programming language standard (ANSI/ISO/IEC 8652:1995 and later corrigenda). The ACATS User's Guide is part of the ACATS and is distributed with the test programs and testing support packages. It explains the contents and use of the test suite.

The ACATS is an important part of the conformity assessment process described in ISO/IEC-18009, Ada: Conformity of a Language Processor [ISO99]. This standard provides a framework for testing language processors, providing a stable and reproducible basis for testing. The Ada Resource Association (ARA) has sponsored an instantiation of that process since October 1998. The process is managed by the Ada Conformity Assessment Authority (ACAA).

Prior to the ISO standard, the U.S. Department of Defense sponsored a similar conformity assessment process under the Ada Joint Program Office (AJPO). The test suite for that process was known as the Ada Compiler Validation Capability (ACVC). The AJPO developed ACVC versions based on ANSI/MIL-STD-1815A-1983, ISO/8652:1987 (Ada 83), which were numbered 1.x where x ranged from 1 to 11. It later developed ACVC versions based on ANSI/ISO/IEC 8652:1995 ([Ada95]), numbered 2.0, 2.0.1, 2.1, and 2.2.

When the ACAA took over Ada conformity assessment, it adopted the ACVC as the basis for its test suite. The ACAA determined to continue to use the same version numbering for the test suite in order to avoid confusion. The version of the ACVC current at the time (2.1) was initially used as ACATS 2.1. Later, the already developed but unreleased ACVC 2.2 was released and used as ACATS 2.2. The ACAA later released ACATS 2.3, ACATS 2.4, ACATS 2.5, and then ACATS 2.6 to include maintenance changes and a few new tests.

In 2007, the ACAA developed ACATS version 3.0 to check for conformity to new language features defined in ISO/IEC 8652:1995/AMD 1:2007 ([Amend1]), as well as test programs to check for conformity to language features defined in earlier versions of Ada, including [Ada95] and [Ada83]. The ACAA later released ACATS 3.1 to improve the coverage and correctness of the ACATS for features defined in [Amend1].

This version of the ACATS is 4.0. It was developed in 2014 to test the enhancements and changes of the third edition of the Ada Standard, ISO/IEC 8652:2012 ([Ada2012]). Subsequent maintenance or enhancement versions of the suite, if they are required, will be numbered 4.1, etc.

The ACATS User's Guide describes the set of ACATS tests and how they are to be used in preparation for conformity assessment. The formal procedures for conformity assessment are described in [Pro31], and the rules in that document govern all conformity assessments, notwithstanding anything in this document that may be interpreted differently. Moreover, this guide does not discuss specific requirements on processing of the ACATS test suite, or submission and grading of results that an Ada Conformity Assessment Laboratory (ACAL) may impose.

The User's Guide is intended to be used by compiler implementers, software developers who maintain a version of the ACATS as a quality control or software acceptance tool, and third-party testers (e.g., Ada Conformity Assessment Laboratories).

Section 2 of the User's Guide for ACATS 4.0 summarizes the changes between ACATS 3.1 and ACATS 4.0. Section 3 describes test objectives and their relationship to ACATS tests and to the rules of the Ada Standards documents. Section 4 describes the configuration of the ACATS, including a description of the ACATS software and delivery files. Section 5 provides step-by-step instructions for installing and using
the test programs and test support packages, and for grading test results. The appendices include other information that characterizes the ACATS 4.0 release, along with information on test construction.

Refer to Annex F and Section 5.6 for the definition of an acceptable result and the rules for grading ACATS 4.0 test program results. Section 5.7.2 provides instructions for submitting a petition against a test program if a user believes that a deviation from the acceptable results for a given test program is in fact conforming behavior.

The ACATS test suite is available from any ACAL and from the ACAA web site. See http://www.ada-auth.org/acats.html.

1.1 ACATS Purpose

The purpose of the ACATS is to check whether an Ada compilation system is a conforming implementation, i.e., whether it produces an acceptable result for every applicable test.

A fundamental goal of conformity assessment (validation) is to promote Ada software portability by ensuring consistent processing of Ada language features as prescribed by the Ada Standard documents ([Ada2012] and any future corrigendum documents). ACATS tests use language features in contexts and idioms expected in production software. While they exercise a wide range of language feature uses, they do not and cannot include examples of all possible feature uses and interactions.

It is important to recognize that the ACATS tests do not guarantee compiler correctness. A compilation system that correctly processes the ACATS tests is not thereby deemed error-free, nor is it thereby deemed capable of correctly processing all software that is submitted to it.

The ACATS tests do not test the quality of an Ada implementation. In particular, ACATS tests do not check or report performance parameters (e.g., compile-time capacities or run-time speed). They do not check or report for characteristics such as the presence and effectiveness of compiler optimization. They do not investigate or report compiler or implementation choices in cases where the standard allows options.

1.2 ACATS Coverage of Ada

The ACATS needs to test as many Ada language rules as possible in order to meet the goal of enhancing Ada software portability. After all, a rule that is not tested is far more likely to be incorrectly implemented than one that is tested.

Therefore the ACATS strives for complete coverage of the standard. Complete coverage means that every rule in the Ada standard has one or more associated tests that ensure that the rule is implemented properly.

Complete coverage is especially important for legality rules and runtime checks. It is easy for implementers to miss these rules, as their compiler may do something useful in the absence of the checks. But allowing such incorrect code can be a major portability problem when a program is moved to a compiler (including a later version of the same compiler) that properly implements the rules and checks.

Of course, complete coverage does not mean that every sentence in the Ada standard has an associated test. There are many lines in the standard that are not rules at all, such as notes and examples. There are also many lines in the standard that are not testable; documentation requirements are but one example. There are also rules in the standard which could be tested, but such tests would be outside of the purpose of the ACATS. Rules which express options for Ada implementations (such as permissions or advice) are in this category (as the ACATS does not test implementation quality).
It should be obvious that the importance of testing particular rules in the Ada standard vary widely. In part, that's because the various Ada features themselves have widely varying importance. (For example, looking at three features that ACATS 4.0 could have tested, proper operation of if expressions is more important than correct functioning of record equality, which in turn is more important than run-time accessibility checks for coextensions.) But it's also because of the differing nature of the rules. For instance, testing of syntax rules is not very important in most cases, as correct syntax will be checked by tests for other rules, and tests for incorrect syntax require guessing what error a compiler might make in interpreting syntax. Unless an error seems particularly likely (as might happen from some irregularity in the syntax), such guessing is unlikely to find actual errors. This makes it impossible for there to be a definitive set of rules that need to be tested to accomplish complete coverage, as the decision as to whether a particular rule is usefully testable can be a judgement call.

In any case, complete coverage is a goal for the ACATS. However, it is not expected to be ever be achieved. As the ACATS gets nearer to the goal, the value of additional tests will drop (since higher value tests get created earlier), and at some point, the cost of creating and processing new tests would outweigh their value.

Since exactly which rules need to be tested to accomplish complete coverage will always be a judgement call, the test coverage analysis of the ACATS is a set of living documents, which will be updated with new information and tests with each new ACATS version.

Coverage testing will generally not test combinations of features, so problems that only manifest themselves in such combinations will not be detected. Tests designed primarily to cover language rules are most useful to prevent gross errors in implementations (such as forgetting to implement checks or features). As such, the ACATS also supplements those tests with tests written to emulate the patterns of use of Ada features. Such tests provide tests of common combinations of features, and ensure that common idioms are implemented properly.

A detailed description of how coverage is determined for the Ada standard can be found in Section 3.2.
Section 2: Changes for ACATS 4.0

Version 4.0 of the ACATS updates version 3.1 with additional tests for features defined in [Ada2012]. It includes more than 110 new tests to check features including Ada.Containers, aspect specification notation, conditional expressions, pre- and postconditions, subtype predicates, new iterator forms, user-defined references and indexing, and formal incomplete types.

In addition, 35 additional tests were corrected or removed in to reflect changes in Ada reflected by [Ada2012], as well as in response to test disputes and ARG issue resolutions.

See Annex A, “Version Description” for lists of added, deleted and modified tests.
**Section 3: Test Objectives and Coverage**

Each ACATS test tests one or more *test objectives*. Test objectives ought to be relatable to rules given in the Ada Standard documents ([Ada2012]).

The test objectives of modern ACATS tests are gathered into a *Test Objectives Document*. Each modern test is listed along with the objectives that it tests (legacy tests aren't included as they use a different form of objective). This document provides easier searching for particular objectives than a raw search of all of the test files.

This section provides information on how new test objectives are constructed and how the adequacy of objectives is checked.

### 3.1 Test Objectives and Rules

A test objective should relate as directly as possible to the rules of Ada. Ideally, a test objective will test an individual rule of Ada. *Rule* means whatever set of sentences makes up a testable statement. A rule may be as small as a single line, or spread across multiple paragraphs of the standards.

For instance, [Ada2012] contains the rule

> The implicit declaration of the limited view of a library package forms an (implicit) compilation unit whose *context_clause* is empty.

in subclause 10.1.1. The corresponding test objective is

> Check that the context clause of a limited view is empty.

However, virtually all rules require the use of other rules (such as those for definitions of terms) in order to be testable. For instance, testing the above rule needs the definition of terms like "limited view" and "context_clause" (among others) in order for a test to be constructed.

As such, a test objective will generally apply to multiple Ada rules. Typically, it will be recorded for a specific rule. For the purposes of deciding if a rule is appropriately tested, it is necessary to study the ACATS coverage documents (see 3.2.1).

### 3.2 Coverage of the Ada Standard

The Ada Standard documents [Ada2012] include many thousands of rules. The ACATS Coverage Documents (see 3.2.1) record the mapping of test objectives and tests to Ada rules, making it possible to determine if individual rules are adequately tested. The following clauses discuss these documents and the rules by which they are constructed.

#### 3.2.1 Coverage Documents

ACVC 2.0 originated coverage documentation for what is now the ACATS. That coverage document mapped tests to paragraph numbers in the [Ada95] Standard. Over time, a number of deficiencies in that approach have been identified:

- Where a paragraph contains multiple rules, it is not possible to tell if all of the rules are tested. Indeed, a number of instances where important rules were untested have been identified.
- Since there is not an indication of the type of rule represented by a paragraph, it is not obvious whether the rule is appropriately tested. For instance, a Legality Rule tested only with C-Tests is not appropriately tested (see 3.2.3 for more on this topic).
• Rules not modified in [Ada95] were not considered. However, sometimes additional test objectives are needed for existing rules because of other changes to the language. Moreover, there is no determination of whether the legacy tests adequately tested the original rule.

• The plaintext coverage document is not easily processed by tools. In particular, paragraphs that are covered are indicated only by the test name(s) that cover the paragraph; there is no common string for a tool to count. As such, it is difficult to get metrics about coverage from the document, and in particular to quantify progress between versions.

For these reasons, the development of new and much more detailed coverage documentation was initiated for ACATS 3.0. The new documentation considers each sentence of the standard individually if necessary, listing the test objectives for each sentence, and documenting why no objectives are needed if none are provided.

It also lists tests for each objective (including Legacy tests), testing priorities for untested objectives, notes on each objective (including any untested cases), and any additional notes needed.

There is also a summary document that summarizes totals for important metrics for the coverage documents as a whole.

The documents are maintained as spreadsheets (in order to make calculation and extraction of metrics easy). Because spreadsheet programs may be unfamiliar to many readers, and compatibility of spreadsheet programs is incomplete, these documents are provided in Adobe Portable Document Format (PDF). (The spreadsheets are available on request from the ACAA Technical Agent, agent@ada-auth.org.)

For ACATS 4.0, the following clauses of the Ada Standard documents are included in the new coverage documents:

3.2.4
3.9.3 through 3.10.1
4.1.3 through 4.4
4.5.7 through 4.5.8
5.4 through 5.5.2
6.5 through 6.8
7.4 through 7.6.1
8.3.1 through 8.5.3
Section 10 (10 through 10.2.1)
13.1.1

These clauses were selected because of their importance in rule changes in either [Ada2012] or [Amend1].

The original coverage document is also included with the ACATS for those portions of the Ada Standard documents that have not yet had new coverage documents constructed. This document is of primary use for features of [Ada95], as it does not cover Legacy tests nor any features added or modified by either [Ada2012] or [Amend1]. As noted above, it is being replaced, so it will disappear completely in a future version of the ACATS.

3.2.2 General Coverage Guidelines

As noted in Section 1.2, the ACATS strives for complete coverage of the Ada standard. However, complete coverage does not mean that every sentence in the standard has an associated test objective and test. Not all rules are testable; one important way to determine testability of a rule is to check its category. The next clause (3.2.3) will discuss how the category of a rule affects its mapping to test objectives.

There are also additional considerations that don't apply to specific rule categories.

Text sometimes contains definitions (especially in Static Semantics rules). Definitions are hard to handle, because they usually are not testable by themselves. The definition has to be used in another rule to make
them visible. For instance, a categorization rule such as "something is either this, that, or fuzzy" cannot be
tested by itself. "Something" has to be used in some other rule in order for the categorization to matter.
That being the case, it usually makes sense to test the definition as part of the other rule(s). Such rules are
marked as having a kind of "Widely Used" or "Subpart" in the coverage documentation. "Subpart" rules
are tested as part of specific other objectives, while "Widely Used" rules are thought to be tested by many
tests indirectly (and no attempt is made to verify that). In some cases, especially those that otherwise
would have a combinatorial explosion, it may make more sense to test the definition as directly as
possible (in which case objectives will be assigned to the definition).

Text sometimes includes sentences marked as redundant by the AARM. Such sentences should be given
normatively elsewhere, and the testing should be done in the place where they are normatively given. The
AARM often will indicate this place.

Combinational explosion is always a problem for visibility rules. Rules that define the scope or visibility
of something should always be tested in place (even though other rules will need to be used to accomplish
that).

### 3.2.3 Coverage Guidelines for Specific Rule Categories

The Ada standard is divided into various categories of rules. These rule categories have different
requirements and thus differ in what is the appropriate testing philosophy. This translates into different
types of test objectives for each rule category.

Uncategorized text:
This is generally introductory text, and does not require any testing. This text sometimes
includes definitions; these are handled as described in 3.2.2, “General Coverage Guidelines”.

Syntax:
Testing that legal syntax is implemented is accomplished by all of the other kinds of tests,
especially by the usage-oriented tests. Generally, tests for illegal syntax have a very low
devalue. Most compilers use a syntax that is similar to that of the standard, and many use
parsers generated by tools, and thus the likelihood of errors is low. Moreover, the number of
possible bugs is essentially unlimited. There are three cases where testing is warranted: First,
rules that are given in text form rather than BNF are treated as legality rules (see below).
Second, the Ada syntax grammar is not directly usable by either of the common technologies
for parsers (LL/recursive descent, and LR/LALR). In cases where the grammar used by a
compiler must necessarily be more general than that given in the standard, the additional
requirements are treated as legality rules. Finally, an irregularity in the syntax rules can
suggest an obvious (but incorrect) syntax extension. It can warrant a test that such an
extension is not implemented. For instance, most places in Ada that declare an object allow
the use of subtype indications, but subprogram profiles only allow subtype marks. It should
be tested that subtype indications are not allowed when declaring parameters, even though
this is a syntax rule.

Name Resolution Rules:
Name resolution rules should usually have a B-Test that checks that too much information is
not used for resolution. (For instance, whether a type is limited or non-limited should
generally not be used. Another example is whether an access type is pool-specific or general
access usually should not be used.) It also may have a C-Test to check that legal cases are
allowed; but this objective is often handled by tests for other test objectives (if that's the case,
a note should point out where they are tested). One common case where a C-Test is needed
occurs when a name resolution rule uses some property for resolution. In this case, a C-Test
that the property can be used to resolve an overloaded function call is needed. (For instance,
an if statement condition must be of a Boolean type; it should be tested that a function
returning some other type is ignored when resolving a condition.)
Legality Rules:
Legality rules should almost always have an associated B-Test that checks that illegal cases are detected. They may also have a C-Test to check that legal cases are allowed; again, this is often handled by tests for other test objectives. (They also can be covered in the B-Test with "OK" cases.)

Static Semantics:
Many Static Semantic rules are definitions, and as such can be handled as described in 3.2.2, “General Coverage Guidelines”.

Many other static semantic rules are miscategorized by the standard. For instance, most of the library packages are defined as static semantics, even though most of the rules are really dynamic. These should be tested as appropriate for the correct category.

Remaining rules should be tested as legality rules (arguably, these too are miscategorized).

Post-Compilation Rules:
A post-compilation rule should always have L-Tests to check that the check is made in illegal cases. These checks are especially likely to be omitted (because they're often complex to implement), so these tests should have a high priority. In some cases, a C-Test may be needed to ensure that legal cases work and the check does not go too far. Often that will be covered by other tests, so a separate test is not needed (a note should be provided to show where it is covered).

Dynamic Semantics:
Dynamic semantics rules can be divided into two sub-categories.

First, these rules can define run-time checks. These checks should have C-Tests that verify that the check is made and the appropriate exception is raised. Tests like this have to be carefully constructed to avoid running afoul of 11.6 permissions.

Otherwise, these rules can define the normal operation of a construct. These should have usage-oriented C-Tests. By making these tests usage-oriented, the ACATS tests the normal usage of these features, not unusual corner-cases.

One special case: rules that say the execution of something "has no effect" can't be usefully tested. Testing that nothing happens is not interesting. And testing for the absence of an effect requires guessing incorrect effects that might have occurred (which is unlikely to actually detect anything), and thus any such test would have a very low value.

Bounded (Run-Time) Errors:
These usually require C-Tests, but not always. If the bound includes "work normally", then the bounded error is not usefully testable, since the ACATS does not test implementation choices. Whether something works or raises Program_Error is not interesting. In other cases, however, since there is a bound on the behavior, it can be useful to ensure that one of the prescribed results happens rather than havoc.

Erroneous Execution:
This is not testable. Since the language allows anything to happen, there is no useful information to be gained from including this in a test. Indeed, it's important to avoid any such case in an executable test.

Implementation Requirements:
These usually are dynamic requirements that should be tested like the appropriate dynamic semantics rules. Otherwise, they should be tested like a legality rule.

Implementation Permissions:
Usually permissions are not testable. Since the ACATS is not testing the quality of implementations, the choices made by an implementation are not appropriate things to test.
So a test should be created only if there is value in testing for implementations exceeding the permission.

Implementation Advice:
Advice is usually not testable. Again, since the ACATS is not testing quality of implementations, the choices made by an implementation are not appropriate things to test. Moreover, advice need not be followed, and other implementation choices can be made. Usually, if there are bounds to what is acceptable, they are covered by other rules, and thus the tests would be there.

Documentation Requirements and Metrics:
No tests are required for documentation requirements. Documentation existence could be part of the conformity assessment process, but in any case it is outside of the scope of the ACATS (which is testing the implementation, not the documentation).

Notes and Examples:
These are non-normative text, and do not require any testing.
Section 4: Configuration Information

This section describes the physical and logical structure of the ACATS delivery, and it describes the test classes, naming conventions used, test program format, test structure, delivery structure, and file format.

ACATS 4.0 is an update of ACATS 3.1, and has a similar delivery structure. The support tools are unchanged, except for updating header comments and version identification.

The test suite does not provide tools or scripts that can be used to manage complete test processing, since such tools are normally specific to particular implementations and host systems.

4.1 Structure

The ACATS 4.0 test software includes test code that exercises specific Ada features, foundation code (used by multiple tests), support code (used to generate test results), and tool code (used to build tools necessary to customize ACATS tests). The suite includes tests for the core language and tests for the Specialized Needs Annexes. The following table summarizes the number of tests and files in the ACATS suite.

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Core Tests</th>
<th>SNA Tests</th>
<th>Foundations</th>
<th>Docs</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Files</td>
<td>4787</td>
<td>4287</td>
<td>250</td>
<td>54</td>
<td>174</td>
<td>22</td>
</tr>
<tr>
<td>Number of Tests</td>
<td>3910</td>
<td>3718</td>
<td>192</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Others consists of:

1. List of all files
2. Code that is referenced by tests
3. Code and data used for preprocessing tests to insert implementation specific information
4. Test routines for reporting code ("CZ" tests)

The delivery structure of the test suite is described in Section 4.7.

4.1.1 Physical Organization

The preceding table summarizes the number of files that compose ACATS 4.0. In addition to files containing test code proper, the ACATS 4.0 test suite includes various support files.

Note that the number of files containing test code is larger than the number of tests in the ACATS suite because some tests use code included in separate files.

A file name consists of a name plus an extension. Multiple files that contain code used by a single test have related names. File names are the same as that of the test contained in the file when possible. File names conform to MS-DOS naming conventions; therefore they may be shorter than the software name because of file name length restrictions (e.g., enumcheck rather than enumcheck). File (and test) names follow conventions that indicate their function in the test suite; naming conventions are explained in...
Section 4.3. The files are organized into distinct directories and subdirectories based on their function in the test suite. The directory organization is explained in Section 4.7.

The ACATS is available to the general public from an ACAL or on the Internet. Links to the ACATS distribution can be found on the ACAA’s ACATS page:


Note that the ACATS files are available in both compressed Unix tar and DOS zipped formats. Section 5.1.2 provides a discussion of techniques to convert these files to a usable format.

4.1.2 Logical Organization

The table summarizes the number of tests that check the conformance of an Ada implementation to the core language and conformance to the Specialized Needs Annexes of Ada.

Core tests apply to all implementations. Specialized Needs Annex tests are not required for any implementation. Tests for a given Specialized Needs Annex may be processed by implementations that claim implementation of that annex.

In general, no test result depends on the processing or the result of any other test. Exceptions are noted in Section 5.4.2. No annex test depends on the implementation of any other annex, except possibly in cases where one annex specifically depends on another in Ada (e.g., no test for the Information Processing Annex uses features from any other annex, however Real Time Annex and Distributed Processing tests may depend on Systems Programming Annex features). (There is a single exception to this rule: see Section 5.5.5.3.) Annex tests may use any core feature.

Tests may be created from one or more compilation units. If a test consists of a single compilation unit (a main subprogram only), the test code will be contained in a single file. Tests built from more than one compilation unit may require multiple files. Moreover, some compilation units, called foundation code, may be used by more than one test. Even in these cases, the resulting tests are strictly independent: if test A and test B use the same foundation code, the results of processing (and running, if appropriate) A have no effect on the results of processing (and running, if appropriate) B. Foundation code is more fully explained in Section 4.1.4.

Tests are named using conventions that provide (limited) information about the test. The test naming conventions are explained in Section 4.3. Each test belongs to a single test class that indicates whether it is or is not an executable test. Test classes are explained in Section 4.2.

In addition to test code and foundation code, there is code on which many or all of the executable tests in the suite depend (e.g., package Report, package ImpDef, package TCTouch). Some of this code must be customized to each implementation. There is also code that must be used to build support tools used to customize the suite of tests to an implementation. The customization process is described in Section 5.2.

4.1.3 Legacy Tests

Legacy tests are tests that were included in ACVC 1.12 that have been incorporated into later ACVC and ACATS versions. These tests check only language features that are common to all versions of Ada. The vast majority of these tests came unmodified from the ACVC 1.12 suite. Some tests were modified to check for the correct implementation of Ada rules in cases where language rules changed from [Ada83].

Unlike more modern tests, legacy tests use an ALL CAPITALS programming style which is unusual and hard to read (4.5). They also use the naming conventions of early ACVC versions (see 4.3.1). Modern tests (those created for the ACATS more recently than ACVC 1.12) use a more typical Mixed Case programming style (see 4.5 and use a more flexible naming scheme (see 4.3.2).
4.1.4 Test Foundation Code

Some tests use foundation code. Foundation code is reusable across multiple tests that are themselves independent of each other. It is intended to be compiled and included in an environment as part of the compilation process of a test. If the test is executable, the foundation code must be bound with all other code for the test prior to execution.

Foundation code is always expected to compile successfully; it is never expected to be run by itself. Foundation code is not, in and of itself, a test, and is therefore not characterized by a test class (see 4.2). One may think of it as providing some utility definitions and routines to a number of different tests. Names of foundation units (and therefore names of files containing foundation code) are distinguished as described in Naming Convention, Section 4.3.

4.1.5 Special Core Tests

This section identifies tests that appear in the Core (since their requirements are enunciated there) but that may be graded as non-supported for implementations not claiming support of certain Specialized Needs Annexes.

Annex C Requirements

Section 13 of the Ada Standard includes implementation advice paragraphs include the words "recommended level of support". The ACATS does not require implementations to conform to those paragraphs unless they claim support for Annex C, Systems Programming (because of C.2(2): "The implementation shall support at least the functionality defined by the recommended levels of support in Section 13.")

Tests that check conformance to the implementation advice are listed below:

CD10001  CD30003  CD33001  CD72A02
CD20001  CD30004  CD33002  CD90001
CD30001  CD30006  CD40001  CDE0002
CD30002  CD30007  CD72A01

Implementations that claim support for Annex C are required to process and pass the tests listed above.

Implementations that do not claim support for Annex C are still required to process these tests. Such implementations may reject the lines marked with the special comment -- ANX-C RQMT, in which case the test will be graded as "unsupported". If an implementation accepts such lines in one of these tests, then the test must be bound (linked) and executed, with a passed or not_applicable result.

4.1.6 Foreign Language Code

Several tests for Annex B features (and one Section 13 test) include files containing non-Ada code (Fortran, C, Cobol). These tests must be compiled, bound, and run by implementations that support foreign language interfaces to the respective non-Ada language. The foreign language code uses only the most basic language semantics and should be compilable by all Fortran, C, and Cobol compilers, respectively. In cases where a foreign language does not accept the code as provided, modifications are allowable. See Section 5.2.5.

Files that contain foreign code are identified by a special file extension. See Section 4.3.2.

The tests that include Fortran code are: CXB5004 and CXB5005

The tests that include C code are: CD30005, CXB3004, CXB3006, CXB3013, CXB3017, and CXB3018
The test that includes Cobol code is: CXB4009

4.2 Test Classes

There are six different classes of ACATS tests, reflecting different testing requirements of language conformity testing. Each test belongs to exactly one of the six classes, and its membership is encoded in the test name, as explained later. The purpose and nature of each test category is explained below. The test classifications provide an initial indication of the criteria that are used to determine whether a test has been passed or failed.

4.2.1 Class A

Class A tests check for acceptance (compilation) of language constructs that are expected to compile without error.

An implementation passes a class A test if the test compiles, binds, and executes reporting "PASSED". Any other behavior is a failure.

Only legacy tests are included in this class.

4.2.2 Class B

Class B tests check that illegal constructs are recognized and treated as fatal errors. They are not expected to successfully compile, bind, or execute. Lines that contain errors are marked -- ERROR: and generally include a brief description of the illegality on the same or following line. (The flag includes a final ":" so that search programs can easily distinguish it from other occurrences of the word "error" in the test code or documentation.) Some tests also mark some lines as -- OK, indicating that the line must not be flagged as an error.

An implementation passes a class B test if each indicated error in the test is detected and reported, and no other errors are reported. The test fails if one or more of the indicated errors are not reported, or if an error is reported that cannot be associated with one of the indicated errors. If the test structure is such that a compiler cannot recover sufficiently to identify all errors, it may be permissible to "split" the test program into separate units for re-processing (see Section 5.2.5 for instructions on modifying tests).

In some cases and for some constructs, compilers may adopt various error handling and reporting strategies. In cases where the test designers determined that an error might or might not be reported, but that an error report would be appropriate, the line is marked with -- OPTIONAL ERROR: or a similar phrase. In such cases, an implementation is allowed to report an error or fail to report an error without affecting the final grade of the test.

Similarly, in cases where the test designers determined that an error might be reported at one of several source locations, all such source locations are marked with -- POSSIBLE ERROR: and an indication of which error (if the test contains several) is expected. In such cases, an implementation is considered passing if it reports an error at any of the possible places for the error to be reported. It fails if no error is reported at any of the places.

4.2.3 Class C

Class C tests check that executable constructs are implemented correctly and produce expected results. These tests are expected to compile, bind, execute and report "PASSED" or "NOT-APPLICABLE". Each
class C test reports "PASSED", "NOT-APPLICABLE", or "FAILED" based on the results of the conditions tested.

An implementation passes a class C test if it compiles, binds, executes, and reports "PASSED". It fails if it does not successfully compile or bind, if it fails to complete execution (hangs or crashes), if the reported result is "FAILED", or if it does not produce a complete output report.

The tests CZ1101A, CZ1102A, CZ1103A, and CZ00004 are treated separately, as described in Section 5.3.2.

4.2.4 Class D

Class D tests check that implementations perform exact arithmetic on large literal numbers. These tests are expected to compile, bind, execute and report "PASSED". Each test reports "PASSED" or "FAILED" based on the conditions tested. Some implementations may report errors at compile time for some of them, if the literal numbers exceed compiler limits.

An implementation passes a class D test if it compiles, binds, executes, and reports "PASSED". It passes if the compiler issues an appropriate error message because a capacity limit has been exceeded. It fails if does not report "PASSED" unless a capacity limits is exceeded. It fails if it does not successfully compile (subject to the above caveat) or bind, if it fails to complete execution (hangs or crashes), if the reported result is "FAILED", or if it does not produce an output report or only partially produces one.

Only legacy tests are included in this class.

4.2.5 Class E

Class E tests check for constructs that may require inspection to verify. They have special grading criteria that are stated within the test source. They are generally expected to compile, bind and execute successfully, but some implementations may report errors at compile time for some tests. The "TENTATIVELY PASSED" message indicates special conditions that must be checked to determine whether the test is passed.

An implementation passes a class E test if it reports "TENTATIVELY PASSED", and the special conditions noted in the test are satisfied. It also passes if there is a compile time error reported that satisfies the special conditions. Class E tests fail if the grading criteria in the test source are not satisfied, or if they fail to complete execution (hang or crash), if the reported result is "FAILED", or if they do not produce a complete output report.

Only legacy tests are included in this class.

4.2.6 Class L

Class L tests check that all library unit dependences within a program are satisfied before the program can be bound and executed, that circularity among units is detected, or that pragmas that apply to an entire partition are correctly processed. These tests are normally expected to compile successfully but not to bind or execute. Some implementations may report errors at compile time; potentially illegal constructs are flagged with "-- ERROR:". Some class L tests indicate where bind errors are expected. Successful processing does not require that a binder match error messages with these indications.

An implementation passes a class L test if it does not successfully complete the bind phase. It passes a class L test if it detects an error and issues a compile time error message. It fails if the test successfully binds and/or begins execution. An L test need not report "FAILED" (although many do if they execute).
As with B-tests, the test designers determined that some constructs may or may not generate an error report, and that either behavior would be appropriate. Such lines are marked with "-- OPTIONAL ERROR." In such cases, an implementation is allowed to report an error or fail to report an error. If an error is reported at compile time, the binder need not be invoked. If no errors are reported at compile time, the binder must be invoked and must not successfully complete the bind phase (as indicated by the inability to begin execution).

### 4.2.7 Foundation Code

Files containing foundation code are named using the regular test name conventions (see Section 4.3). It may appear from their names that they represent class F tests. There is no such test class. Foundation code is only used to build other tests, so foundation units are not graded. However, if a foundation unit fails to compile, then the tests that depend on it cannot be compiled, and therefore will be graded as failed.

### 4.2.8 Specialized Needs Annex Tests

Specialized Needs Annex tests have no separate classifications and are classified in the same way as all other tests. There are Class B, Class C, and Class L SNA tests.

### 4.3 Naming Convention

This section describes the naming conventions used in ACATS 4.0, specifically as they apply to files. All file names are of the form `<name>.<type>`, where `<type>` is a one, two, or three character extension. File names indicate test class, compilation order (if applicable), and whether the test is implementation dependent or requires customization. When a test is included in a single file, `<name>` duplicates the test name. The same is true of a foundation. In multiple file tests, the first 7 characters of the file `<name>` are normally the same as the name of the test, however in some cases, the structure of the test requires that the file name be different from the Ada unit. The application of the conventions to tests is straightforward.

There are two different but similar naming conventions used in ACATS 4.0. Legacy tests use the naming conventions of early ACVC versions. Tests new since ACVC 1.12 use the modern convention. The conventions are consistently distinguishable at the 7th character of the name: legacy names have a letter in the 7th position, whereas newer (modern) names have a digit.

#### 4.3.1 Legacy Naming

The name of a legacy test is composed of seven or eight characters. Each character position serves a specific purpose as described in the table below. The first column identifies the character position(s) starting from the left, the second column gives the kind of character allowed, and the third gives the corresponding meaning:

<table>
<thead>
<tr>
<th>Position</th>
<th>Kind</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Letter</td>
<td>Test class (see Section 4.2)</td>
</tr>
<tr>
<td>2</td>
<td>Hexadecimal</td>
<td>AIG chapter containing the test objective</td>
</tr>
<tr>
<td>3</td>
<td>Hexadecimal</td>
<td>Section within the above AIG chapter</td>
</tr>
<tr>
<td>4</td>
<td>Alphanumeric</td>
<td>Sub-section of the above AIG section</td>
</tr>
<tr>
<td>5-6</td>
<td>Decimal</td>
<td>Number of the test objective within the above sub-section</td>
</tr>
</tbody>
</table>

29 August 2014  22
7 Letter
Letter identifier of the sub-objective of the above objective.

8 Alphanumeric
optional – Compilation sequence identifier — indicates the compilation order of multiple files that make up a single test. This position is used only if the test comprises multiple files.

The convention is illustrated below.

![Legacy File Name Convention Diagram]

In multiple file tests, the intended order of compilation is indicated by a numeral at position 8. The first file to be compiled has '0', the second has '1', and so forth.

The chapter and section numbers of the AIG (ACVC Implementer's Guide) correspond to those in [Ada83].

Note: The use of a ninth character ('m') to indicate the file containing the main subprogram has been discontinued. The following table lists the files containing the main subprograms of the legacy multiple file tests.

<table>
<thead>
<tr>
<th>AD7001C0.ADA</th>
<th>BA1010E0.ADA</th>
<th>BA1109A2.ADA</th>
<th>C83F01D0.ADA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD7001D0.ADA</td>
<td>BA1010F0.ADA</td>
<td>BA1110A1.ADA</td>
<td>C83F03C2.ADA</td>
</tr>
<tr>
<td>B38103C3.ADA</td>
<td>BA1010G0.ADA</td>
<td>BA2001F0.ADA</td>
<td>C83F03D0.ADA</td>
</tr>
<tr>
<td>B38103E0.ADA</td>
<td>BA1010H0.ADA</td>
<td>BA2003B0.ADA</td>
<td>C86004B2.ADA</td>
</tr>
<tr>
<td>B63009C3.ADA</td>
<td>BA1010I0.ADA</td>
<td>BA2011A1.ADA</td>
<td>C86004C2.ADA</td>
</tr>
<tr>
<td>B73004B0.ADA</td>
<td>BA1010J0.ADA</td>
<td>BA3001A0.ADA</td>
<td>CA1011A6.ADA</td>
</tr>
<tr>
<td>B83003B0.ADA</td>
<td>BA1010K0.ADA</td>
<td>BA3001B0.ADA</td>
<td>CA1012A4.ADA</td>
</tr>
<tr>
<td>B83004B0.ADA</td>
<td>BA1010L0.ADA</td>
<td>BA3001C0.ADA</td>
<td>CA1012B4.ADA</td>
</tr>
<tr>
<td>B83004C2.ADA</td>
<td>BA1010M0.ADA</td>
<td>BA3001E0.ADA</td>
<td>CA1013A6.ADA</td>
</tr>
<tr>
<td>B83004D0.ADA</td>
<td>BA1010N0.ADA</td>
<td>BA3001F0.ADA</td>
<td>CA1014A0.ADA</td>
</tr>
<tr>
<td>B83024F0.ADA</td>
<td>BA1010P0.ADA</td>
<td>BA3006A6.ADA</td>
<td>CA1020E3.ADA</td>
</tr>
<tr>
<td>B83E01E0.ADA</td>
<td>BA1010Q0.ADA</td>
<td>BA3006B4.ADA</td>
<td>CA1022A6.ADA</td>
</tr>
<tr>
<td>B83E01F0.ADA</td>
<td>BA1011B0.ADA</td>
<td>C38108C1.ADA</td>
<td>CA1102A2.ADA</td>
</tr>
<tr>
<td>B86001A1.ADA</td>
<td>BA1011C0.ADA</td>
<td>C38108D0.ADA</td>
<td>CA2001H3.ADA</td>
</tr>
<tr>
<td>B95020B2.ADA</td>
<td>BA1020A0.ADA</td>
<td>C39006C0.ADA</td>
<td>CA2002A0.ADA</td>
</tr>
<tr>
<td>BA1001A0.ADA</td>
<td>BA1020B6.ADA</td>
<td>C39006F3.ADA</td>
<td>CA2003A0.ADA</td>
</tr>
<tr>
<td>BA1010A0.ADA</td>
<td>BA1020C0.ADA</td>
<td>C64005D0.ADA</td>
<td>CA2004A0.ADA</td>
</tr>
<tr>
<td>BA1010B0.ADA</td>
<td>BA1020F2.ADA</td>
<td>C83022G0.ADA</td>
<td>CA2007A0.ADA</td>
</tr>
<tr>
<td>BA1010C0.ADA</td>
<td>BA1101B0.ADA</td>
<td>C83024E1.ADA</td>
<td>CA2008A0.ADA</td>
</tr>
<tr>
<td>BA1010D0.ADA</td>
<td>BA1101C2.ADA</td>
<td>C83F01C2.ADA</td>
<td>CA2009C0.ADA</td>
</tr>
</tbody>
</table>
The file name extension is three characters long. There are four extensions:

- **.ada**: A file that contains only Ada code. It does not require any pre-processing to create a compilable test. It will be submitted directly to the implementation for determination of test results. All implementations must correctly process these tests.

- **.dep**: A file that has a test involving implementation-dependent features of the language. These tests may not apply to all implementations.

- **.tst**: A file that has "code" that is not quite Ada; it contains "macro" symbols to be replaced by implementation-dependent values, and it must be customized (macro expanded) to prepare it for compilation (see Section 5.2.2). Once customized, the resulting test must be processed as indicated by its class.

- **.adt**: A file that has been modified by the macro processor. It contains only Ada code and may be submitted to the implementation for results. All implementations must correctly process these tests. There are no files in the ACATS distribution with this extension; they are only produced as the output of the macro processor.

Modern tests use different file name extensions (see 4.3.2).

> Note that legacy tests have not been renamed for ACATS 4.0. Since [Ada2012] includes some organizational differences from [Ada83], this means that the name of a legacy test sometimes will not correspond to the clause of [Ada2012] in which the tested feature is described.

### 4.3.2 Modern Naming

The name of a modern ACATS test is composed of seven or eight characters. Foundation code has a name composed of seven characters. The use of each character position is described below. The first column indicates the character position(s) starting from the left, and the second column indicates the kind of character allowed, and the third column gives the corresponding meaning:

<table>
<thead>
<tr>
<th>Position</th>
<th>Kind</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Letter</td>
<td>Test class; foundations are marked 'F'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If other than an 'x', the section of the Ada Standard describing the feature under test. An 'x' indicates that the test includes one or more features from an annex of the Ada Standard</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
<td>Core clause or annex letter identifier (either core or Specialized Needs Annex); clauses are a hexadecimal value</td>
</tr>
<tr>
<td>3</td>
<td>Alphanumeric</td>
<td>Sub-clause (if a core test), or clause (if an annex test); a number if less than 10, otherwise a letter with 10='A', 11='B', and so on</td>
</tr>
<tr>
<td>4</td>
<td>Alphanumeric</td>
<td>Foundation identifier (alphabetic, unless no foundation is required, in which case a '0')</td>
</tr>
</tbody>
</table>

### 4.3.1 Legacy Naming
Sequence number of this test in a series of tests for the same clause; foundation code will have "00".

optional — Compilation sequence identifier — indicates the suggested or required compilation order of multiple files that make up a single test (0 is compiled first). This position is used only if the test comprises multiple files.

The file name extension is a one or two character file name extension. There are six extensions:

.a A file that contains only Ada code (except for configuration pragmas in the case of some Specialized Needs Annex tests). It does not require any processing to prepare it for compilation (unless configuration pragmas must be handled separately). It is normally submitted directly to the implementation for determination of test results.

.am A file that contains the main subprogram for a multi-file test. Generally, this extension is used for only one file of a test. In rare cases (some Annex E tests), a multi-file test may have more than one file containing a "main" subprogram; in such cases, the correct testing procedure is described in the Special Requirements section of the test prologue.

.au A file that contains only Ada code that contains characters outside of the 7-bit ASCII character set. These files are provided in UTF-8 format with a starting byte-order mark. For ACATS 4.0, these tests must be compiled and run as all other tests of its test class, although usage of a different workflow (which must be documented if it is necessary) is allowed. (Note that [Ada2012] requires compilers to be able to process UTF-8 files, although the details [such as compiler options] might be different than ASCII source files.)

.ftn A file that contains Fortran language code and must be compiled by a Fortran compiler. These files are used by tests that check a foreign language interface to Fortran.

.c A file that contains C language code and must be compiled by a C compiler. These files are used by tests that check a foreign language interface to C.

.cbl A file that contains Cobol language code and must be compiled by a Cobol compiler. These files are used by tests that check a foreign language interface to Cobol.

A test that depends on foundation code has an alphabetic character in the fifth position of its name. The required foundation will have the same characters in the second through fifth positions of its name. For example, C123Axx depends on F123A00.
4.3.3 Multiple File Tests

When tests are contained in multiple files (i.e., compilation units are contained in different files), the file names are related. The first seven positions of the names of all the files (other than foundation files) comprised by a single test will be identical. The eighth position will provide a distinguishing alphanumeric which indicates the required compilation order. In legacy tests, the main subprogram is not indicated (see the table in section 4.3.1 for files containing main subprograms). For newer (modern) tests, the extension ".am" indicates the file with the main subprogram.

All tests apply the convention of naming the main subprogram the same as the file (excluding the file extension) plus, for legacy tests only, the letter 'm'. For example, the legacy test, C39006F, is contained in four files, named c39006f0.ada, c39006f1.ada, c39006f2.ada, and c39006f3.ada. The main subprogram of the test is contained in c39006f3.ada and is named C39006F3M. The test C390006 is also contained in four files, named c3900060.a, c3900061.a, c3900062.a, and c3900063.am. The main subprogram of the test is contained in c3900063.am and is named C3900063.

Unless otherwise required by a test objective, other library units in a test are named with the test name and a suffix. Typically, the suffix will be a number or an underscore followed by a few letters. Similarly, library units making up a foundation are usually named with the foundation name (or the foundation name and a suffix if there are multiple units in the foundation). This convention reduces name collisions with other tests and with implementation-defined units.

There are a small number of Specialized Needs Annex tests for the Distributed Processing Annex that require two active partitions and have two main subprograms. These tests have two files with the .am extension to signify the location of the (multiple) main subprograms.

4.4 Test Program Format

Each test file is composed of a test prologue, documenting the test, and the test code proper. All prologue lines are marked as comments. (The prologue in files containing non-Ada code is marked according to the comment conventions of the foreign language.)

The prologue for all tests is based on that of legacy tests. Legacy tests are generally, but not entirely, consistent in their use of the prologue. The format of the prologue between test files and foundation files is slightly different.

The general format of the prologue is as follows:

```
<file name>

The distribution name of the file containing this prologue.

DISCLAIMER
Use restrictions for ACATS tests; included in all tests.

OBJECTIVE
A statement of the test objective; included in all tests.

TEST DESCRIPTION
A short description of the design or strategy of the test or other pertinent information. Included in most newer tests but not generally included in legacy tests.

SPECIAL REQUIREMENTS
optional – Included if the test has any special requirements for processing. Normally, this section will be found only in Specialized Needs Annex tests. For example, an Annex E test may check for the correct implementation of partitions; the requirements for test
```
partitioning and what to use as a main subprogram in each partition would be
documented in this section.

TEST FILES
  optional – Included if the test depends on multiple files; identifies the component files of
  a multi-file test.

APPLICABILITY CRITERIA
  optional – Specifies the conditions under which the test can be ruled inapplicable.

PASS/FAIL CRITERIA
  optional – Explains how to interpret compilation, binding, and/or run-time results for
  grading the test.

MACRO SUBSTITUTIONS
  optional – Identifies the macro symbol(s) in the file that must be replaced and provides a
  brief description of what the replacement(s) represent. Appears only in legacy tests.

CHANGE HISTORY
  History of the test file. Included in all tests.

All tests have the line immediately after the disclaimer marked --*. Modern tests have the line after the
last prologue line (before the first line of executable code) marked -- ! No other comment lines are
marked with those conventions, so the start of the objective (which is the next line after the disclaimer)
and the first line of code may be found quickly with an editor search.

Some tests are composed of multiple files (other than foundation code). Rather than repeating the
complete prologue in each file, an alternate approach has been used. One file (usually the one containing
the main subprogram or the first file in the set) has the complete prologue; the other, related files have
those sections that apply to files (TEST FILES, CHANGE HISTORY) and refer to the file with the
complete prologue for the other sections.

4.5 General Standards

ACATS tests were developed to a general set of standards. To promote a variety of code styles and usage
idioms in the tests, standards were not necessarily rigorously enforced but were used as guidelines for test
writers. A maximum line length of 79 characters was used to enhance electronic distribution of tests
(except when specific testing requirements dictated otherwise, usually in .dep and .tst files). Tests tend to
be about 120 executable lines long, though many tests deviate from this norm (either longer or shorter) to
achieve a design that focuses on the objective and a readable, maintainable test. Sometimes complex
objectives have been divided into sub-objectives to achieve complete coverage in comprehensible,
maintainable tests. Some tests check multiple objectives; in other cases, sub-objectives are checked in
separate tests.

Legacy tests use only the basic 55-character set (26 capital letters, 10 digits, and 19 punctuation marks).
Unless there is a specific test requirement, numeric values are in the range (-2048..2047), which can be
represented in 12 bits. Numeric values are generally in the range (-128..127). Modern tests use both upper
and lower case letters and may use larger numeric values (but within the range (-65536..65535) except in
rare cases).

Legacy tests tend to use as few Ada features as necessary to write a self-checking executable test that can
be read and maintained. Modern tests tend to exhibit a usage-oriented style, employing a rich assortment
and interaction of features and exemplifying the kind of code styles and idioms that compilers may
encounter in practice.
In modern tests, Ada reserved words are entirely in lower case. Identifiers normally have their initial letter capitalized. Every attempt has been made to choose meaningful identifiers. In B class tests, identifier names often provide a clue to the specific case or situation under test. In C class tests, identifiers are normally chosen to help document the test design or the intent of the code.

Modern executable tests generally provide some visual separation of those test elements that focus on conformance issues from those that govern the flow of a test. For example, there is frequently a need to establish preconditions for a test and examine post-conditions after a section of test code has executed. To distinguish between constructs (types, objects, etc.) that are part of the test code and those that are artifacts of the testing process (e.g., pre-, post-conditions), the latter have TC_ prefixed to the identifier name. This prefix is shorthand for Test_Control.

### 4.6 Test Structure

Executable tests (class A, C, D, and E) generally use the following format:

```ada
with Report;
procedure Testname is
   <declarations>
begin
   Report.Test ("Testname", "Description ...");
   ...
   <test situation yielding result>
   if Post_Condition /= Correct_Value then
       Report.Failed ("Reason");
   end if;
   ...
   Report.Result;
end Testname;
```

The initial call to Report.Test prints the test objective using Text_IO output (unless the body of Report has been modified to do something else). After each section of test code, there is normally a check of post conditions. The if statement in this skeleton is such a check; unexpected results produce a call to Report.Failed. The sequence of test code / check of results may be repeated several times in a single test. Finally, there is a call to Report.Result that will print the test result to Text_IO output. Often, but not always, this structure is enclosed in a declare block.

One or more calls to Report.Failed will report a result of "FAILED" and a brief suggestion of the likely reason for that result.

More complex tests may include calls to Report.Failed in the code other than in the main program, and therefore exhibit the following format for the main procedure:

```ada
with Report;
procedure Testname is
   <declarations>
begin
   Report.Test ("Testname", "Description ...");
   ...
   Subtest_Call;
   ...
   Report.Result;
end Testname;
```

Fail conditions are detected in subprograms (or tasks) and Report.Failed is called within them.

Occasionally, as a test is running, it will determine that it is not applicable. In such a case, it will call Report.Not_Applicable that will report a result of "NOT_APPLICABLE" (unless there is also a call to Report.Failed).
Often, a test calls one of the functions Report.Ident_Int or Report.Ident_Bool to obtain a value that could be provided as a literal. These functions are intended to prevent optimizers from eliminating certain sections of test code. The ACATS suite has no intention of trying to discourage the application of optimizer technology, however satisfactory testing of language features often requires the presence and execution of specific lines of test code. Report.Ident_Int and Report.Ident_Bool are structured so that they can be modified when needed to defeat optimizer advances.

Class B tests may be structured differently. Since they are not executable, they normally do not include calls to Report.Test or Report.Result (since those lines of code would have no output effect). Instead, intentional errors are coded that invoke specific legality rules. The source code includes comments that document expected compiler results. Legal constructs may also be included in B class tests. Constructs that are allowed by the legality rules are marked -- OK; constructs that are disallowed are marked -- ERROR:. (Some additional markings can also be used, see 4.2.2.) There is usually a brief indication of the nature of an intentional error on the same line or the line following a comment. The indications of expected results are approximately right justified to the code file margin, about column 79, for quick visual identification.

Class L tests are multifile tests with illegalities that should be detected at bind time. They are generally structured like class C tests, often with calls to Report.Test and Report.Result, but they are not expected to execute.

### 4.7 Delivery Directory Structure

The delivery of ACATS tests is structured into a directory tree that reflects the organization of the test suite and support code.

The top-level directory contains the support subdirectory, the docs subdirectory, and a subdirectory for each major grouping of tests. The support subdirectory contains all support packages (Report, ImpDef, TCTouch) and the source code for all test processing tools (Macro expander, Wide Character processor). Each of the other subdirectories contains all tests that begin with the indicated prefix. For example, all of the B2* tests are in the b2 subdirectory; all of the CXH* tests are in the cxh subdirectory. Note that all of the A* tests are in the a directory, all of the D* tests are included in the d subdirectory, and all of the E* tests are included in the e subdirectory. The l directory contains the L tests for the core; other L tests are in directories named with three letters, indicating the class (l) and the Specialized Needs Annex to which the tests apply.

Subdirectories that would be empty are not stubbed.

The following figure sketches this scheme, but does not show complete detail. A list of all subdirectories is included in Section 5.1.2.
4.8 File Format

To conserve space and ease downloading, all files in the delivered ACATS 4.0 (including test files, foundation files, and support files) have been compressed. Except as noted below, decompressed files (see Section 5.1.2) use only ASCII characters. A few tests use Unicode characters; these are indicated by an .au extension. Some of the documentation files are provided in PDF and/or HTML forms for greater readability. (The HTML documentation files include GIF and PNG graphics files.) Other than the documentation files, no formatting control characters, rulers or other information intended for editors or display programs are included in the files.

Files with the .zip extension have been compressed using a DOS zip utility; files with the .Z extension have been first put in Unix tar format and then compressed with Unix compress.
Section 5: Using the ACATS

There are eight major steps involved in using the ACATS test suite; two of them are sometimes not required. The steps are: installing the software, tailoring the software, processing the support files, establishing command scripts, processing the ACATS tests, grading the test results, addressing problems (if necessary), and reprocessing problem tests (if necessary). The first six of these tasks must be completed successfully to accomplish a test run. The first four normally need be completed only once for each ACATS release. Each step is explained in the following sections. The flow from one to the next is illustrated in the following figures.
5.1 Installation of the ACATS Test Suite

The ACATS test suite must be unloaded from the delivery medium or downloaded from a delivery site before it can be unpacked, customized for an implementation, run, and graded.

5.1.1 Contents of the ACATS Delivery

The delivery consists of 1 ZIP archive (set of compressed files) or 1 compressed tar file. Each ZIP archive or compressed tar file contains compressed versions of ACATS software (test, foundation, and/or support code) structured into a directory tree. Files must be extracted from the archives. The archive contents is described later in this section.

Usually, some test errors will be noted in the test suite. If possible, the ACAA will correct the errors and issue a corrected test. If a correction is not possible, the test will be withdrawn. Withdrawn tests are not used in conformity assessments. For a period after the issuance of a corrected test, either the original or the corrected test can be used for conformity assessment. See the ACAA's procedures [Pro31] for details.

The ACAA also will issue new tests periodically. As with modified tests, new tests must be available for a period of time before they are required in conformity assessments.

These changes to the issued ACATS are documented in the ACATS Modification List (AML). This list includes a list of all new tests, all modified tests, and all withdrawn tests, and an indication as to when each will be (or is) required for conformity assessments. Each version of the modification list is given a suffix letter. A ZIP archive and tar file containing the new and/or modified tests is available. The files are named MOD_4_0x, where 'x' represents the suffix letter for the AML version.

These files can be found on the ACAA's web site:
The AML is also distributed by e-mail. To receive these lists, join the ACAA mailing list. To do so, simply send a message to 
listserv@ada-auth.org.
with a body of
Join Acaa

5.1.2 Guide to Decompressing Files

The ACATS files are provided in two forms: compressed in zip format and compressed in Unix compress format. Zipped files are included in a zip archive (files) with the file extension .zip. A Unix compressed files, with extension .Z, contains a Unix tar file. This section provides generic instructions for uncompressing them. These instructions are not the only ways to uncompress the files; sophisticated users may wish to use their own procedures.

If the instructions below are used, the following subdirectories will have been created and populated with test files after all decompression:

```
./acats4_0/a   ./acats4_0/be   ./acats4_0/cd   ./acats4_0/cxc
./acats4_0/b2  ./acats4_0/bxa  ./acats4_0/ce   ./acats4_0/cxd
./acats4_0/b3  ./acats4_0/c2   ./acats4_0/cxa   ./acats4_0/cxe
./acats4_0/b4  ./acats4_0/c3   ./acats4_0/cxb   ./acats4_0/cxf
./acats4_0/b5  ./acats4_0/c4   ./acats4_0/cz    ./acats4_0/cxg
./acats4_0/b6  ./acats4_0/c5   ./acats4_0/d     ./acats4_0/cxh
./acats4_0/b7  ./acats4_0/c6   ./acats4_0/e     ./acats4_0/lxd
./acats4_0/b8  ./acats4_0/c7   ./acats4_0/1     ./acats4_0/lxe
./acats4_0/b9  ./acats4_0/c8   ./acats4_0/cxc    ./acats4_0/lxh
./acats4_0/ba  ./acats4_0/c9   ./acats4_0/bxd    ./acats4_0/docs
./acats4_0/bb  ./acats4_0/ca   ./acats4_0/bxe    ./acats4_0/support
./acats4_0/bc  ./acats4_0/cb   ./acats4_0/bxf
./acats4_0/bd  ./acats4_0/cc   ./acats4_0/bxh
```

Note that the names are given here in all lowercase; some systems may create uppercase names. The path separator, shown here as '/', may also differ.

5.1.2.1 Decompressing Zipped Files

All ACATS files have been compressed (zipped) into compressed archives (zip-files) that have the MS-DOS file extension ".zip". A Windows command-line utility was used to compress them. They must be decompressed before they can be further processed. A decompression utility is available from the source of the ACATS distribution. All ACATS 4.0 files may be decompressed using the following steps. Approximately 48 MB of free space on a Windows machine hard drive will be required to accomplish the decompression using this technique.

Create a directory on the hard disk to contain ACATS. In these examples, we assume the name is acats4_0, but any name can be used. Copy the archive (file with .zip extension) to the hard disk in the new directory. Decompress it insuring that directories are used. For the unzip program, this is the default setting. For the pkunzip program, this is the -d option. For the winzip program, ensure that "Use Directory Names" is checked. Also, ensure that the files are decompressed into the proper directory.
For command line decompressors, this means ensuring that the current subdirectory is acats4_0. For winzip, this simply means selecting acats4_0 as the extract path.

For example, using unzip, and assuming that the archive name is ACATS40.zip, type

```
cd acats4_0
unzip ACATS40
```

to set the proper directory, and
to extract the files.

The files were compressed on a Windows system, where <CR><LF> is used as a line terminator. Decompressors for other systems using other line terminators should be able convert the line terminators. The ACAA has a short Ada program which converts a file from Windows to Unix format; please send the ACAA mail at agent@ada-auth.org to request it if needed.

After all files have been extracted from the archive, delete the archive file from the hard disk if you wish to conserve space.

As it decompresses files, `unzip` will restore the directory structure of the files, creating all needed subdirectories.

Some users may prefer to work with ACATS files in an alternate directory structure or none at all. If the `unzip` utility is invoked with the "-j" option, all files in the archive will be decompressed and placed in the local working directory. In other words, none of the above subdirectories will be created. Since there are too many ACATS files to fit into a root DOS directory, if you wish to put all files in a single directory, you must first create a subdirectory (e.g., `mkdir \ACATS`) and unzip all archives there.

### 5.1.2.2 Decompressing Unix Compress Files

All ACATS files have been included in 1 Unix tar format file and then compressed using the Unix compress utility. To create a set of ACATS files, first copy the compressed files acats_40.tar.Z from the distribution source to a hard drive. Uncompress the file with the Unix command

```
uncompress acats_40.tar.Z
```

(Note that particular Unix implementations may have different formats or require specific qualifiers.) After the ACATS file has been uncompressed, it must be untarred. Move to the directory where you want the acats4_0 directory to be created and then untar the ACATS files

```
tar -xvf <path>/acats_40.tar
```

where `<path>` is the location of the uncompressed tar file.

Please note that these are generic instructions and may need to be customized or modified for specific systems.

### 5.1.3 Files With Non-Graphic Characters

Four ACATS test files contain ASCII non-graphic (control) characters that may be lost or corrupted in the file transfer and decompression process. The user must ensure that the proper characters are restored as necessary.

There are also a number of ACATS test files provided in UTF-8 format. These files have an .au file extension. The user must take care that any processing tools do not corrupt UTF-8 files. (This would be unusual: it is most likely if a tool does not recognize UTF-8 formatted files and also modifies Latin-1 characters with codes larger than 127.)
The following paragraphs describe the four tests with ASCII non-graphic characters.

### 5.1.3.1 A22006C

This test checks that format effectors can appear at the beginning of a compilation. At the beginning of the file, the first line is empty (indicated by the system's end-of-line marker, which may be a sequence of one or more characters or may be indicated by some other means). The second line contains 20 characters: 6 control characters followed by the comment delimiter, a space, and the file name (A22006C.ADA). The control characters are:

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Ada Name</th>
<th>Decimal Value</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Vertical tab</td>
<td>ASCII.VT</td>
<td>11</td>
<td>0B</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Form feed</td>
<td>ASCII.FF</td>
<td>12</td>
<td>0C</td>
</tr>
</tbody>
</table>

### 5.1.3.2 B25002A

This test checks that ASCII control characters (other than format effectors) are not permitted in character literals. The expected characters are documented in source code comments, using the customary 2- or 3-letter mnemonics. The 28 characters are used in their ASCII order, and have ASCII values 0 through 8, 14 through 31, and 127.

### 5.1.3.3 B25002B

This test checks that the five ASCII format effector characters cannot be used in character literals. There are two groups of code containing the illegal characters; in each group, the characters appear in the order given below:

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Ada Name</th>
<th>Decimal Value</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal tab</td>
<td>ASCII_HT</td>
<td>9</td>
<td>09</td>
</tr>
<tr>
<td>Vertical tab</td>
<td>ASCII.VT</td>
<td>11</td>
<td>0B</td>
</tr>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Form feed</td>
<td>ASCII.FF</td>
<td>12</td>
<td>0C</td>
</tr>
</tbody>
</table>

### 5.1.3.4 B26005A

This test checks the illegality of using control characters in string literals. Each string literal (ASCII codes 0 through 31 and 127) is used once, and the uses appear in ASCII order. Each use is also documented in a source code comment, which identifies the character by its common 2- or 3-character mnemonic.
5.2 Tailoring the ACATS Test Suite

There are some files in the delivery that require modification before ACATS 4.0 is ready for processing by an Ada implementation. Package ImpDef (impdef.a) must be edited to include values suitable for proper testing of an implementation if the defaults are not acceptable. The macros.dfs file must similarly be edited to include values suitable for testing. All .tst files (including package Spprt13 (spprt13s.tst)) must have their macro symbols replaced by implementation specific values. A body for FcnDecl (fcndcl.ada) must be provided if necessary. Finally, Package Report (repbody.ada) must be modified if necessary.

The required customization is described in the following sections.

Customizations of these files from previous versions of the ACATS suite generally can be used with ACATS 4.0, but users should ensure that neither their requirements nor the underlying files have changed since the customizations were made.

5.2.1 ImpDef Customization

For ACATS 4.0, there are two new parameters in Impdef. Other than those parameters, there was no change to Impdef or any of its children from ACATS 3.1 to ACATS 4.0. A version of any of these packages that was tailored for ACATS 3.1 should be valid for ACATS 4.0 once the new parameters are defined unless some implementation characteristics have changed.

ACATS tests use the entities in ImpDef to control test execution. Much of the information in ImpDef relates to the timing of running code; for example, the minimum time required to allow a task switch may be used by a test as a parameter to a delay statement. The time to use is obtained as an ImpDef constant. impdef.a was added as a new feature to ACATS 2.0 suite. It is related to macro.dfs in that it must be customized with values specific to an implementation and ACATS tests will rely on these values. ImpDef is different in the following respects:

- Defaults are provided. Some implementations may be able to rely entirely on the default values and subprograms, so no customization would be necessary.
- Some implementations may choose to provide bodies for procedures and/or functions. Bodies so provided must satisfy requirements stated in ImpDef.
- Tests depending on Impdef do not need customization (macro substitution). Instead, ImpDef must be available at compile time (i.e., included in the environment) for tests that rely upon it. This simplifies the customization process and management and also is similar to the way that Ada projects typically manage configuration parameters.

There are child packages of ImpDef for each of the Specialized Needs Annexes. An implementation that uses one or more of the Specialized Needs Annexes in its conformity assessment must customize the associated ImpDef child packages (or rely on their defaults) and must set the appropriate Booleans in impdef.a. It is not necessary to customize Impdef children for Specialized Needs Annexes that are not included in a particular conformity assessment.

Specific instructions for the values required by ImpDef and its children are included in impdef.a, impdefc.a, impdefd.a, impdefe.a, impdefg.a, and impdefh.a. (Note that impdefc, for example, refers to Annex C.) An excerpt from ImpDef is included in Annex B, “Parameterization Files”.

All implementations must customize impdef.a unless they wish to rely on the defaults provided. ImpDef must be part of the environment whenever a test that depends on it is processed. Similarly, the child of Impdef corresponding to each Specialized Needs Annex that the implementer intends to
test during a conformity assessment must be customized and be part of the environment when the Annex tests are processed.

5.2.2 Macro Defs Customization

There was no change to the macro.dfs file from ACATS 3.1 to ACATS 4.0. A version of macro.dfs that was tailored for ACATS 3.1 should be valid for ACATS 4.0 unless some implementation characteristics have changed.

Tests in files with the extension .tst contain symbols that represent implementation dependent values. The symbols are identifiers with a initial dollar sign ($). Each symbol must be replaced with an appropriate textual value to make the tests compilable. This process is sometime known as macro substitution.

The Macrosub program distributed with the ACATS can automatically perform the required substitutions. This program reads the replacement values for the symbols from the file macro.dfs and edits all the .tst tests in the suite to make the needed changes. It writes the resulting, compilable programs into files with the same name as the original but with the extension .adt. A sample macro.dfs is included with the ACATS; it contains descriptions of all the symbols used in the test suite.

Substitutions using the Macrosub program may be made as follows:
1. Edit the file macro.dfs using values appropriate for the implementation. Symbols that use the value of MAX_IN_LEN are calculated automatically and need not be entered.
2. Create a file called tsttests.dat that includes all of the .tst test file names, and their directory locations if necessary. A version of this file (without directory information) is supplied.
3. Compile and bind Macrosub.
4. Run Macrosub.

The program will replace all symbols in the .tst files with values from macro.dfs. Test files with the original test name but the extension .adt will contain the processable tests. The original .tst files will not be modified.

5.2.3 Packages SPPRT13 and FCNDECL

Package SPPRT13 declares six constants of type System.Address that are primarily used by tests of Section 13 features. It is in the file spprt13s.tst. As distributed, the package uses macro symbols that must be replaced. In most cases, the substitution can be accomplished by the macro substitution described in the preceding section. If appropriate literals, constants, or predefined function calls can be used to initialize these constants, they should be supplied in macro.dfs. Otherwise, the package FCNDECL must be modified.

All implementations should verify that package SPPRT13 can be properly customized using the macro substitution technique. Note that a body for SPPRT13 is illegal, nor is it allowed to add declarations to package SPPRT13.

The specification for package FCNDECL is in the file fcndecl.ada. SPPRT13 depends on FCNDECL (in a context clause that both withs it and uses it). As supplied with the ACATS, FCNDECL is an empty package specification. If appropriate literals, constants, or predefined function calls cannot be used to customize the constants declared in SPPRT13, the implementer must declare appropriate functions in the specification of FCNDECL and provide bodies for them in a package body or with a pragma Import.

Modifications to FCNDECL must receive advance approval from the ACAL (and, if necessary, the ACAAL) before use in a conformity assessment.
5.2.4 Modification of Package REPORT

All executable tests use the Report support package. It contains routines to automate test result reporting as well as routines designed to prevent optimizers from removing key sections of test code. The specification of package Report is in the file repspec.ada; the body is in repbody.ada.

Under some conditions, the body of package Report may need to be modified. For example, the target system for a cross-compiler may require a simpler I/O package than the standard package Text_IO. In such a case, it may be necessary to replace the context clause and the I/O procedure names in the body of Report.

Modifications to Report must receive advance approval from the ACAL (and, if necessary, the ACAA) before use in a conformity assessment.

5.2.5 Allowed Test Modifications

Class B tests have one or more errors that implementations must identify. These tests are structured such that, normally, implementations can report all included errors. Occasionally, an implementation will fail to find all errors in a B-test because it encounters a limit (e.g., error cascading, resulting in too many error reports) or is unable to recover from an error. In such cases, a user may split a single B-test into two or more tests. The resulting tests must contain all of the errors included in the original test, and they must adhere as closely as possible to the style and content of the original test. Very often, the only modification needed is to comment out earlier errors so that later errors can be identified. In some cases, code insertion will be required. An implementation must be able to demonstrate that it can detect and report all intended B-test errors.

Splits may also be required in executable tests, if, for example, an implementation capacity limitation is encountered (e.g., a number of generic instantiations too large for the implementation). In very exceptional cases, tests may be modified by the addition of an attribute definition clause (to alter the default size of a collection), or by the addition of an elaboration Pragma (to force an elaboration order).

Tests that use configuration pragmas (see 5.5.5.5) may require modification since the method of processing configuration pragmas is implementation dependent.

Some tests include foreign language code (Fortran, C, or COBOL). While the features used should be acceptable to all Fortran, C, and COBOL implementations, respectively, some implementations may require modification to the non-Ada code. Modifications must, of course, preserve the input-output semantics of the (foreign language) subprogram; otherwise, the ACATS test will report a failure.

All splits and modifications must be approved in advance by the ACAL (and, if necessary, the ACAA) before they are used in a conformity assessment. It is the responsibility of the user to propose a B-test split that satisfies the intention of the original test. Modified tests should be named by appending an alphanumeric character to the name of the original test. When possible, line numbers of the original test should be preserved in the modification.

All tests must be submitted to the compiler as distributed (and customized, if required). If a test is executable (class A, C, D, E) and compiles successfully, then it must be run. Modified tests or split tests may be processed next. Only the results of the modified tests will be graded.

If the ACAA has issued an ACATS Modification List (see Section 5.1.1), then the modified versions of tests with modifications must be used. Either the original version or the modified version of a test with an allowed modification may be used.
5.3 Processing the Support Files

After all the files identified in Section 5.2 have been customized as needed and required, the support files can be processed and the reporting mechanism can be verified.

5.3.1 Support Files

The following files are necessary to many of the ACATS tests. Implementations that maintain program libraries may wish to compile them into the program library used for conformity assessment:

- repspec.ada
- repbody.ada
- impdef.a
- impdefc.a (if testing Annex C)
- fnndecl.ada
- impdefd.a (if testing Annex D)
- checkfil.ada
- impdefe.a (if testing Annex E)
- lencheck.ada
- impdefg.a (if testing Annex G)
- enumchek.ada
- impdefh.a (if testing Annex H)
- tctouch.ada
- spprt13s.adt (after macro substitution)

Depending on local requirements and strategy, it may also be convenient to compile all foundation code into the program library as well.

5.3.2 "CZ" Acceptance Tests

Four tests having names beginning "CZ" are part of the ACATS suite. Unlike other tests in the suite, they do not focus on Ada language features. Instead, they are intended primarily to verify that software needed for the correct execution of the test suite works as expected and required. They check, for example, to see that package Report and package TCTouch work correctly.

All CZ tests must execute correctly and exhibit the prescribed behavior for a successful conformity assessment. CZ tests must be processed and run as the first step of a conformity assessment to ensure correct operation of the support software.

The acceptance test CZ1101A tests the correct operation of the reporting facilities of package Report, including checks that Not_Applicable and Failed calls are reported properly, and that premature calls cause failure. Therefore, CZ1101A will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ1101A is included in Annex C, "Results of CZ Tests" (times and dates in the actual output will differ).

The acceptance test CZ1102A tests the correct operation of the dynamic value routines in Report. This test should report "PASSED"; any other result constitutes a test failure.

The acceptance test CZ1103A ensures the correct operation of procedure Checkfile. (Some of the executable file I/O tests use a file checking procedure named Checkfile that determines an implementation's text file characteristics. The source code for this procedure is in the file checkfil.ada.) CZ1103A checks whether errors in text files are properly detected, therefore, CZ1103A will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ1103A is included in Annex C, "Results of CZ Tests" (times and dates in the actual output will differ).
The acceptance test CZ00004 produces output that verifies the intent of the conformity assessment. It relies on ImpDef having been correctly updated for the conformity assessment and produces output identifying the annexes (if any) that will be included as part of the conformity assessment. This test also checks for the proper operation of the TCTouch package, includes checks that assertion failures are reported properly, therefore CZ00004 will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ00004 is included in Annex C, “Results of CZ Tests”; since this output includes values from the customized impdef, non-failure lines may vary from those in the expected output. However, the number of lines and their relative positions should not change (only the contents of the lines can change).

5.4 Establishing Command Scripts

Users will often find it convenient to run large numbers of ACATS tests with command scripts. This section discusses some of the issues to be considered in developing a script.

5.4.1 Command Scripts

All compiler options and switches that are appropriate and necessary to run the ACATS tests must be identified and included in commands that invoke the compiler. The same is true for the binder or any other post-compilation tools. Any implementation dependent processing of partitions, configuration pragmas, and strict mode processing must be part of the scripts for running tests that rely on these features.

A script should compile (only) all class B tests. It should compile and bind all class L tests; if link errors are not explicitly given, the script should attempt to execute the L tests. It should compile all class F files. It should compile, bind, and execute all class A, C, D, and E tests.

Sample commands for processing the ACATS are a required part of a formal Ada Conformity Assessment Test Report. If a test report is available for the implementation being tested, these commands can be used as a guideline for developing command scripts.

5.4.2 Dependencies

A command script must take account of all required dependencies. As noted earlier, some tests are composed of multiple test files. Also, some tests include foundation code, which may be used by other tests. If a foundation is not already in the environment, it must be compiled as part of building the test. All files that are used in a test must be compiled in the proper order, as indicated by the file name. For implementations that require the extraction individual compilation units from test files before submission to the compiler, the individual units must be submitted to the compiler in the same order in which they appear in the file.

5.5 Processing ACATS Tests

After the ACATS tests and support code has been installed and all required modifications and preliminary processing have been completed, the suite can be processed by an implementation. This section describes the tests required for conformity assessment, required partitioning, how tests may be bundled for efficiency, and certain processing that may be streamlined. It also describes how the suite has been organized to allow a user to focus on specific development needs.
5.5.1 Required Tests

An implementation may be tested against the core language only or the core language plus one or more Specialized Needs Annexes. All core tests (except as noted in 5.5.4) must be processed with acceptable results for conformity assessment of the core language. All legacy tests, as well as all modern tests for clauses 2-13 and annexes A and B are core tests. Conformity assessment including one or more Specialized Needs Annexes requires that all tests for the annex(es) in question be correctly processed in addition to all core tests.

Tests that are not applicable to an implementation (e.g., because of size limitations) and tests that report "NOT APPLICABLE" when run by an implementation must nevertheless be processed and demonstrate appropriate results.

Tests that are withdrawn on the current ACATS Modification List as maintained by the ACAA need not be processed.

5.5.2 Test Partitions

Unless otherwise directed by the Special Requirements section of a test, all tests are to be configured and run in a single partition. The method of specifying such a partition is implementation dependent and not determined by the ACATS. The only tests that must be run in multiple partitions are those that test Annex E, Distributed Systems.

5.5.3 Bundling Test Programs

In some situations, the usual test processing sequence may require an unacceptable amount of time. For example, running tests on an embedded target may impose significant overhead time to download individual tests. In these cases, executable tests may be bundled into aggregates of multiple tests. A set of bundled tests will have a driver that calls each test in turn; ACATS tests will then be called procedures rather than main procedures. No source changes in the tests are allowed when bundling; that is, the only allowed change is the method of calling the test. Since ACATS tests are often designed to follow common usage patterns, including reuse of units, and reusable units are often self-initializing (so that they are resilient against misuse), not all ACATS tests can be bundled arbitrarily. In particular, foundations and shared support code may not (and cannot, in general, because of the need to test elaboration actions) support re-initialization and thus a bundled test may malfunction if it runs after another use of the same foundation or support routine. It is the responsibility of the ACATS user to ensure that bundled tests execute properly when bundled.

All bundles must be approved by the ACAL (and, if necessary, the ACAA) to qualify for a conformity assessment. It is the responsibility of the user to identify the tests to be bundled and to write a driver for them.

5.5.4 Processing that may be Omitted

A user may streamline processing of the ACATS tests to the greatest degree possible consistent with complete processing of all tests.

Many modern tests rely on foundation code. A foundation need not be compiled anew each time a different test uses it. In a processing model based on a program library, it is reasonable to compile the code into the library only once and allow the binder to use the processed results for each test that withs the foundation.
A user may determine, with ACAL concurrence, that some tests require support that is impossible for the implementation under test to provide. For example, there are tests that assume the availability of file I/O whereas some (embedded target) implementations do not support file I/O. Those tests need not be processed during witness testing; however, the implementer must demonstrate that they are handled in accordance with the language standard. This demonstration may be performed before witness testing, in which case it need not be repeated.

Annex B tests that require foreign language code (Fortran, C, COBOL) to be compiled and bound with Ada code need not be processed if an implementation does not support a foreign language interface to the respective language.

Tests for the Specialized Needs Annexes of Ada need not be processed except by implementations that wish to have Annex results documented. In that case, only the tests for the annex(es) in question (in addition to all core tests) need be processed. If any tests for a particular Annex are processed, then all tests for that Annex must be processed. If an implementation does not support a feature in a Specialized Needs Annex test, then it must indicate the non-support by rejecting the test at compile time or by raising an appropriate exception at run time. (See Ada 1.1.3(17).)

No withdrawn test need be processed. Tests classified as Pending New in the current ACATS Modification List also do not need to be processed. Pending New tests are new tests included with the ACATS for review purposes, and are not yet required for conformity assessment. (Tests classified as New in the current ACATS Modification List do need to be processed; these are required for conformity assessments.)

5.5.5 Tests with Special Processing Requirements

Some tests may require special handling. These are primarily SNA tests, but some core tests are affected. For example, distributed processing tests may require an executable image in multiple partitions, where partitions are constructed in an implementation specific manner. Real-time processing tests may have configuration pragmas that have to be handled in an implementation specific way. Numeric processing tests require strict mode processing to be selected. Each such test has a Special Requirements section in the test header describing any implementation specific handling that is required for the test.

A list of all such tests is provided in A.6, “Tests With Special Requirements”.

5.5.5.1 Tests Involving Limited Views

[Amend1] added the concept of limited views to Ada. For most ACATS tests, the possibility of limited views can be ignored, as either they are not used at all, or they need not be separated from the full view in the environment. (It is presumed that adding the full view to the environment also adds the corresponding limited view.)

However, a few tests require that the limited view of a unit be added to the environment separately from the full view of the unit. Such tests have dependencies in the full views on units that have not yet been compiled (added to the environment). For these tests, any extra steps needed to add the limited view to the environment separately from the full view will need to be accomplished.

The tests identified below need to add the limited view of one or more units to the environment separately from the full view of the units.

c11023  c12001
5.5.5.2 Foreign Language Interface Tests

Annex B, Interface to Other Languages, is part of the Ada core language. Any implementation that provides one or more of the packages Interfaces.C, Interfaces.COBOL, or Interfaces.Fortran must correctly process, and pass, the tests for interfaces to C, COBOL, and/or Fortran code respectively, with the possible exception of tests containing actual foreign code.

An implementation that provides one or more of these Interfaces child packages must successfully compile the Ada units of tests with actual foreign language code. If the implementation does not support the actual binding of the foreign language code to Ada, these tests may report binding errors, or may reject the pragma Import, in which case they may be graded as inapplicable. If the implementation supports the binding and an appropriate compiler is available, the tests must execute and report "Passed". If the implementation supports the binding, but it is not feasible to have an appropriate compiler available, then the tests may be graded as inapplicable by demonstrating that they fail to bind.

If one of the Interfaces child packages is not provided, then the corresponding tests may be graded as inapplicable, provided they reject the corresponding with clause.

The tests involving interfaces to foreign code are listed below.

The foreign language code included in ACATS tests uses no special or unique features, and should be accepted by any standard (C, COBOL, or Fortran) compiler. However, there may be dialect problems that prevent the code from compiling correctly. Modifications to the foreign language code are allowable; the modifications must follow the code as supplied as closely as possible and the result must satisfy the requirements stated in the file header. Such modifications must be approved in advance by the ACAL (and, if necessary, the ACAA).

The method for compiling foreign code is implementation dependent and not specified as part of the ACATS. Ada code in these tests must be compiled as usual. The Ada code includes Pragma Import that references the foreign language code. The link name of foreign language object code must be provided in ImpDef. When all code has been compiled, the test must be bound (including the foreign language object code) and run. The method for binding Ada and foreign language code is implementation dependent and not specified as part of the ACATS. The test must report "PASSED" when executed.

**C Language Interface**

If the implementation provides the package Interfaces.C, the tests identified below must be satisfactorily processed as described above.

The starred tests contain C code that must be compiled and linked if possible, as described above. The C code is easily identifiable because the file has the extension .C. The C code may be modified to satisfy dialect requirements of the C compiler. The C code files must be compiled through a C compiler, and the resulting object code must be bound with the compiled Ada code. Pragma Import will take the name of the C code from ImpDef.

<table>
<thead>
<tr>
<th>CD Code</th>
<th>EXB Code 1</th>
<th>EXB Code 2</th>
<th>EXB Code 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd30005*</td>
<td>exb3005</td>
<td>cxb3010</td>
<td>cxb3015</td>
</tr>
<tr>
<td>cxb3001</td>
<td>exb3006*</td>
<td>cxb3011</td>
<td>cxb3016</td>
</tr>
<tr>
<td>cxb3002</td>
<td>exb3007</td>
<td>cxb3012</td>
<td>cxb3017*</td>
</tr>
<tr>
<td>cxb3003</td>
<td>exb3008</td>
<td>cxb3013*</td>
<td>cxb3018*</td>
</tr>
<tr>
<td>cxb3004*</td>
<td>exb3009</td>
<td>cxb3014</td>
<td></td>
</tr>
</tbody>
</table>
COBOL Language Interface
If the implementation provides the package Interfaces.COBOL, the tests identified below must be processed satisfactorily, as described above.

The starred test contains COBOL code that must be compiled and linked if possible, as described above. The COBOL code is easily identifiable because the file has the extension .CBL. The COBOL code may be modified to satisfy dialect requirements of the COBOL compiler. The COBOL code files must be compiled through a COBOL compiler, and the resulting object code must be bound with the compiled Ada code. Pragma Import will take the name of the COBOL code from ImpDef.

<table>
<thead>
<tr>
<th>Test Code</th>
<th>Test Code</th>
<th>Test Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxb4001</td>
<td>cxb4004</td>
<td>cxb4007</td>
</tr>
<tr>
<td>cxb4002</td>
<td>cxb4005</td>
<td>cxb4008</td>
</tr>
<tr>
<td>cxb4003</td>
<td>cxb4006</td>
<td>cxb4009*</td>
</tr>
</tbody>
</table>

Fortran Language Interface
If the implementation has a Fortran language interface, the tests identified below must be processed satisfactorily, as described above.

The starred tests contain Fortran code that must be compiled and linked if possible, as described above. The Fortran code is easily identifiable because the file has the extension .FTN. The Fortran code may be modified to satisfy dialect requirements of the Fortran compiler. The Fortran code files must be compiled through a Fortran compiler, and the resulting object code must be bound with the compiled Ada code. Pragma Import will take the name of the Fortran code from ImpDef.

<table>
<thead>
<tr>
<th>Test Code</th>
<th>Test Code</th>
<th>Test Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxb5001</td>
<td>cxb5003</td>
<td>cxb5005*</td>
</tr>
<tr>
<td>cxb5002</td>
<td>cxb5004*</td>
<td></td>
</tr>
</tbody>
</table>

5.5.5.3 Tests for the Distributed Systems Annex
The ACATS tests for the Distributed Systems Annex are applicable only to implementations that wish to test this SNA. Not all of these tests apply to all implementations, since the annex includes some implementation permissions that affect the applicability of some tests.

The principal factors affecting test applicability are:
1. whether the Remote_Call_Interface pragma is supported;
2. whether a Partition Communication System (PCS) is provided (i.e., whether a body for System.RPC is provided by the implementation);
3. whether the implementation has taken advantage of the permission to change the specification of System.RPC;
4. whether the Real-Time Annex is also supported.

An implementation may test for the annex without providing a PCS. In order to test for the Distributed Systems Annex, an implementation must allow a body for System.RPC to be compiled.

Remote_Call_Interface pragma
Ada allows explicit message-based communication between active partitions as an alternative to RPC [see Ada E.2.3(20)]. If an implementation does not support the Remote_Call_Interface pragma then the following tests are not applicable:

<table>
<thead>
<tr>
<th>Test Code</th>
<th>Test Code</th>
<th>Test Code</th>
<th>Test Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>bxe2009</td>
<td>bxe2010</td>
<td>bxe2011</td>
<td>bxe2013</td>
</tr>
</tbody>
</table>
Partition Communication System

An implementation is not required to provide a PCS [see Ada E.5(27)] in order to test the Distributed Systems Annex. If no PCS is provided then the following tests are not applicable:

cxe1001 cxe4002 cxe4005
cxe2001 cxe4003 cxe4006
cxe4001 cxe5001

System.RPC

Two tests provide a body for System.RPC, and a third test checks the specification of System.RPC. An alternative declaration is allowed for package System.RPC [see Ada E.5(27.1/2)]. If an alternative declaration is used for System.RPC, the following tests are not applicable:

cxe5001 cxe5002 cxe5003

Real-Time Annex Support

Many implementations that support the Distributed Systems Annex will also support the Real-Time Annex. Test cxe4003 is designed to take advantage of Real-Time Annex features in order to better test the Distributed Systems Annex.

For implementations that do not support the Real-Time Annex, test cxe4003 must be modified. This modification consists of deleting all lines that end with the comment --RT.

Configuring Multi-Partition Tests

Some Distributed Systems Annex tests require multiple partitions to run the test, but no more than two partitions are required for running any of them. All multi-partition tests contain a main procedure for each of the two partitions. The two partitions are referred to as "A" and "B" and the main procedures for these partitions are named <test_name>_A and <test_name>_B respectively. Each test contains instructions naming the compilation units to be included in each partition. Most implementations will be primarily concerned with the main procedure and RCI packages that are to be assigned to each partition; the remainder of the partition contents will be determined by the normal dependency rules. The naming convention used in multi-partition tests aid in making the partition assignments. If the name of a compilation unit ends in "_A<optional_digit>" then it should be assigned to partition A. Compilation units with names ending in "_B<optional_digit>" should be assigned to partition B.

The following tests require that two partitions be available to run the test:

cxe1001 cxe4002 cxe4006 lxe3002*
cxe2001* cxe4003 cxe5002

cxe2002 cxe4004 cxe5003
cxe4001 cxe4005 lxe3001

(*) Tests cxe2001 and lxe3002 contain a Shared_Passive package and two active partitions. They may be configured with either two or three partitions. The two-partition configuration must have two active partitions and the Shared_Passive package may be assigned to either one of the active partitions. The three-partition configuration consists of two active partitions and a single passive partition, and the passive partition will contain the single Shared_Passive package.
Running Multi-Partition Tests

All of the multi-partition tests include the package Report in both of the active partitions. In order for the test to pass, both partitions must produce a passed message (except for lxe3002 - see special instructions for that test). If either partition produces a failed message, or if one or both partitions do not produce a passed message, the test is graded "failed".

When running the multi-partition tests it is not important which partition is started first. Generally, partition A acts as a server and partition B is a client, so starting partition A first is usually best.

In the event a test fails due to the exception Communication_Error being raised, it is permissible to rerun the test.

5.5.5.4 Tests for the Numerics Annex

Many of the tests for Annex G, Numerics, must be run in strict mode. The method for selecting strict mode is implementation dependent and not specified by the ACATS. (Note that the tests for numerical functions specified in Annex A may, but need not, be run in strict mode.) The following tests must be run in strict mode:

- cxg2003
- cxg2004
- cxg2006
- cxg2007
- cxg2008
- cxg2009
- cxg2010
- cxg2011
- cxg2012
- cxg2013
- cxg2014
- cxg2015
- cxg2016
- cxg2017
- cxg2018
- cxg2019
- cxg2020
- cxg2021

5.5.5.5 Tests that use Configuration Pragmas

Several of the tests in Annex D, Real Time Systems, Annex E, Distributed Systems, and Annex H, High Integrity Systems, use configuration pragmas. The technique for applying a configuration pragma to a test composed of multiple compilation units is implementation dependent and not specified by the ACATS. Every implementation that uses any such test in a conformity assessment must therefore take the appropriate steps, which may include modifications to the test code and/or post-compilation processing, to ensure that such a pragma is correctly applied. The following tests require special processing of the configuration pragma:

- ba15001
- bxc5001
- bnx4001
- bnx4002
- bnx4003
- bnx4004
- bnx4005
- bnx4006
- bnx4007
- bnx4008
- bnx4009
- bnx4010
- bnx4011
- bnx4012
- bnx4013
- cxd1004
- cxd1005
- cxd2001
- cxd2002
- cxd2003
- cxd2004
- cxd2005
- cxd2006
- cxd2007
- cxd2008
- cxd3001
- cxd3002
- cxd3003
- cxd4001
- cxd4002
- cxd4003
- cxd4004
- cxd4005
- cxd4006
- cxd4007
- cxd4008
- cxd4009
- cxd4010
- cxd5002
- cxd6002
- cxd6003
- cxd6004
- cxd7003
- cxd7005
- cxd7006
- cxd7007
- cxd7008
- cxd7009
- cxd7002
- cxd7004
- cxd7005
- cxd7006
- cxd7007
- cxd7008
- cxd7009
- cxd7003
- cxd7004
- cxd7005
- cxd7006
- cxd7007
- cxd7008
- cxd7009
- cxd7004
- cxd7005
- cxd7006
- cxd7007
- cxd7008
- cxd7009
5.5.6 Focus on Specific Areas

The ACATS test suite is structured to allow compiler developers and testers to use parts of the suite to focus on specific compiler feature areas.

Both the legacy tests and the modern tests tend to focus on specific language features in individual tests. The name of the test is generally a good indicator of the primary feature content of the test, as explained in the discussion of naming conventions. Beware that legacy test names have not changed, but the Ada Reference Manual organization has changed from [Ada83] to [Ada95], so some legacy test names point to the wrong clause of the Ada Standard. Further, note that the general style and approach of the modern tests creates user-oriented test situations by including a variety of features and interactions. Only the primary test focus can be indicated in the test name.

ACATS 4.0 tests are divided into core tests and Specialized Needs Annex tests. Recall that annexes A and B are part of the core language. All annex tests (including those for annexes A and B) have an 'X' as the second character of their name; Specialized Needs Annex tests have a letter between 'C' and 'H' (inclusive) corresponding to the annex designation, as the third character of the test name.

5.6 Grading Test Results

Although a single test may examine multiple language issues, ACATS test results are graded "passed", "failed", or "not applicable" as a whole.

All customized, applicable tests must be processed by an implementation. Results must be evaluated against the expected results for each class of test. Results that do not conform to expectations constitute failures. The only exceptions allowed are discussed in 5.2.5; in such cases, processing the approved modified test(s) must produce the expected behavior. Any differences from the general discussion of expected results below for executable or non-executable tests are included as explicit test conditions in test prologues.

Warning or other informational messages do not affect the pass/fail status of tests.

Expected results for executable and non-executable tests are discussed in Sections 5.6.1, 5.6.2, and 5.6.3. Tests that are non-applicable for an implementation are discussed in 5.6.4. Withdrawn tests are discussed in 5.6.5.

5.6.1 Expected Results for Executable Tests

Executable tests (classes A, C, D, E) must be processed by the compiler and any post-compilation steps (e.g., binder, partitioner) without any errors. They must be loaded into an execution target and run. Normal execution of tests results in an introductory message that summarizes the test objective, possibly some informative comments about the test progress, a final message giving pass / fail status, and graceful, silent termination. They may report "PASSED", "TENTATIVELY PASSED", "FAILED", OR "NOT APPLICABLE".

A test that fails to compile and bind, including compiling and binding any foundation code on which it depends is graded as "failed", unless the test includes features that need not be supported by all implementations. For example, an implementation may reject the declaration of a numeric type that it does not support. Allowable cases are clearly stated in the Applicability Criteria of tests. Annex M of the Ada Standard requires implementations to document such implementation-defined characteristics.
A test that reports "FAILED" is graded as "failed" unless the ACAL, and possibly the ACAA, determine that the test is not applicable for the implementation.

A test that reports "PASSED" is graded as "passed" unless the test produces the pass message but fails to terminate gracefully (e.g., crashes, hangs, raises an unexpected exception, produces an earlier or later "FAILED" message). This kind of aberrant behavior may occur, for example, in certain tasking tests, where there are multiple threads of control. A pass status message may be produced by one thread, but another thread may asynchronously crash or fail to terminate properly.

A test that reports "NOT APPLICABLE" must be run by the implementation and is graded as "not applicable" unless it produces the not-applicable message and then fails to terminate gracefully.

A test that reports "TENTATIVELY PASSED" is graded as "passed" if the test results satisfy the pass/fail criteria in the test. Normally, verification requires manual inspection of the test output.

A test that fails to report, or produces only a partial report, will be graded as "failed" unless the ACAL, and possibly the ACAA, determine that the test is not applicable for the implementation.

5.6.2 Expected Results for Class B Tests

Class B tests are expected to be compiled but are not subject to further processing and are not intended to be executable. An implementation must correctly report each clearly marked error (the notation -- ERROR: occurs at the right hand side of the source). A multiple unit B test file generally will have errors only in one compilation unit per source file. Error messages must provide some means of specifying the location of an error, but they are not required to be in direct proximity with the -- ERROR: marking of the errors.

Some B-tests also include the notation -- OK to indicate constructs that must not be identified as errors. Such constructs are typically similar to illegal constructs and serve to ensure that implementations do not reject too much. Not identifying -- OK constructs as errors is especially important since some constructs that were errors in [Ada83] are now legal in later versions of Ada.

Some B-tests exercise constructs whose correctness depends on source code that is textually separated (for example, a deferred constant and its full declaration). In these cases, it may be reasonable to report an error at both locations. Such cases are marked with -- OPTIONAL ERROR. These lines may be flagged as errors by some, but not all, implementations. Unless an optional error is marked as an error for the wrong reason, an error report (or lack of it) does not affect the pass/fail status of the test.

Some B-tests contain constructs where it would be reasonable for a compiler to report an error at one of several source locations. When it would not be appropriate for the ACATS to insist on a particular source location, all such source locations are marked with -- POSSIBLE ERROR: and an indication of which error set (if the test contains several) the location belongs to. In such cases, an implementation is considered to have properly reported the error if it reports an error at any of the places marked -- POSSIBLE ERROR: for a particular set. The implementation may flag more than one such place; this does not affect the pass/fail status of the test. However, the test is graded "failed" if no error is reported at any of the places marked -- POSSIBLE ERROR: for an error set.

A test is graded as "passed" if it reports each error in the test. The content of error messages is considered only to determine that they are indeed indications of errors (as opposed to warnings) and that they refer to the expected errors. The Reference Manual does not specify the form or content of error messages. In particular, a test with just one expected error is graded as "passed" if the test is rejected at compile time for any reason.

A test is graded as "failed" if it fails to report on each error in the test or if it marks legal code as incorrect.
5.6.3 Expected Results for Class L Tests

Class L tests are expected to be rejected before execution begins. They must be submitted to the compiler and to the linker/binder. If an executable is generated, then it must be submitted for execution. Unless otherwise documented, the test is graded as "failed" if it begins execution, regardless of whether any output is produced. (Twenty-eight L tests contain documentation indicating that they may execute. See below.)

In general, an L test is expected to be rejected at link/bind time. Some tests contain `ERROR:` indications; an implementation that reports an error associated with one of these lines is judged to have passed the test (provided, of course, that the link attempt fails).

The following tests are exceptions to the general rule that an L test must not execute:

- Test LXE3002, for the Distributed Systems Annex, is a test that has two partitions, each of which may execute. As documented in the source code, this test is graded "failed" if both partitions report "TENTATIVELY PASSED". Other outcomes are graded as appropriate for Class L tests.
- Tests LA14001..27 and LA20002 (twenty-seven core language tests), as documented in the source code, may execute if automatic recompilation is supported. These tests are graded as "passed" if they execute and report "PASSED". Other outcomes are graded as appropriate for Class L tests.

5.6.4 Inapplicable Tests

Each ACATS test has a test objective that is described in the test prologue. Some objectives address Ada language features that need not be supported by every Ada implementation (e.g., "check floating-point operations for digits 18"). These test programs generally also contain an explicit indication of their applicability and the expected behavior of an implementation for which they do not apply. Annex D, “Test Applicability Criteria” lists common reasons for a test to be inapplicable, and lists the tests affected.

A test may be inapplicable for an implementation given:

- appropriate ACATS grading criteria; or
- an ACAA ruling on a petition to accept a deviation from expected results.

Appropriate grading criteria include:

1. whether a test completes execution and reports "NOT APPLICABLE";
2. whether a test is rejected at compile or bind time for a reason that satisfies grading criteria stated in the test program.

All applicable test programs must be processed and passed.

5.6.5 Withdrawn Tests

From time to time, the ACAA determines that one or more tests included in a release of the ACATS should be withdrawn from the test suite. Tests that are withdrawn are not processed during a conformity assessment and are not considered when grading an implementation.

Usually, a test is withdrawn because an error has been discovered in it. A withdrawn test will not be reissued as a modified test, although it may be revised and reissued as a new test in the future.

Withdrawn tests are listed in the ACATS Modification List, which is maintained by the ACAA.
5.7 Addressing Problems or Issues

After all tests have been processed and graded, any remaining problems should be addressed. Test failures must be identified and resolved. This section discusses issues that are not due to implementation errors (bugs).

5.7.1 Typical Issues

Here are some typical causes of unexpected ACATS test failures (often resulting from clerical errors):

- Processing a test that is withdrawn;
- Processing the original version of a test that has been modified by the ACAA to correct a test error;
- Processing a test that is not applicable to the implementation (as explained in Section 5.6.4);
- Processing files (or tests, see Section 5.4.2) in an incorrect order;
- Processing tests when units required in the environment are not present.

Test result failures resulting from technical errors may include:

- Incorrect values in ImpDef, which provide inappropriate values to tests at run-time (refer to 5.2.1);
- Incorrect values in macro.dfs, which result in incorrectly customized tests (refer to 5.2.2);
- Need to modify a test (e.g., split a B-test).

Finally, occasionally a user discovers an error in a new ACATS test. More rarely, errors are uncovered by compiler advances in tests that are apparently stable. In either case, if users believe that a test is in error, they may file a dispute with the ACAL. The dispute process is described in the next section.

5.7.2 Deviation from Expected Results - Petition & Review

Each test indicates in its prologue what it expects from a conforming implementation. The result of processing a test is acceptable if and only if the result is explicitly allowed by the grading criteria for the test.

A user may challenge an ACATS test on the grounds of applicability or correctness. A challenger should submit a petition against the test program to an ACAL or to the ACAA, following the procedure and the format presented in [Pro31]. A petition must clearly state whether it is a claim that the test does not apply to the implementation or that the test is incorrect. The petition must indicate the specific section of code that is disputed and provide a full explanation of the reason for the dispute.

ACALs will forward petitions from their customers to the ACAA for decisions. The ACAA will evaluate the petitioner's claims and decide whether:

- the test is applicable to the implementation (i.e., deviation *is not* allowed);
- the test is not applicable to the implementation (i.e., deviation *is* allowed);
- the test should be repaired (deviation is allowed, and the modified test should be used for determining conformity assessment results);
- the test should be withdrawn (deviation is allowed and the test is not considered in determining conformity assessment results).
A deviation is considered to be a test failure unless a petition to allow the deviation has been accepted by the ACAA.

5.8 Reprocessing and Regrading

After all problems have been resolved, tests that failed can be reprocessed and regraded. This step completes the ACATS testing process.
Annex A: Version Description

ACATS 4.0 includes 3910 tests in 4537 files, not including foundation and other support units. From Version 3.1 to 4.0, 118 tests were added, comprising 129 files (including three foundation files). 29 tests (30 files) were modified. Three support files also were modified. 6 tests (6 files) and one foundation were removed. 11 documentation files were added, 151 documentation files were modified, and 6 documentation files were deleted.

The following sections present a detailed description of ACATS 4.0, as follows:

A.1, “Core Test Files”
A.2, “Specialized Needs Annex Test Files”
A.3, “Foundation Code Files”
A.4, “Documentation Files”
A.5, “Other Files”
A.6, “Tests With Special Requirements”
A.7, “Test Files Added Since ACATS 3.1”
A.8, “Documentation Files Added Since ACATS 3.1”
A.9, “Test Files Modified Since ACATS 3.1”
A.11, “Documentation Files Modified Since ACATS 3.1”
A.10, “Support Files Modified Since ACATS 3.1”
A.12, “Test Files Deleted Since ACATS 3.1”
A.13, “Documentation Files Deleted Since ACATS 3.1”

A.1 Core Test Files

The following files contain the tests for core language features of Ada; that is, for requirements specified in Sections 2 through 13, Annex A, Annex B and Annex J.

a22006b.ada       a49027b.ada       a83009a.ada       aa2010a.ada
a22006c.ada       a49027c.ada       a83009b.ada       aa2012a.ada
a22006d.ada       a54b01a.ada       a83a02a.ada       ac1015b.ada
a26007a.tst       a54b02a.ada       a83a02b.ada       ac3106a.ada
a27003a.ada       a55b12a.ada       a83a06a.ada       ac3206a.ada
a29003a.ada       a55b13a.ada       a83a08a.ada       ac3207a.ada
a2a031a.ada       a55b14a.ada       a83c01c.ada       ad7001b.ada
a33003a.ada       a71004a.ada       a83c01h.ada       ad7001c0.ada
a34017c.ada       a73001i.ada       a83c01i.ada       ad7001c1.ada
a35101b.ada       a73001j.ada       a85007d.ada       ad7001d0.ada
a35402a.ada       a74105b.ada       a85013b.ada       ad7001d1.ada
a35801f.ada       a74106a.ada       a87b59a.ada       ad7006a.ada
a35902c.ada       a74106b.ada       a95001c.ada       ad7101a.ada
a38106d.ada       a74106c.ada       a95074d.ada       ad7101c.ada
a38106e.ada       a74205e.ada       a97106a.ada       ad7102a.ada
a49027a.ada       a74205f.ada       a99006a.ada       ad7103a.ada
ad7103c.ada  b25002a.ada  b324003.a  b34014i.ada
ad7104a.ada  b25002b.ada  b330001.a  b34014m.ada
ad7201a.ada  b26001a.ada  b33001a.ada  b34014o.ada
ad7203b.ada  b26002a.ada  b33101a.ada  b34014q.ada
ad7205b.ada  b26005a.ada  b33102a.ada  b34014s.ada
ad8011a.tst  b28001a.ada  b33102b.ada  b34014v.ada
ada101a.ada  b28001b.ada  b33102c.ada  b34014z.ada
ae2113a.ada  b28001c.ada  b33102d.ada  b35004a.ada
ae2113b.ada  b28001d.ada  b33102e.ada  b35101a.ada
ae3002g.ada  b28001e.ada  b33201a.ada  b35103a.ada
ae3101a.ada  b28001f.ada  b33201b.ada  b35103b.ada
ae3702a.ada  b28001g.ada  b33201c.ada  b35302a.ada
ae3709a.ada  b28001h.ada  b33201d.ada  b354001.a
b22001a.tst  b28001i.ada  b33201e.ada  b35401a.ada
b22001b.tst  b28001j.ada  b33204a.ada  b35401b.ada
b22001c.tst  b28001k.ada  b33205a.ada  b35403a.ada
b22001d.tst  b28001l.ada  b33302a.ada  b35501a.ada
b22001e.tst  b28001m.ada  b34001b.ada  b35501b.ada
b22001f.tst  b28001n.ada  b34001e.ada  b35506a.ada
b22001g.tst  b28001o.ada  b34002b.ada  b35506b.ada
b22001h.ada  b28001p.ada  b34003b.ada  b35506c.ada
b22001i.tst  b28001q.ada  b34004b.ada  b35506d.ada
b22001j.tst  b28001r.ada  b34005b.ada  b35701a.ada
b22001k.tst  b28001s.ada  b34005e.ada  b35709a.ada
b22001l.tst  b28001t.ada  b34005h.ada  b35901a.ada
b22001m.tst  b28001u.ada  b34005k.ada  b35901c.ada
b22001n.tst  b28001v.ada  b34005n.ada  b35901d.ada
b23002a.ada  b28001w.ada  b34005q.ada  b35a01a.ada
b23004a.ada  b29001a.ada  b34005t.ada  b35a08a.ada
b23004b.ada  b2a003a.ada  b34006b.ada  b36001a.ada
b24001a.ada  b2a003b.ada  b34006c.ada  b36002a.ada
b24001b.ada  b2a003c.ada  b34006h.ada  b36101a.ada
b24001c.ada  b2a003d.ada  b34006k.ada  b36102a.ada
b24005a.ada  b2a003e.ada  b34007b.ada  b36103a.ada
b24005b.ada  b2a003f.ada  b34007e.ada  b36105c.dep
b24007a.ada  b2a005a.ada  b34007h.ada  b36171a.ada
b24009a.ada  b2a005b.ada  b34007k.ada  b36171b.ada
b24009b.ada  b2a007a.ada  b34007n.ada  b36171c.ada
b24104a.ada  b2a010a.ada  b34007q.ada  b36171d.ada
b24204a.ada  b2a021a.ada  b34007t.ada  b36171e.ada
b24204b.ada  b32103a.ada  b34008b.ada  b36171f.ada
b24204c.ada  b32104a.ada  b34009b.ada  b36171g.ada
b24204d.ada  b32106a.ada  b34009e.ada  b36171h.ada
b24204e.ada  b32201a.ada  b34009h.ada  b36171i.ada
b24204f.ada  b32202a.ada  b34009k.ada  b36201a.ada
b24205a.ada  b32202b.ada  b34011a.ada  b36307a.ada
b24206a.ada  b32202c.ada  b34014b.ada  b370001.a
b24206b.ada  b324001.a  b34014d.ada  b370002.a
b24211b.ada  b324002.a  b34014f.ada  b37004a.ada
<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>b45208b.ada</td>
<td>b46004c.ada</td>
<td>b53001a.ada</td>
<td>b55b09d.dep</td>
</tr>
<tr>
<td>b45208c.ada</td>
<td>b46004d.ada</td>
<td>b53001b.ada</td>
<td>b55b12b.ada</td>
</tr>
<tr>
<td>b45208g.ada</td>
<td>b46004e.ada</td>
<td>b53002a.ada</td>
<td>b55b12c.ada</td>
</tr>
<tr>
<td>b45208h.ada</td>
<td>b46005a.ada</td>
<td>b53002b.ada</td>
<td>b55b17a.ada</td>
</tr>
<tr>
<td>b45208l.ada</td>
<td>b47001a.ada</td>
<td>b53009a.ada</td>
<td>b55b17b.ada</td>
</tr>
<tr>
<td>b45208m.ada</td>
<td>b48001a.ada</td>
<td>b53009b.ada</td>
<td>b55b17c.ada</td>
</tr>
<tr>
<td>b45208n.ada</td>
<td>b48001a.ada</td>
<td>b53009c.ada</td>
<td>b55b18a.ada</td>
</tr>
<tr>
<td>b45208s.ada</td>
<td>b48001b.ada</td>
<td>b54001a.ada</td>
<td>b56001a.ada</td>
</tr>
<tr>
<td>b45208t.ada</td>
<td>b48002a.ada</td>
<td>b54002a.ada</td>
<td>b56001b.ada</td>
</tr>
<tr>
<td>b45209a.ada</td>
<td>b48002b.ada</td>
<td>b54a01b.ada</td>
<td>b56001c.ada</td>
</tr>
<tr>
<td>b45209b.ada</td>
<td>b48002c.ada</td>
<td>b54a01f.ada</td>
<td>b56001d.ada</td>
</tr>
<tr>
<td>b45209c.ada</td>
<td>b48002d.ada</td>
<td>b54a01g.ada</td>
<td>b56001e.ada</td>
</tr>
<tr>
<td>b45209d.ada</td>
<td>b48002e.ada</td>
<td>b54a01l.ada</td>
<td>b56001f.ada</td>
</tr>
<tr>
<td>b45209e.ada</td>
<td>b48002g.ada</td>
<td>b54a05a.ada</td>
<td>b56001g.ada</td>
</tr>
<tr>
<td>b45209f.ada</td>
<td>b48003a.ada</td>
<td>b54a05b.ada</td>
<td>b56001h.ada</td>
</tr>
<tr>
<td>b45209g.ada</td>
<td>b48003b.ada</td>
<td>b54a10a.ada</td>
<td>b57001a.ada</td>
</tr>
<tr>
<td>b45209h.ada</td>
<td>b48003d.ada</td>
<td>b54a12a.ada</td>
<td>b57001b.ada</td>
</tr>
<tr>
<td>b45209i.ada</td>
<td>b48003e.ada</td>
<td>b54a20a.ada</td>
<td>b57001c.ada</td>
</tr>
<tr>
<td>b45209j.ada</td>
<td>b490001a.ada</td>
<td>b54a21a.ada</td>
<td>b58003a.ada</td>
</tr>
<tr>
<td>b45209k.ada</td>
<td>b490002a.ada</td>
<td>b54a25a.ada</td>
<td>b59001a.ada</td>
</tr>
<tr>
<td>b45221a.ada</td>
<td>b490003a.ada</td>
<td>b54a60a.ada</td>
<td>b59001c.ada</td>
</tr>
<tr>
<td>b45261a.ada</td>
<td>b49002a.ada</td>
<td>b54a60b.ada</td>
<td>b59001d.ada</td>
</tr>
<tr>
<td>b45261b.ada</td>
<td>b49004a.ada</td>
<td>b54b01b.tst</td>
<td>b59001e.ada</td>
</tr>
<tr>
<td>b45261c.ada</td>
<td>b49005a.ada</td>
<td>b54b01c.ada</td>
<td>b59001f.ada</td>
</tr>
<tr>
<td>b45261d.ada</td>
<td>b49007a.ada</td>
<td>b54b02b.ada</td>
<td>b59001g.ada</td>
</tr>
<tr>
<td>b45301a.ada</td>
<td>b49007b.ada</td>
<td>b54b02c.ada</td>
<td>b59001h.ada</td>
</tr>
<tr>
<td>b45301b.ada</td>
<td>b49008a.ada</td>
<td>b54b02d.ada</td>
<td>b59001i.ada</td>
</tr>
<tr>
<td>b45301c.ada</td>
<td>b49008c.ada</td>
<td>b54b04a.ada</td>
<td>b610001.a</td>
</tr>
<tr>
<td>b45302a.ada</td>
<td>b49009b.ada</td>
<td>b54b04b.ada</td>
<td>b610002.a</td>
</tr>
<tr>
<td>b45341a.ada</td>
<td>b49009c.ada</td>
<td>b54b05a.ada</td>
<td>b61001f.ada</td>
</tr>
<tr>
<td>b455002.a</td>
<td>b49010a.ada</td>
<td>b54b06a.ada</td>
<td>b61005a.ada</td>
</tr>
<tr>
<td>b45501a.ada</td>
<td>b49011a.ada</td>
<td>b552a01a.ada</td>
<td>b61006a.ada</td>
</tr>
<tr>
<td>b45501b.ada</td>
<td>b4a010c.ada</td>
<td>b552a02a.ada</td>
<td>b61011a.ada</td>
</tr>
<tr>
<td>b45501c.ada</td>
<td>b4a016a.ada</td>
<td>b552a03a.ada</td>
<td>b62001a.ada</td>
</tr>
<tr>
<td>b45522a.ada</td>
<td>b51001a.ada</td>
<td>b55a01a.ada</td>
<td>b62001b.ada</td>
</tr>
<tr>
<td>b45537a.ada</td>
<td>b51004b.ada</td>
<td>b55a01d.ada</td>
<td>b62001c.ada</td>
</tr>
<tr>
<td>b45601a.ada</td>
<td>b51004c.ada</td>
<td>b55a01e.ada</td>
<td>b62001d.ada</td>
</tr>
<tr>
<td>b45625a.ada</td>
<td>b52002a.ada</td>
<td>b55a01j.ada</td>
<td>b63001a.ada</td>
</tr>
<tr>
<td>b45661a.ada</td>
<td>b52002b.ada</td>
<td>b55a01k.ada</td>
<td>b63001b.ada</td>
</tr>
<tr>
<td>b457001.a</td>
<td>b52002c.ada</td>
<td>b55a01l.ada</td>
<td>b63005a.ada</td>
</tr>
<tr>
<td>b460001.a</td>
<td>b52002d.ada</td>
<td>b55a01n.ada</td>
<td>b63005b.ada</td>
</tr>
<tr>
<td>b460002.a</td>
<td>b52002e.ada</td>
<td>b55a01o.ada</td>
<td>b63006a.ada</td>
</tr>
<tr>
<td>b460004.a</td>
<td>b52002f.ada</td>
<td>b55a01t.ada</td>
<td>b63009a.ada</td>
</tr>
<tr>
<td>b460005.a</td>
<td>b52002g.ada</td>
<td>b55a01u.ada</td>
<td>b63009b.ada</td>
</tr>
<tr>
<td>b460006.a</td>
<td>b52004a.ada</td>
<td>b55a01v.ada</td>
<td>b63009c0.ada</td>
</tr>
<tr>
<td>b460002a.ada</td>
<td>b52004b.ada</td>
<td>b55b01a.ada</td>
<td>b63009c1.ada</td>
</tr>
<tr>
<td>b460003a.ada</td>
<td>b52004c.ada</td>
<td>b55b01b.ada</td>
<td>b63009c2.ada</td>
</tr>
<tr>
<td>b460004a.ada</td>
<td>b52004d.dep</td>
<td>b55b09b.ada</td>
<td>b63009c3.ada</td>
</tr>
<tr>
<td>b46004b.ada</td>
<td>b52004e.dep</td>
<td>b55b09c.dep</td>
<td>b63103a.ada</td>
</tr>
<tr>
<td>b83a01c.ada</td>
<td>b850011.ada</td>
<td>b91004a.ada</td>
<td>b95082b.ada</td>
</tr>
<tr>
<td>b83a05a.ada</td>
<td>b85002a.ada</td>
<td>b91005a.ada</td>
<td>b95082c.ada</td>
</tr>
<tr>
<td>b83a06b.ada</td>
<td>b85003a.ada</td>
<td>b92001a.ada</td>
<td>b95082d.ada</td>
</tr>
<tr>
<td>b83a06h.ada</td>
<td>b85003b.ada</td>
<td>b92001b.ada</td>
<td>b95082e.ada</td>
</tr>
<tr>
<td>b83a07a.ada</td>
<td>b85004a.ada</td>
<td>b940001.a</td>
<td>b95082f.ada</td>
</tr>
<tr>
<td>b83a07b.ada</td>
<td>b85008f.ada</td>
<td>b940002.a</td>
<td>b95083a.ada</td>
</tr>
<tr>
<td>b83a07c.ada</td>
<td>b85008g.ada</td>
<td>b940003.a</td>
<td>b95094a.ada</td>
</tr>
<tr>
<td>b83a08b.ada</td>
<td>b85008h.ada</td>
<td>b940004.a</td>
<td>b95094b.ada</td>
</tr>
<tr>
<td>b83a09a.ada</td>
<td>b85010a.ada</td>
<td>b940005.a</td>
<td>b95094c.ada</td>
</tr>
<tr>
<td>b83b01a.ada</td>
<td>b85010b.ada</td>
<td>b940006.a</td>
<td>b951001.a</td>
</tr>
<tr>
<td>b83b02c.ada</td>
<td>b85012a.ada</td>
<td>b940007.a</td>
<td>b951002.a</td>
</tr>
<tr>
<td>b83e01a.ada</td>
<td>b85013c.ada</td>
<td>b95001.a</td>
<td>b952001.a</td>
</tr>
<tr>
<td>b83e01b.ada</td>
<td>b85013d.ada</td>
<td>b95002.a</td>
<td>b952002.a</td>
</tr>
<tr>
<td>b83e01c.ada</td>
<td>b85015a.ada</td>
<td>b95001a.ada</td>
<td>b952003.a</td>
</tr>
<tr>
<td>b83e01d.ada</td>
<td>b8510010.a</td>
<td>b95001b.ada</td>
<td>b952004.a</td>
</tr>
<tr>
<td>b83e01e0.ada</td>
<td>b8510011.a</td>
<td>b95001d.ada</td>
<td>b954001.a</td>
</tr>
<tr>
<td>b83e01e1.ada</td>
<td>b8510012.am</td>
<td>b95002a.ada</td>
<td>b954003.a</td>
</tr>
<tr>
<td>b83e01e2.ada</td>
<td>b851002.a</td>
<td>b95003.a</td>
<td>b954004.a</td>
</tr>
<tr>
<td>b83e01e3.ada</td>
<td>b851003.a</td>
<td>b95004.a</td>
<td>b954005.a</td>
</tr>
<tr>
<td>b83e01f0.ada</td>
<td>b851004.a</td>
<td>b9504b.ada</td>
<td>b960001.a</td>
</tr>
<tr>
<td>b83e01f1.ada</td>
<td>b854001.a</td>
<td>b9506a.ada</td>
<td>b96002.a</td>
</tr>
<tr>
<td>b83e01f2.ada</td>
<td>b86001a0.ada</td>
<td>b9506b.ada</td>
<td>b97102.b</td>
</tr>
<tr>
<td>b83e01f3.ada</td>
<td>b86001a1.ada</td>
<td>b9506c.ada</td>
<td>b97102.c</td>
</tr>
<tr>
<td>b83e01f4.ada</td>
<td>b87b23b.ada</td>
<td>b9506d.ada</td>
<td>b97102.d</td>
</tr>
<tr>
<td>b83e01f5.ada</td>
<td>b87b26a.ada</td>
<td>b9507a.ada</td>
<td>b97102.f</td>
</tr>
<tr>
<td>b83e01f6.ada</td>
<td>b87b48c.ada</td>
<td>b9507b.ada</td>
<td>b97102.g</td>
</tr>
<tr>
<td>b83e11a.ada</td>
<td>b91001b.ada</td>
<td>b95020.a</td>
<td>b97102.h</td>
</tr>
<tr>
<td>b83f02a.ada</td>
<td>b91001c.ada</td>
<td>b95020b.ada</td>
<td>b97102.i</td>
</tr>
<tr>
<td>b83f02b.ada</td>
<td>b91001d.ada</td>
<td>b95020b1.a.</td>
<td>b97103.a</td>
</tr>
<tr>
<td>b83f02c.ada</td>
<td>b91001e.ada</td>
<td>b95020b2.ada</td>
<td>b97103.b</td>
</tr>
<tr>
<td>b840001.a</td>
<td>b91001f.ada</td>
<td>b95030.a</td>
<td>b97103.d</td>
</tr>
<tr>
<td>b84001.a</td>
<td>b91001g.ada</td>
<td>b95031.a</td>
<td>b97103.e</td>
</tr>
<tr>
<td>b84002.a</td>
<td>b91002a.ada</td>
<td>b95032.a</td>
<td>b97103.f</td>
</tr>
<tr>
<td>b84004.a</td>
<td>b91002b.ada</td>
<td>b95061.a</td>
<td>b97103.g</td>
</tr>
<tr>
<td>b84005.a</td>
<td>b91002c.ada</td>
<td>b95061b.ada</td>
<td>b97104.a</td>
</tr>
<tr>
<td>b84006.a</td>
<td>b91002d.ada</td>
<td>b95061c.ada</td>
<td>b97104.b</td>
</tr>
<tr>
<td>b84007.a</td>
<td>b91002e.ada</td>
<td>b95061d.ada</td>
<td>b97104.c</td>
</tr>
<tr>
<td>b84008.b</td>
<td>b91002f.ada</td>
<td>b95061e.ada</td>
<td>b97104.d</td>
</tr>
<tr>
<td>b85001.a</td>
<td>b91002g.ada</td>
<td>b95061f.ada</td>
<td>b97104.e</td>
</tr>
<tr>
<td>b85001.b</td>
<td>b91002h.ada</td>
<td>b95061g.ada</td>
<td>b97104.f</td>
</tr>
<tr>
<td>b85001.c</td>
<td>b91002i.ada</td>
<td>b95062.a</td>
<td>b97104.g</td>
</tr>
<tr>
<td>b85001.d</td>
<td>b91002j.ada</td>
<td>b95063.a</td>
<td>b97107.a</td>
</tr>
<tr>
<td>b85001.e</td>
<td>b91002k.ada</td>
<td>b95064.a</td>
<td>b97108.a</td>
</tr>
<tr>
<td>b85001.f</td>
<td>b91002l.ada</td>
<td>b95068.a</td>
<td>b97108.b</td>
</tr>
<tr>
<td>b85001.g</td>
<td>b91003.a</td>
<td>b95070.a</td>
<td>b97109.a</td>
</tr>
<tr>
<td>b85001.h</td>
<td>b91003a.ada</td>
<td>b95080.a</td>
<td>b97110.a</td>
</tr>
<tr>
<td>b85001.i</td>
<td>b91003c.ada</td>
<td>b95080c.a</td>
<td>b97110.b</td>
</tr>
<tr>
<td>b85001.j</td>
<td>b91003d.ada</td>
<td>b95081.a</td>
<td>b97111.a</td>
</tr>
<tr>
<td>b85001k.ada</td>
<td>b91003e.ada</td>
<td>b95082.a</td>
<td>b99001.a</td>
</tr>
</tbody>
</table>
ACATS 4.0 User's Guide

A.1 Core Test Files 29 August 2014

ba12008.a  ba120145.a  ba2013a.ada  bb2002a.ada
ba120090.a ba120146.a  ba2013b.ada  bb2003a.ada
ba120091.a ba120147.a  ba21002.a  bb2003b.ada
ba120092.a ba120150.a  ba210030.a bb2003c.ada
ba120093.a ba120151.a  ba210031.a bb3001a.ada
ba120094.a ba120152.a  ba210032.a bb3002a.ada
ba120095.a ba120153.a  ba210033.a bc1001a.ada
ba120096.a ba120154.a  ba210034.a bc1002a.ada
ba120097.a ba120160.a  ba210035.a bc1005a.ada
ba120098.a ba120161.a  ba210040.a bc1008a.ada
ba120100.a ba120162.a  ba210041.a bc1008b.ada
ba120101.a ba120163.a  ba210042.a bc1008c.ada
ba120102.a ba120164.a  ba210043.a bc1009a.ada
ba120103.a ba120165.a  ba210044.a bc1011a.ada
ba120104.a ba120166.a  ba210045.am bc1011b.ada
ba120105.a ba13b01.a   ba21a01.a  bc1011c.ada
ba120110.a ba13b02.a   ba21a02.a  bc1012a.ada
ba120111.a ba15001.a   ba21a03.a  bc1013a.ada
ba120112.a ba150020.a  ba3001a0.a  bc1014a.ada
ba120113.a ba150021.a  ba3001a1.a  bc1014b.ada
ba120114.a ba150022.a  ba3001a2.a  bc1016a.ada
ba120115.a ba150023.a  ba3001a3.a  bc1016b.ada
ba120120.a ba150024.a  ba3001b0.a  bc1101a.ada
ba120121.a ba150025.a  ba3001b1.a  bc1102a.ada
ba120122.a ba150026.a  ba3001c0.a  bc1103a.ada
ba120123.a ba150027.a  ba3001c1.a  bc1106a.ada
ba120124.a ba150028.a  ba3001e0.a  bc1107a.ada
ba120125.a ba150029.am ba3001e1.a  bc1109a.ada
ba120126.a ba16001.a   ba3001e2.a  bc1109b.ada
ba120127.a ba16002.a   ba3001e3.a  bc1109c.ada
ba120128.a ba2001a.ada  ba3001f0.a  bc1109d.ada
ba120129.a ba2001b.ada  ba3001f1.a  bc1110a.ada
ba12012a.a ba2001c.ada  ba3001f2.a  bc1201a.ada
ba120130.a ba2001d.ada  ba3001f3.a  bc1201b.ada
ba120131.a ba2001f0.ada ba3006a0.a  bc1201c.ada
ba120132.a ba2001f1.ada ba3006a1.a  bc1201d.ada
ba120133.a ba2001f2.ada ba3006a2.a  bc1201e.ada
ba120134.a ba2003b0.ada ba3006a3.a  bc1201f.ada
ba120135.a ba2003b1.ada ba3006a4.a  bc1201g.ada
ba120136.a ba2011a0.ada ba3006a5.a  bc1201h.ada
ba120137.a ba2011a1.ada ba3006a6.a  bc1201i.ada
ba120138.a ba2011a2.ada ba3006b0.a  bc1201j.ada
ba120139.a ba2011a3.ada ba3006b1.a  bc1201k.ada
ba12013a.a ba2011a4.ada ba3006b2.a  bc1201l.ada
ba120140.a ba2011a5.ada ba3006b3.a  bc1202a.ada
ba120141.a ba2011a6.ada ba3006b4.a  bc1202c.ada
ba120142.a ba2011a7.ada bb10001.a  bc1202f.ada
ba120143.a ba2011a8.ada bb20001.a  bc1202g.ada
ba120144.a ba2011a9.ada bb2001a.ada  bc1203a.ada
<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>bc1205a.ada</td>
<td>bc3205c.ada</td>
<td>bc3502n.ada</td>
<td>bc54a03.a</td>
</tr>
<tr>
<td>bc1206a.ada</td>
<td>bc3301a.ada</td>
<td>bc3502o.ada</td>
<td>bc54a04.a</td>
</tr>
<tr>
<td>bc1207a.ada</td>
<td>bc3301b.ada</td>
<td>bc3503a.ada</td>
<td>bc54a05.a</td>
</tr>
<tr>
<td>bc1208a.ada</td>
<td>bc3302a.ada</td>
<td>bc3503c.ada</td>
<td>bc54a06.a</td>
</tr>
<tr>
<td>bc1226a.ada</td>
<td>bc3302b.ada</td>
<td>bc3503d.ada</td>
<td>bc70001.a</td>
</tr>
<tr>
<td>bc1230a.ada</td>
<td>bc3303a.ada</td>
<td>bc3503e.ada</td>
<td>bc70002.a</td>
</tr>
<tr>
<td>bc1303a.ada</td>
<td>bc3304a.ada</td>
<td>bc3503f.ada</td>
<td>bc70003.a</td>
</tr>
<tr>
<td>bc1303b.ada</td>
<td>bc3401a.ada</td>
<td>bc3604a.ada</td>
<td>bc70004.a</td>
</tr>
<tr>
<td>bc1303c.ada</td>
<td>bc3401b.ada</td>
<td>bc3604b.ada</td>
<td>bc70005.a</td>
</tr>
<tr>
<td>bc1303d.ada</td>
<td>bc3402a.ada</td>
<td>bc3607a.ada</td>
<td>bc70006.a</td>
</tr>
<tr>
<td>bc1303e.ada</td>
<td>bc3402b.ada</td>
<td>bc40002.a</td>
<td>bc70007.a</td>
</tr>
<tr>
<td>bc1303f.ada</td>
<td>bc3403a.ada</td>
<td>bc50001.a</td>
<td>bc70008.a</td>
</tr>
<tr>
<td>bc1303g.ada</td>
<td>bc3403b.ada</td>
<td>bc50002.a</td>
<td>bc70009.a</td>
</tr>
<tr>
<td>bc1306a.ada</td>
<td>bc3403c.ada</td>
<td>bc50003.a</td>
<td>bc70010.a</td>
</tr>
<tr>
<td>bc2001b.ada</td>
<td>bc3404a.ada</td>
<td>bc50004.a</td>
<td>bd11001.a</td>
</tr>
<tr>
<td>bc2001c.ada</td>
<td>bc3404b.ada</td>
<td>bc51002.a</td>
<td>bd1b01a.ada</td>
</tr>
<tr>
<td>bc2001d.ada</td>
<td>bc3404c.ada</td>
<td>bc51003.a</td>
<td>bd1b02h.ada</td>
</tr>
<tr>
<td>bc2001e.ada</td>
<td>bc3404d.ada</td>
<td>bc51004.a</td>
<td>bd1b03c.ada</td>
</tr>
<tr>
<td>bc2004a.ada</td>
<td>bc3404e.ada</td>
<td>bc51005.a</td>
<td>bd1b05e.ada</td>
</tr>
<tr>
<td>bc2004b.ada</td>
<td>bc3404f.ada</td>
<td>bc51006.a</td>
<td>bd1b06j.ada</td>
</tr>
<tr>
<td>bc30001.a</td>
<td>bc3405a.ada</td>
<td>bc51007.a</td>
<td>bd2001b.ada</td>
</tr>
<tr>
<td>bc3001a.ada</td>
<td>bc3405b.ada</td>
<td>bc51011.a</td>
<td>bd2a01h.ada</td>
</tr>
<tr>
<td>bc3002a.ada</td>
<td>bc3405d.ada</td>
<td>bc51012.a</td>
<td>bd2a02a.tst</td>
</tr>
<tr>
<td>bc3002b.ada</td>
<td>bc3405e.ada</td>
<td>bc51013.a</td>
<td>bd2a03a.ada</td>
</tr>
<tr>
<td>bc3002c.ada</td>
<td>bc3405f.ada</td>
<td>bc51015.a</td>
<td>bd2a03b.ada</td>
</tr>
<tr>
<td>bc3002d.ada</td>
<td>bc3501a.ada</td>
<td>bc51016.a</td>
<td>bd2a06a.ada</td>
</tr>
<tr>
<td>bc3002e.ada</td>
<td>bc3501b.ada</td>
<td>bc51017.a</td>
<td>bd2a25a.ada</td>
</tr>
<tr>
<td>bc3005a.ada</td>
<td>bc3501c.ada</td>
<td>bc51018.a</td>
<td>bd2a35a.ada</td>
</tr>
<tr>
<td>bc3005b.ada</td>
<td>bc3501d.ada</td>
<td>bc51019.a</td>
<td>bd2a45a.ada</td>
</tr>
<tr>
<td>bc3005c.ada</td>
<td>bc3501e.ada</td>
<td>bc51020.a</td>
<td>bd2a55a.ada</td>
</tr>
<tr>
<td>bc3005d.ada</td>
<td>bc3501f.ada</td>
<td>bc510210.a</td>
<td>bd2a55b.ada</td>
</tr>
<tr>
<td>bc3006a.ada</td>
<td>bc3501g.ada</td>
<td>bc510211.a</td>
<td>bd2a67a.ada</td>
</tr>
<tr>
<td>bc3009c.ada</td>
<td>bc3501h.ada</td>
<td>bc510212.a</td>
<td>bd2a77a.ada</td>
</tr>
<tr>
<td>bc3011b.ada</td>
<td>bc3501i.ada</td>
<td>bc510213.am</td>
<td>bd2a85a.ada</td>
</tr>
<tr>
<td>bc3013a.ada</td>
<td>bc3501j.ada</td>
<td>bc510220.a</td>
<td>bd2a85b.ada</td>
</tr>
<tr>
<td>bc3016g.ada</td>
<td>bc3501k.ada</td>
<td>bc510221.a</td>
<td>bd2b01c.ada</td>
</tr>
<tr>
<td>bc3018a.ada</td>
<td>bc3502a.ada</td>
<td>bc510222.a</td>
<td>bd2b02a.ada</td>
</tr>
<tr>
<td>bc3101a.ada</td>
<td>bc3502b.ada</td>
<td>bc510223.am</td>
<td>bd2b03a.ada</td>
</tr>
<tr>
<td>bc3101b.ada</td>
<td>bc3502c.ada</td>
<td>bc51b01.a</td>
<td>bd2b03b.ada</td>
</tr>
<tr>
<td>bc3102a.ada</td>
<td>bc3502d.ada</td>
<td>bc51b02.a</td>
<td>bd2b03c.ada</td>
</tr>
<tr>
<td>bc3102b.ada</td>
<td>bc3502e.ada</td>
<td>bc51c01.a</td>
<td>bd2c01d.tst</td>
</tr>
<tr>
<td>bc3103b.ada</td>
<td>bc3502f.ada</td>
<td>bc51c02.a</td>
<td>bd2c02a.tst</td>
</tr>
<tr>
<td>bc3123c.ada</td>
<td>bc3502g.ada</td>
<td>bc53001.a</td>
<td>bd2c03a.tst</td>
</tr>
<tr>
<td>bc3201a.ada</td>
<td>bc3502h.ada</td>
<td>bc53002.a</td>
<td>bd2d01c.ada</td>
</tr>
<tr>
<td>bc3201b.ada</td>
<td>bc3502i.ada</td>
<td>bc54001.a</td>
<td>bd2d01d.ada</td>
</tr>
<tr>
<td>bc3202a.ada</td>
<td>bc3502j.ada</td>
<td>bc54002.a</td>
<td>bd2d02a.ada</td>
</tr>
<tr>
<td>bc3202b.ada</td>
<td>bc3502k.ada</td>
<td>bc54003.a</td>
<td>bd2d03a.ada</td>
</tr>
<tr>
<td>bc3202c.ada</td>
<td>bc3502l.ada</td>
<td>bc54a01.a</td>
<td>bd2d03b.ada</td>
</tr>
<tr>
<td>bc3202d.ada</td>
<td>bc3502m.ada</td>
<td>bc54a02.a</td>
<td>bd3001a.ada</td>
</tr>
</tbody>
</table>
c341a03.a  c35507j.ada  c36205b.ada  c37213h.ada
1 c341a04.a  c35507k.ada  c36205c.ada  c37213j.ada
1 c35003a.ada  c35507l.ada  c36205d.ada  c37213k.ada
1 c35003b.ada  c35507m.ada  c36205e.ada  c37213l.ada
1 c35003d.ada  c35507n.ada  c36205f.ada  c37215b.ada
1 c35102a.ada  c35507o.ada  c36205g.ada  c37215d.ada
1 c352001.a  c35507p.ada  c36205h.ada  c37215f.ada
1 c354002.a  c35508a.ada  c36205i.ada  c37215h.ada
1 c354003.a  c35508b.ada  c36205j.ada  c37217a.ada
1 c35502a.ada  c35508c.ada  c36205k.ada  c37217b.ada
1 c35502b.ada  c35508e.ada  c36205l.ada  c37217c.ada
1 c35502c.ada  c35508g.ada  c36301a.ada  c37304a.ada
1 c35502d.tst  c35508h.ada  c36301b.ada  c37305a.ada
1 c35502e.ada  c35508k.ada  c36302a.ada  c37306a.ada
1 c35502f.tst  c35508l.ada  c36304a.ada  c37309a.ada
1 c35502g.ada  c35508o.ada  c36305a.ada  c37310a.ada
1 c35502h.ada  c35508p.ada  c37002a.ada  c37312a.ada
1 c35502i.ada  c35703a.ada  c37003a.ada  c37402a.ada
1 c35502j.ada  c35704a.ada  c37003b.ada  c37403a.ada
1 c35502k.ada  c35704b.ada  c37005a.ada  c37404a.ada
1 c35502l.ada  c35704c.ada  c37006a.ada  c37404b.ada
1 c35502m.ada  c35704d.ada  c37008a.ada  c37405a.ada
1 c35502n.ada  c35801d.ada  c37008b.ada  c37411a.ada
1 c35502o.ada  c35902d.ada  c37009a.ada  c380001.a
1 c35502p.ada  c35904a.ada  c37010a.ada  c380002.a
1 c35503a.ada  c35904b.ada  c37010b.ada  c380003.a
1 c35503b.ada  c35a02a.ada  c371001.a  c380004.a
1 c35503c.ada  c35a05a.ada  c371002.a  c38002a.ada
1 c35503d.tst  c35a05d.ada  c371003.a  c38002b.ada
1 c35503e.ada  c35a05n.ada  c37102b.ada  c38005a.ada
1 c35503f.tst  c35a05q.ada  c37103a.ada  c38005b.ada
1 c35503g.ada  c35a07a.ada  c37105a.ada  c38005c.ada
1 c35503h.ada  c35a07d.ada  c37107a.ada  c38102a.ada
1 c35503k.ada  c35a08b.ada  c37108b.ada  c38102b.ada
1 c35503l.ada  c360002.a  c37206a.ada  c38102c.ada
1 c35503o.ada  c36104a.ada  c37207a.ada  c38102d.ada
1 c35503p.ada  c36104b.ada  c37208a.ada  c38102e.ada
1 c35504a.ada  c36172a.ada  c37208b.ada  c38104a.ada
1 c35504b.ada  c36172b.ada  c37209a.ada  c38107a.ada
1 c35505c.ada  c36172c.ada  c37209b.ada  c38107b.ada
1 c35505e.ada  c36174a.ada  c37210a.ada  c38108a.ada
1 c35505f.ada  c36180a.ada  c37211a.ada  c38108b.ada
1 c35507a.ada  c36202c.ada  c37211b.ada  c38108c0.ada
1 c35507b.ada  c36203a.ada  c37211c.ada  c38108c1.ada
1 c35507c.ada  c36204a.ada  c37211d.ada  c38108c2.ada
1 c35507e.ada  c36204b.ada  c37211e.ada  c38108d0.ada
1 c35507g.ada  c36204c.ada  c37213b.ada  c38108d1.ada
1 c35507h.ada  c36204d.ada  c37213d.ada  c3900010.a
1 c35507i.ada  c36205a.ada  c37213f.ada  c3900011.am
c390002.a  c392a01.a  c3a0024.a  c41303u.ada
  c390003.a  c392c05.a  c3a0025.a  c41303v.ada
  c390004.a  c392c07.a  c3a0026.a  c41303w.ada
  c3900050.a c392d01.a  c3a0027.a  c41304a.ada
  c3900051.a  c392d02.a  c3a0028.a  c41304b.ada
  c3900052.a  c392d03.a  c3a0029.a  c41306b.ada
  c3900053.am  c393001.a  c3a1001.a  c41306c.ada
  c3900060.a  c393007.a  c3a1002.a  c41307d.ada
  c3900061.a  c393008.a  c3a2001.a  c41309a.ada
  c3900062.a  c393009.a  c3a2002.a  c41320a.ada
  c3900063.am  c393010.a  c3a2003.a  c41321a.ada
  c390007.a  c393011.a  c3a2a01.a  c41322a.ada
  c390010.a  c393012.a  c3a2a02.a  c41323a.ada
  c390011.a  c393013.a  c410001.a  c41324a.ada
  c39006a.ada  c393a02.a  c41101d.ada  c41325a.ada
  c39006b.ada  c393a03.a  c41103a.ada  c41326a.ada
  c39006c0.ada  c393a05.a  c41103b.ada  c41327a.ada
  c39006c1.ada  c393a06.a  c41104a.ada  c41328a.ada
  c39006d.ada  c393b12.a  c41105a.ada  c41401a.ada
  c39006e.ada  c393b13.a  c41107a.ada  c41402a.ada
  c39006f0.ada  c393b14.a  c41201d.ada  c41404a.ada
  c39006f1.ada  c394001.a  c41203a.ada  c415001.a
  c39006f2.ada  c394002.a  c41203b.ada  c420001.a
  c39006f3.ada  c394003.a  c41204a.ada  c42006a.ada
  c39006g.ada  c3a0001.a  c41205a.ada  c42007e.ada
  c39007a.ada  c3a0002.a  c41206a.ada  c43003a.ada
  c39007b.ada  c3a0003.a  c41207a.ada  c43004a.ada
  c39008a.ada  c3a0004.a  c413001.a  c43004c.ada
  c39008b.ada  c3a0005.a  c413002.a  c431001.a
  c39008c.ada  c3a0006.a  c413003.a  c43103a.ada
  c390a010.a  c3a0007.a  c413004.a  c43103b.ada
  c390a011.am  c3a0008.a  c413005.a  c43104a.ada
  c390a020.a  c3a0009.a  c413006.a  c43105a.ada
  c390a021.a  c3a0010.a  c41301a.ada  c43105b.ada
  c390a022.am  c3a0011.a  c41303a.ada  c43106a.ada
  c390a030.a  c3a00120.a  c41303b.ada  c43107a.ada
  c390a031.am  c3a00121.a  c41303c.ada  c43108a.ada
  c391001.a  c3a00122.am  c41303e.ada  c431a01.a
  c391002.a  c3a0013.a  c41303f.ada  c431a02.a
  c392002.a  c3a0014.a  c41303g.ada  c431a03.a
  c392003.a  c3a0015.a  c41303i.ada  c432001.a
  c392004.a  c3a0016.a  c41303j.ada  c432002.a
  c392005.a  c3a0017.a  c41303k.ada  c432003.a
  c392008.a  c3a0018.a  c41303m.ada  c432004.a
  c392010.a  c3a0019.a  c41303n.ada  c432005.a
  c392011.a  c3a0020.a  c41303o.ada  c432004a.ada
  c392013.a  c3a0021.a  c41303q.ada  c432004c.ada
  c392014.a  c3a0022.a  c41303r.ada  c432004e.ada
  c392015.a  c3a0023.a  c41303s.ada  c43204f.ada
c43204g.ada  c45202b.ada  c45347b.ada  c45532k.ada
c43204h.ada  c45210a.ada  c45347c.ada  c45532l.ada
c43204i.ada  c45211a.ada  c45347d.ada  c45532m.dep

c43205a.ada  c45220a.ada  c45411a.ada  c45532n.dep

c43205b.ada  c45220b.ada  c45411b.dep  c45532o.dep

c43205c.ada  c45220c.ada  c45411c.ada  c45532p.dep

c43205d.ada  c45220d.ada  c45411d.ada  c45534b.ada

c43205e.ada  c45220e.ada  c45413a.ada  c45536a.dep

c43205g.ada  c45220f.ada  c45431a.ada  c456001.a

c43205h.ada  c45231a.ada  c455001.a  c45611a.ada

c43205i.ada  c45231b.dep  c45502b.dep  c45611b.dep

c43205j.ada  c45231c.dep  c45502c.dep  c45611c.dep

c43205k.ada  c45231d.tst  c45503a.ada  c45613a.dep

c43206a.ada  c45232b.ada  c45503b.dep  c45613b.dep

c43207b.ada  c45242a.ada  c45503c.dep  c45613c.dep

c43207d.ada  c45251a.ada  c45504a.ada  c45614a.ada

c43208a.ada  c45252a.ada  c45504b.dep  c45614b.dep

c43208b.ada  c45252b.ada  c45504c.dep  c45614c.dep

c43209a.ada  c45253a.ada  c45504d.ada  c45631a.ada

c43210a.ada  c45262a.ada  c45504e.dep  c45631b.ada

c43211a.ada  c45262b.ada  c45504f.dep  c45631c.dep

c43212a.ada  c45262c.ada  c45505a.ada  c45632a.ada

c43212c.ada  c45262d.ada  c45523a.ada  c45632b.dep

c43214a.ada  c45264a.ada  c45531a.ada  c45632c.dep

c43214b.ada  c45264b.ada  c45531b.ada  c45651a.ada

c43214c.ada  c45264c.ada  c45531c.ada  c45662a.ada

c43214d.ada  c45265a.ada  c45531d.ada  c45662b.ada

c43214e.ada  c45271a.ada  c45531e.ada  c45672a.ada

c43214f.ada  c45272a.ada  c45531f.ada  c457001.a

c43215a.ada  c45273a.ada  c45531g.ada  c457002.a

c43215b.ada  c45274a.ada  c45531h.ada  c457003.a

c43222a.ada  c45274b.ada  c45531i.ada  c457004.a

c43224a.ada  c45274c.ada  c45531j.ada  c458001.a

c433001.a  c45281a.ada  c45531k.ada  c460001.a

c433a01.a  c45282a.ada  c45531l.ada  c460002.a

c433a02.a  c45282b.ada  c45531m.dep  c460004.a

c433a03.a  c45291a.ada  c45531n.dep  c460005.a

c433a04.a  c45303a.ada  c45531o.dep  c460006.a

c44003d.ada  c45304a.ada  c45531p.dep  c460007.a

c44003f.ada  c45304b.dep  c45532a.ada  c460008.a

c44003g.ada  c45304c.dep  c45532b.ada  c460009.a

c450001.a  c45322a.ada  c45532c.ada  c460010.a

c45112a.ada  c45323a.ada  c45532d.ada  c460011.a

c45112b.ada  c45331a.ada  c45532e.ada  c460012.a

c45113a.ada  c45342a.ada  c45532f.ada  c460013.a

c45114b.ada  c45343a.ada  c45532g.ada  c460014.a

c452001.a  c45344a.ada  c45532h.ada  c46013a.ada

c45201a.ada  c45345b.ada  c45532i.ada  c46014a.ada

c45201b.ada  c45347a.ada  c45532j.ada  c46021a.ada
ACATS 4.0 User's Guide

A.1 Core Test Files 29 August 2014 66

c46024a.ada  c48009g.ada  c52103c.ada  c55b04a.ada
c46031a.ada  c48009h.ada  c52103f.ada  c55b05a.ada
c46032a.ada  c48009i.ada  c52103g.ada  c55b06a.ada
c46033a.ada  c48009j.ada  c52103h.ada  c55b06b.ada
c46041a.ada  c48010a.ada  c52103k.ada  c55b07a.dep
c46042a.ada  c48011a.ada  c52103l.ada  c55b07b.dep
c46043b.ada  c48012a.ada  c52103m.ada  c55b10a.ada
c46044b.ada  c490001.a  c52103p.ada  c55b11a.ada
c46051a.ada  c490002.a  c52103q.ada  c55b11b.ada
c46051b.ada  c490003.a  c52103r.ada  c55b15a.ada
c46051c.ada  c49020a.ada  c52103x.ada  c55b16a.ada
c46052a.ada  c49021a.ada  c52104a.ada  c55c02a.ada
c46053a.ada  c49022a.ada  c52104b.ada  c55c02b.ada
c46054a.ada  c49022b.ada  c52104c.ada  c56002a.ada
c460a01.a  c49022c.ada  c52104f.ada  c57003a.ada
c460a02.a  c49023a.ada  c52104g.ada  c57004a.ada
c47002a.ada  c49024a.ada  c52104h.ada  c57004b.ada
c47002b.ada  c49025a.ada  c52104k.ada  c58004c.ada
c47002c.ada  c49026a.ada  c52104l.ada  c58004d.ada
c47002d.ada  c4a005b.ada  c52104m.ada  c58004g.ada
c47003a.ada  c4a006a.ada  c52104p.ada  c58005a.ada
c47004a.ada  c4a007a.tst  c52104q.ada  c58005b.ada
c47005a.ada  c4a010a.ada  c52104r.ada  c58005h.ada
c47006a.ada  c4a010b.ada  c52104x.ada  c58006a.ada
c47007a.ada  c4a011a.ada  c52104y.ada  c58006b.ada
c47008a.ada  c4a012b.ada  c53007a.ada  c59002a.ada
c47009a.ada  c4a013a.ada  c540001.a  c59002b.ada
c47009b.ada  c4a014a.ada  c540002.a  c59002c.ada
c48004a.ada  c51004a.ada  c54a03a.ada  c61008a.ada
c48004b.ada  c52005a.ada  c54a04a.ada  c61009a.ada
c48004c.ada  c52005b.ada  c54a07a.ada  c61010a.ada
c48004d.ada  c52005c.ada  c54a13a.ada  c611001.a
c48004e.ada  c52005d.ada  c54a13b.ada  c620001.a
c48004f.ada  c52005e.ada  c54a13c.ada  c62002a.ada
c48005a.ada  c52005f.ada  c54a13d.ada  c62003b.ada
c48005b.ada  c52008a.ada  c54a22a.ada  c62004a.ada
c48006a.ada  c52008b.ada  c54a23a.ada  c62006a.ada
c48006b.ada  c52009a.ada  c54a24a.ada  c631001.a
c48007a.ada  c52009b.ada  c54a24b.ada  c640001.a
c48007b.ada  c52010a.ada  c54a42a.ada  c64002b.ada
c48007c.ada  c52011a.ada  c54a42b.ada  c64004g.ada
c48008a.ada  c52011b.ada  c54a42c.ada  c64005a.ada
c48008c.ada  c52011a.ada  c54a42d.ada  c64005b.ada
c48009a.ada  c52012a.ada  c54a42e.ada  c64005c.ada
c48009b.ada  c52012b.ada  c54a42f.ada  c64005d.ada
c48009c.ada  c52012c.ada  c54a42g.ada  c64005da.ada
c48009d.ada  c52012d.ada  c552a01.a  c64005db.ada
c48009e.ada  c52013a.ada  c552a02.a  c64005dc.ada
c48009f.ada  c52013b.ada  c55b03a.ada  c641001.a
<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>c95085c.ada</td>
<td>c950420.a</td>
<td>c97205a.ada</td>
<td>ca1004a.ada</td>
</tr>
<tr>
<td>c95085d.ada</td>
<td>c954021.a</td>
<td>c97205b.ada</td>
<td>ca1005a.ada</td>
</tr>
<tr>
<td>c95085e.ada</td>
<td>c954022.a</td>
<td>c97301a.ada</td>
<td>ca1006a.ada</td>
</tr>
<tr>
<td>c95085f.ada</td>
<td>c954023.a</td>
<td>c97301b.ada</td>
<td>ca1011a0.ada</td>
</tr>
<tr>
<td>c95085g.ada</td>
<td>c954024.a</td>
<td>c97301c.ada</td>
<td>ca1011a1.ada</td>
</tr>
<tr>
<td>c95085h.ada</td>
<td>c954025.a</td>
<td>c97301d.ada</td>
<td>ca1011a2.ada</td>
</tr>
<tr>
<td>c95085i.ada</td>
<td>c954026.a</td>
<td>c97301e.ada</td>
<td>ca1011a3.ada</td>
</tr>
<tr>
<td>c95085j.ada</td>
<td>c954027.a</td>
<td>c97302a.ada</td>
<td>ca1011a4.ada</td>
</tr>
<tr>
<td>c95085k.ada</td>
<td>c954a01.a</td>
<td>c97303a.ada</td>
<td>ca1011a5.ada</td>
</tr>
<tr>
<td>c95085l.ada</td>
<td>c954a02.a</td>
<td>c97303b.ada</td>
<td>ca1011a6.ada</td>
</tr>
<tr>
<td>c95085m.ada</td>
<td>c954a03.a</td>
<td>c97303c.ada</td>
<td>ca1012a0.ada</td>
</tr>
<tr>
<td>c95085n.ada</td>
<td>c960001.a</td>
<td>c97304a.ada</td>
<td>ca1012a1.ada</td>
</tr>
<tr>
<td>c95085o.ada</td>
<td>c960002.a</td>
<td>c97304b.ada</td>
<td>ca1012a2.ada</td>
</tr>
<tr>
<td>c95086a.ada</td>
<td>c960004.a</td>
<td>c97305a.ada</td>
<td>ca1012a3.ada</td>
</tr>
<tr>
<td>c95086b.ada</td>
<td>c96001a.ada</td>
<td>c97305b.ada</td>
<td>ca1012a4.ada</td>
</tr>
<tr>
<td>c95086c.ada</td>
<td>c96004a.ada</td>
<td>c97305c.ada</td>
<td>ca1012b0.ada</td>
</tr>
<tr>
<td>c95086d.ada</td>
<td>c96005a.ada</td>
<td>c97305d.ada</td>
<td>ca1012b2.ada</td>
</tr>
<tr>
<td>c95086e.ada</td>
<td>c96005b.tst</td>
<td>c97307a.ada</td>
<td>ca1012b4.ada</td>
</tr>
<tr>
<td>c95086f.ada</td>
<td>c96005d.ada</td>
<td>c974001.a</td>
<td>ca1013a0.ada</td>
</tr>
<tr>
<td>c95087a.ada</td>
<td>c96005f.ada</td>
<td>c974002.a</td>
<td>ca1013a1.ada</td>
</tr>
<tr>
<td>c95087b.ada</td>
<td>c96006a.ada</td>
<td>c974003.a</td>
<td>ca1013a2.ada</td>
</tr>
<tr>
<td>c95087c.ada</td>
<td>c96007a.ada</td>
<td>c974004.a</td>
<td>ca1013a3.ada</td>
</tr>
<tr>
<td>c95087d.ada</td>
<td>c96008a.ada</td>
<td>c974005.a</td>
<td>ca1013a4.ada</td>
</tr>
<tr>
<td>c95088a.ada</td>
<td>c96008b.ada</td>
<td>c974006.a</td>
<td>ca1013a5.ada</td>
</tr>
<tr>
<td>c95089a.ada</td>
<td>c97112a.ada</td>
<td>c974007.a</td>
<td>ca1013a6.ada</td>
</tr>
<tr>
<td>c95090a.ada</td>
<td>c97113a.ada</td>
<td>c974008.a</td>
<td>ca1014a0.ada</td>
</tr>
<tr>
<td>c95092a.ada</td>
<td>c97114a.ada</td>
<td>c974009.a</td>
<td>ca1014a1.ada</td>
</tr>
<tr>
<td>c95093a.ada</td>
<td>c97115a.ada</td>
<td>c974010.a</td>
<td>ca1014a2.ada</td>
</tr>
<tr>
<td>c95095a.ada</td>
<td>c97116a.ada</td>
<td>c974011.a</td>
<td>ca1014a3.ada</td>
</tr>
<tr>
<td>c95095b.ada</td>
<td>c97117a.ada</td>
<td>c974012.a</td>
<td>ca1020e0.ada</td>
</tr>
<tr>
<td>c95095c.ada</td>
<td>c97117b.ada</td>
<td>c974013.a</td>
<td>ca1020e1.ada</td>
</tr>
<tr>
<td>c95095d.ada</td>
<td>c97117c.ada</td>
<td>c974014.a</td>
<td>ca1020e2.ada</td>
</tr>
<tr>
<td>c95095e.ada</td>
<td>c97118a.ada</td>
<td>c980001.a</td>
<td>ca1020e3.ada</td>
</tr>
<tr>
<td>c951001.a</td>
<td>c97120a.ada</td>
<td>c980002.a</td>
<td>ca1022a0.ada</td>
</tr>
<tr>
<td>c951002.a</td>
<td>c97120b.ada</td>
<td>c980003.a</td>
<td>ca1022a1.ada</td>
</tr>
<tr>
<td>c953001.a</td>
<td>c97201a.ada</td>
<td>c990001.a</td>
<td>ca1022a2.ada</td>
</tr>
<tr>
<td>c953002.a</td>
<td>c97201b.ada</td>
<td>c99005a.ada</td>
<td>ca1022a3.ada</td>
</tr>
<tr>
<td>c953003.a</td>
<td>c97201c.ada</td>
<td>c9a003a.ada</td>
<td>ca1022a4.ada</td>
</tr>
<tr>
<td>c954001.a</td>
<td>c97201d.ada</td>
<td>c9a004a.ada</td>
<td>ca1022a5.ada</td>
</tr>
<tr>
<td>c954010.a</td>
<td>c97201e.ada</td>
<td>c9a007a.ada</td>
<td>ca1022a6.ada</td>
</tr>
<tr>
<td>c954011.a</td>
<td>c97201g.ada</td>
<td>c9a009a.ada</td>
<td>ca11001.a</td>
</tr>
<tr>
<td>c954012.a</td>
<td>c97201h.ada</td>
<td>c9a009c.ada</td>
<td>ca11002.a</td>
</tr>
<tr>
<td>c954013.a</td>
<td>c97201x.ada</td>
<td>c9a009f.ada</td>
<td>ca11003.a</td>
</tr>
<tr>
<td>c954014.a</td>
<td>c97202a.ada</td>
<td>c9a009g.ada</td>
<td>ca110040.a</td>
</tr>
<tr>
<td>c954015.a</td>
<td>c97203a.ada</td>
<td>c9a009h.ada</td>
<td>ca110041.a</td>
</tr>
<tr>
<td>c954016.a</td>
<td>c97203b.ada</td>
<td>c9a010a.ada</td>
<td>ca110042.am</td>
</tr>
<tr>
<td>c954017.a</td>
<td>c97203c.ada</td>
<td>c9a011a.ada</td>
<td>ca110050.a</td>
</tr>
<tr>
<td>c954018.a</td>
<td>c97204a.ada</td>
<td>c9a011b.ada</td>
<td>ca110051.am</td>
</tr>
<tr>
<td>c954019.a</td>
<td>c97204b.ada</td>
<td>ca1003a.ada</td>
<td>ca11006.a</td>
</tr>
</tbody>
</table>
ca11007.a  ca140232.am  ca5003a2.ada  cb40a020.a
ca11008.a  ca140233.a  ca5003a3.ada  cb40a021.am
ca11009.a  ca140280.a  ca5003a4.ada  cb40a030.a
ca11010.a  ca140281.a  ca5003a5.ada  cb40a031.am
ca11011.a  ca140282.a  ca5003a6.ada  cb40a04.a
ca11012.a  ca140283.am  ca5003b0.ada  cb41001.a
ca11013.a  ca15003.a  ca5003b1.ada  cb41002.a
ca11014.a  ca200020.a  ca5003b2.ada  cb41003.a
ca11015.a  ca200021.a  ca5003b3.ada  cb41004.a
ca11016.a  ca200022.am  ca5003b4.ada  cb5001a.ada
ca11017.a  ca200030.a  ca5003b5.ada  cb5001b.ada
ca11018.a  ca200031.am  ca5004a.ada  cb5002a.ada
ca11019.a  ca2001h0.ada  ca5004b0.ada  cc1004a.ada
ca11020.a  ca2001h1.ada  ca5004b1.ada  cc1005b.ada
ca11021.a  ca2001h2.ada  ca5004b2.ada  cc1010a.ada
ca11022.a  ca2001h3.ada  ca5006a.ada  cc1010b.ada
ca110230.a  ca2002a0.ada  cb10002.a  cc1018a.ada
ca110231.a  ca2002a1.ada  cb1001a.ada  cc1104c.ada
ca110232.am  ca2002a2.ada  cb1004a.ada  cc1107b.ada
ca1102a0.ada  ca2003a0.ada  cb1005a.ada  cc1111a.ada
ca1102a1.ada  ca2003a1.ada  cb1010a.ada  cc1204a.ada
ca1102a2.ada  ca2004a0.ada  cb1010c.ada  cc1207b.ada
ca1106a.ada  ca2004a1.ada  cb1010d.ada  cc1220a.ada
ca1108a.ada  ca2004a2.ada  cb20001.a  cc1221a.ada
ca1108b.ada  ca2004a3.ada  cb20003.a  cc1221b.ada
ca11a01.a  ca2004a4.ada  cb20004.a  cc1221c.ada
ca11a02.a  ca2007a0.ada  cb20005.a  cc1221d.ada
ca11b01.a  ca2007a1.ada  cb20006.a  cc1222a.ada
ca11b02.a  ca2007a2.ada  cb20007.a  cc1223a.ada
ca11c01.a  ca2007a3.ada  cb2004a.ada  cc1224a.ada
ca11c02.a  ca2008a0.ada  cb2005a.ada  cc1225a.tst
ca11c03.a  ca2008a1.ada  cb2006a.ada  cc1226b.ada
ca11d010.a  ca2008a2.ada  cb2007a.ada  cc1227a.ada
ca11d011.a  ca2009a.ada  cb20a02.a  cc1301a.ada
ca11d012.a  ca2009c0.ada  cb3003a.ada  cc1302a.ada
ca11d013.am  ca2009c1.ada  cb3003b.ada  cc1304a.ada
ca11d02.a  ca2009d.ada  cb3004a.ada  cc1304b.ada
ca11d03.a  ca2009f0.ada  cb40005.a  cc1307a.ada
ca120010.a  ca2009f1.ada  cb4001a.ada  cc1307b.ada
ca120011.a  ca2009f2.ada  cb4002a.ada  cc1308a.ada
ca120012.am  ca2011b.ada  cb4003a.ada  cc1310a.ada
ca12002.a  ca21001.a  cb4004a.ada  cc1311a.ada
ca13001.a  ca3011a0.ada  cb4005a.ada  cc1311b.ada
ca13002.a  ca3011a1.ada  cb4006a.ada  cc2002a.ada
ca13003.a  ca3011a2.ada  cb4007a.ada  cc30001.a
ca13a01.a  ca3011a3.ada  cb4008a.ada  cc30002.a
ca13a02.a  ca3011a4.ada  cb4009a.ada  cc30003.a
ca140230.a  ca5003a0.ada  cb4013a.ada  cc30004.a
ca140231.a  ca5003a1.ada  cb40a01.a  cc3004a.ada
| cc3007a.ada | cc3305d.ada | cd1009k.tst | cd2a32j.ada |
| cc3007b.ada | cc3601a.ada | cd1009l.ada | cd2a32j.ada |
| cc3011a.ada | cc3601c.ada | cd1009m.ada | cd2a51a.ada |
| cc3011d.ada | cc3602a.ada | cd1009n.ada | cd2a53a.ada |
| cc3012a.ada | cc3603a.ada | cd1009o.ada | cd2a53e.ada |
| cc3015a.ada | cc3605a.ada | cd1009p.ada | cd2a83c.tst |
| cc3016b.ada | cc3606a.ada | cd1009q.ada | cd2a91c.tst |
| cc3016c.ada | cc3606b.ada | cd1009r.ada | cd2b11b.ada |
| cc3016f.ada | cc3607b.ada | cd1009s.ada | cd2b11d.ada |
| cc3016i.ada | cc40001.a | cd1009t.tst | cd2b11e.ada |
| cc3017b.ada | cc50001.a | cd1009u.tst | cd2b11f.ada |
| cc3019a.ada | cc50a01.a | cd1009v.ada | cd2b16a.ada |
| cc3019b0.ada | cc50a02.a | cd1009w.ada | cd2c11a.tst |
| cc3019b1.ada | cc51001.a | cd1009x.ada | cd2c11d.tst |
| cc3019b2.ada | cc51002.a | cd1009y.ada | cd2d11a.ada |
| cc3019c0.ada | cc51003.a | cd1009z.ada | cd2d13a.ada |
| cc3019c1.ada | cc51004.a | cd1c03a.ada | cd30001.a |
| cc3019c2.ada | cc51006.a | cd1c03b.ada | cd30002.a |
| cc3106b.ada | cc40001.a | cd1c03c.ada | cd30003.a |
| cc3120a.ada | cc51008.a | cd1c03e.tst | cd30004.a |
| cc3120b.ada | cc51009.a | cd1c03f.ada | cd300050.am |
| cc3121a.ada | cc51a01.a | cd1c03g.ada | cd300051.c |
| cc3123a.ada | cc51b03.a | cd1c03h.ada | cd30006.a |
| cc3125a.ada | cc51d01.a | cd1c03i.ada | cd30007.a |
| cc3125b.ada | cc51d02.a | cd1c04a.ada | cd3014a.ada |
| cc3125c.ada | cc51d04.a | cd1c04d.ada | cd3014c.ada |
| cc3125d.ada | cc54002.a | cd1c04e.ada | cd3014d.ada |
| cc3126a.ada | cc54003.a | cd1c06a.tst | cd3014f.ada |
| cc3127a.ada | cc54004.a | cd20001.a | cd3015a.ada |
| cc3128a.ada | cc70001.a | cd2a121a.ada | cd3015c.ada |
| cc3203a.ada | cc70002.a | cd2a12c.ada | cd3015e.ada |
| cc3207b.ada | cc70003.a | cd2a211a.ada | cd3015f.ada |
| cc3220a.ada | cc70a01.a | cd2a221a.ada | cd3015g.ada |
| cc3221a.ada | cc70a02.a | cd2a222a.ada | cd3015h.ada |
| cc3222a.ada | cc70b01.a | cd2a223a.ada | cd3015i.ada |
| cc3223a.ada | cc70b02.a | cd2a22j.ada | cd3015k.ada |
| cc3224a.ada | cc70c01.a | cd2a23a.ada | cd3021a.ada |
| cc3225a.ada | cc70c02.a | cd2a23e.ada | cd333001.a |
| cc3230a.ada | cd10001.a | cd2a24a.ada | cd333002.a |
| cc3231a.ada | cd10002.a | cd2a24e.ada | cd40001.a |
| cc3232a.ada | cd1009a.ada | cd2a24i.ada | cd4031a.ada |
| cc3233a.ada | cd1009b.ada | cd2a24j.ada | cd4041a.tst |
| cc3234a.ada | cd1009d.ada | cd2a31a.ada | cd4051a.ada |
| cc3235a.ada | cd1009e.ada | cd2a31c.ada | cd4051b.ada |
| cc3236a.ada | cd1009f.ada | cd2a31e.ada | cd4051c.ada |
| cc3240a.ada | cd1009g.ada | cd2a32a.ada | cd4051d.ada |
| cc3305a.ada | cd1009h.ada | cd2a32c.ada | cd5003a.ada |
| cc3305b.ada | cd1009i.ada | cd2a32e.ada | cd5003b.ada |
| cc3305c.ada | cd1009j.ada | cd2a32g.ada | cd5003c.ada |
| cd5003d.ada | cd7103d.ada | cc2102x.ada | ce2206a.ada |
| cd5003e.ada | cd7202a.ada | cc2102y.ada | ce2208b.ada |
| cd5003f.ada | cd7204b.ada | cc2103a.tst | ce2401a.ada |
| cd5003g.ada | cd7204c.ada | cc2103b.tst | ce2401b.ada |
| cd5003h.ada | cd72a01.a  | cc2103c.ada | ce2401c.ada |
| cd5003i.ada | cd72a02.a  | cc2103d.ada | ce2401e.ada |
| cd5011a.ada | cd7305a.ada | cc2104a.ada | ce2401f.ada |
| cd5011c.ada | cd90001.a  | cc2104b.ada | ce2401h.ada |
| cd5011e.ada | cd92001.a  | cc2104c.ada | ce2401l.ada |
| cd5011g.ada | cda201a.ada | cc2104d.ada | ce2401j.ada |
| cd5011i.ada | cda201b.ada | cc2106a.ada | ce2401k.ada |
| cd5011k.ada | cda201c.ada | cc2106b.ada | ce2401l.ada |
| cd5011m.ada | cda201e.ada | cc2108c.ada | ce2402a.ada |
| cd5011q.ada | cdb0001.a  | cc2108f.ada | ce2403a.tst |
| cd5011y.ada | cdb0002.a  | cc2108g.ada | ce2404a.ada |
| cd5012a.ada | cdb0a01.a  | cc2108h.ada | ce2404b.ada |
| cd5012b.ada | cdb0a02.a  | cc2109a.ada | ce2405b.ada |
| cd5012c.ada | cdb4001.a  | cc2109b.ada | ce2406a.ada |
| cd5012d.ada | cdd1001.a  | cc2109c.ada | ce2407a.ada |
| cd5012i.ada | cdd2001.a  | cc2110a.ada | ce2407b.ada |
| cd5012m.ada | cdd2a01.a  | cc2110c.ada | ce2408a.ada |
| cd5013a.ada | cdd2a02.a  | cc2111a.ada | ce2408b.ada |
| cd5013c.ada | cdd2a03.a  | cc2111b.ada | ce2409a.ada |
| cd5013e.ada | cde0001.a  | cc2111c.ada | ce2409b.ada |
| cd5013g.ada | cde0002.a  | cc2111e.ada | ce2410a.ada |
| cd5013i.ada | cde0003.a  | cc2111f.ada | ce2410b.ada |
| cd5013k.ada | ce2102a.ada | cc2111g.ada | ce2411a.ada |
| cd5013m.ada | ce2102b.ada | cc2111i.ada | ce3002b.tst |
| cd5013o.ada | ce2102c.tst | cc2201a.ada | ce3002c.tst |
| cd5014a.ada | ce2102d.ada | cc2201b.ada | ce3002d.ada |
| cd5014c.ada | ce2102e.ada | cc2201c.ada | ce3002f.ada |
| cd5014d.ada | ce2102f.ada | cc2201d.dep | ce3102a.ada |
| cd5014g.ada | ce2102g.ada | cc2201e.dep | ce3102b.tst |
| cd5014i.ada | ce2102h.tst | cc2201f.ada | ce3102d.ada |
| cd5014k.ada | ce2102i.ada | cc2201g.ada | ce3102e.ada |
| cd5014m.ada | ce2102j.ada | cc2201h.ada | ce3102f.ada |
| cd5014o.ada | ce2102k.ada | cc2201i.ada | ce3102g.ada |
| cd5014t.ada | ce2102l.ada | cc2201j.ada | ce3102h.ada |
| cd5014v.ada | ce2102m.ada | cc2201k.ada | ce3102i.ada |
| cd5014x.ada | ce2102n.ada | cc2201l.ada | ce3102j.ada |
| cd5014y.ada | ce2102o.ada | cc2201m.ada | ce3102k.ada |
| cd5014z.ada | ce2102p.ada | cc2201n.ada | ce3103a.ada |
| cd70001.a  | ce2102q.ada | cc2202a.ada | ce3104a.ada |
| cd70002.a  | ce2102r.ada | cc2203a.tst | ce3104b.ada |
| cd7007b.ada | ce2102s.ada | cc2204a.ada | ce3104c.ada |
| cd7101d.ada | ce2102t.ada | cc2204b.ada | ce3106a.ada |
| cd7101e.dep| ce2102u.ada | cc2204c.ada | ce3106b.ada |
| cd7101f.dep| ce2102v.ada | cc2204d.ada | ce3107a.tst |
| cd7101g.tst| ce2102w.ada | cc2205a.ada | ce3107b.ada |
cxa5a05.a  cxa5a06.a  cxa5a07.a  cxa5a08.a  cxa5a09.a  cxa5a10.a  cxa8001.a  cxa8002.a  cxa8003.a  cxa9001.a  cxa9002.a  cxa9003.a  cxa9004.a  cxa9005.a  cxa9006.a  cxa9007.a  cxa9008.a  cxa9009.a  cxa9010.a  cxa9011.a  cxa9012.a  cxa9013.a  cxa9014.a  cxa9015.a  cxa9016.a  cxa9017.a  cxa9018.a  cxa9019.a  cxa9020.a  cxa9021.a  cxa9022.a  cxa9023.a  cxa9024.a  cxa9025.a  cxa9026.a  cxa9027.a  cxa9028.a  cxa9029.a  cxa9030.a  cxa9031.a  cxa9032.a  cxa9033.a  cxa9034.a  cxa9035.a  cxa9036.a  cxa9037.a  cxa9038.a  cxa9039.a  cxa9040.a  cxa9041.a  cxa9042.a  cxa9043.a  cxa9044.a  cxa9045.a  cxa9046.a  cxa9047.a  cxa9048.a  cxa9049.a  cxa9050.a  cxa9051.a  cxa9052.a  cxa9053.a  cxa9054.a  cxa9055.a  cxa9056.a  cxa9057.a  cxa9058.a  cxa9059.a  cxa9060.a  cxa9061.a  cxa9062.a  cxa9063.a  cxa9064.a  cxa9065.a  cxa9066.a  cxa9067.a  cxa9068.a  cxa9069.a  cxa9070.a  cxa9071.a  cxa9072.a  cxa9073.a  cxa9074.a  cxa9075.a  cxa9076.a  cxa9077.a  cxa9078.a  cxa9079.a  cxa9080.a  cxa9081.a  cxa9082.a  cxa9083.a  cxa9084.a  cxa9085.a  cxa9086.a  cxa9087.a  cxa9088.a  cxa9089.a  cxa9090.a  cxa9091.a  cxa9092.a  cxa9093.a  cxa9094.a  cxa9095.a  cxa9096.a  cxa9097.a  cxa9098.a  cxa9099.a  cxa9100.a  cxa9101.a  cxa9102.a  cxa9103.a  cxa9104.a  cxa9105.a  cxa9106.a  cxa9107.a  cxa9108.a  cxa9109.a  cxa9110.a  cxa9111.a  cxa9112.a  cxa9113.a  cxa9114.a  cxa9115.a  cxa9116.a  cxa9117.a  cxa9118.a  cxa9119.a  cxa9120.a  cxa9121.a  cxa9122.a  cxa9123.a  cxa9124.a  cxa9125.a  cxa9126.a  cxa9127.a  cxa9128.a  cxa9129.a  cxa9130.a  cxa9131.a  cxa9132.a  cxa9133.a  cxa9134.a  cxa9135.a  cxa9136.a  cxa9137.a  cxa9138.a  cxa9139.a  cxa9140.a  cxa9141.a  cxa9142.a  cxa9143.a  cxa9144.a  cxa9145.a  cxa9146.a  cxa9147.a  cxa9148.a  cxa9149.a  cxa9150.a  cxa9151.a  cxa9152.a  cxa9153.a  cxa9154.a  cxa9155.a  cxa9156.a  cxa9157.a  cxa9158.a  cxa9159.a  cxa9160.a  cxa9161.a  cxa9162.a  cxa9163.a  cxa9164.a  cxa9165.a  cxa9166.a  cxa9167.a  cxa9168.a  cxa9169.a  cxa9170.a  cxa9171.a  cxa9172.am  cxa9173.a  cxb2001.a  cxb2002.a  d4a002a.ada  d4a002b.ada  d4a004a.ada  d4a004b.ada  e28002b.ada  e28005d.ada  e52103y.ada  eb4011a.ada  eb4012a.ada  eb4014a.ada  ec3203a.ada  ec3204a.ada  ee3402b.ada  ee3409f.ada  ee3412c.ada  la140010.a  la140011.am  la140012.a  la140020.a  la140021.am  la140022.a  la140030.a  la140031.a  la140032.am  la140033.a  la140040.a  la140041.am  la140042.a  la140050.a  la140051.a  la140052.am  la140053.a  la140060.a  la140061.a  la140062.am  la140063.a  la140070.a  la140071.a  la140072.am  la140073.a  la140080.a  la140081.a  la140082.am  la140083.a  la140090.a  la140091.a  la140092.am  la140093.a  la140100.a  la140101.a
A.2 Specialized Needs Annex Test Files

This section lists the files containing Specialized Needs Annex tests; that is, tests for requirements specified in Annex C, Annex D, Annex E, Annex G, or Annex H.

<table>
<thead>
<tr>
<th>bxc3001.a</th>
<th>bxe2008.a</th>
<th>bxf1001.a</th>
<th>bxh4013.a</th>
</tr>
</thead>
<tbody>
<tr>
<td>bxc3002.a</td>
<td>bxe2009.a</td>
<td>bxh4001.a</td>
<td>cxc3001.a</td>
</tr>
<tr>
<td>bxc5001.a</td>
<td>bxe2010.a</td>
<td>bxh4002.a</td>
<td>cxc3002.a</td>
</tr>
<tr>
<td>bxe6001.a</td>
<td>bxe2011.a</td>
<td>bxh4003.a</td>
<td>cxc3003.a</td>
</tr>
<tr>
<td>bxe6002.a</td>
<td>bxe2012.a</td>
<td>bxh4004.a</td>
<td>cxc3004.a</td>
</tr>
<tr>
<td>bxe6003.a</td>
<td>bxe2013.a</td>
<td>bxh4005.a</td>
<td>cxc3005.a</td>
</tr>
<tr>
<td>bxc6a01.a</td>
<td>bxe2a01.a</td>
<td>bxh4006.a</td>
<td>cxc3006.a</td>
</tr>
<tr>
<td>bxc6a02.a</td>
<td>bxe2a02.a</td>
<td>bxh4007.a</td>
<td>cxc3007.a</td>
</tr>
<tr>
<td>bxc6a03.a</td>
<td>bxe2a03.a</td>
<td>bxh4008.a</td>
<td>cxc3008.a</td>
</tr>
<tr>
<td>bxc6a04.a</td>
<td>bxe2a04.a</td>
<td>bxh4009.a</td>
<td>cxc3009.a</td>
</tr>
<tr>
<td>bxd1001.a</td>
<td>bxe2a05.a</td>
<td>bxh4010.a</td>
<td>cxc3010.a</td>
</tr>
<tr>
<td>bxd1002.a</td>
<td>bxe2a06.a</td>
<td>bxh4011.a</td>
<td>cxc6001.a</td>
</tr>
<tr>
<td>bxe2007.a</td>
<td>bxe4001.a</td>
<td>bxh4012.a</td>
<td>cxc6002.a</td>
</tr>
</tbody>
</table>
Specialized Needs Annex Test Files

A.2

29 August 2014  76
A.3 Foundation Code Files

This section lists the foundation files. These files contain packages that may be used by more than one test for related objectives.

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>f340a000.a</td>
<td>f460a00.a</td>
<td>fa13a00.a</td>
<td>fd72a00.a</td>
</tr>
<tr>
<td>f340a001.a</td>
<td>f552a00.a</td>
<td>fa13b00.a</td>
<td>fdb9a00.a</td>
</tr>
<tr>
<td>f341a00.a</td>
<td>f650a00.a</td>
<td>fa21a00.a</td>
<td>fdd2a00.a</td>
</tr>
<tr>
<td>f390a00.a</td>
<td>f730a000.a</td>
<td>fb20a00.a</td>
<td>fxa5a00.a</td>
</tr>
<tr>
<td>f392a00.a</td>
<td>f730a001.a</td>
<td>fb40a00.a</td>
<td>fxaca00.a</td>
</tr>
<tr>
<td>f392c00.a</td>
<td>f731a00.a</td>
<td>fc50a00.a</td>
<td>fxacb00.a</td>
</tr>
<tr>
<td>f392d00.a</td>
<td>f750a00.a</td>
<td>fc51a00.a</td>
<td>fxacc00.a</td>
</tr>
<tr>
<td>f393a00.a</td>
<td>f760a00.a</td>
<td>fc51b00.a</td>
<td>fxai0a0.a</td>
</tr>
<tr>
<td>f393b00.a</td>
<td>f940a00.a</td>
<td>fc51c00.a</td>
<td>fxce0a0.a</td>
</tr>
<tr>
<td>f394a00.a</td>
<td>f954a00.a</td>
<td>fc51d00.a</td>
<td>fxce2a0.a</td>
</tr>
<tr>
<td>f3a1a00.a</td>
<td>fa11a00.a</td>
<td>fc54a00.a</td>
<td>fxfa0a0.a</td>
</tr>
<tr>
<td>f3a2a00.a</td>
<td>fa11b00.a</td>
<td>fc70a00.a</td>
<td>fxf3a00.a</td>
</tr>
<tr>
<td>f431a00.a</td>
<td>fa11c00.a</td>
<td>fc70b00.a</td>
<td></td>
</tr>
<tr>
<td>f433a00.a</td>
<td>fa11d00.a</td>
<td>fc70c00.a</td>
<td></td>
</tr>
</tbody>
</table>

A.4 Documentation Files

This section lists all the files containing ACATS 4.0 documentation. Files with the .pdf extension are in Adobe Acrobat format. Files with the .htm extension are in HTML; files with the extensions .gif and .png are associated graphics files. All other files are in ASCII text format.

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>acats-ug.pdf</td>
<td>next.gif</td>
<td>ug-415.htm</td>
</tr>
<tr>
<td>cont.gif</td>
<td>prev.gif</td>
<td>ug-416.htm</td>
</tr>
<tr>
<td>cover03a.pdf</td>
<td>testobj.txt</td>
<td>ug-42.htm</td>
</tr>
<tr>
<td>cover03b.pdf</td>
<td>ug-01.htm</td>
<td>ug-421.htm</td>
</tr>
<tr>
<td>cover04a.pdf</td>
<td>ug-02.htm</td>
<td>ug-422.htm</td>
</tr>
<tr>
<td>cover04b.pdf</td>
<td>ug-1.htm</td>
<td>ug-423.htm</td>
</tr>
<tr>
<td>cover05.pdf</td>
<td>ug-11.htm</td>
<td>ug-424.htm</td>
</tr>
<tr>
<td>cover06.pdf</td>
<td>ug-12.htm</td>
<td>ug-425.htm</td>
</tr>
<tr>
<td>cover07.pdf</td>
<td>ug-2.htm</td>
<td>ug-426.htm</td>
</tr>
<tr>
<td>cover08.pdf</td>
<td>ug-3.htm</td>
<td>ug-427.htm</td>
</tr>
<tr>
<td>cover10.pdf</td>
<td>ug-31.htm</td>
<td>ug-428.htm</td>
</tr>
<tr>
<td>cover13.pdf</td>
<td>ug-32.htm</td>
<td>ug-43.htm</td>
</tr>
<tr>
<td>coverkey.pdf</td>
<td>ug-321.htm</td>
<td>ug-431.htm</td>
</tr>
<tr>
<td>coversum.pdf</td>
<td>ug-322.htm</td>
<td>ug-432.htm</td>
</tr>
<tr>
<td>coverage.txt</td>
<td>ug-323.htm</td>
<td>ug-433.htm</td>
</tr>
<tr>
<td>dirs.png</td>
<td>ug-4.htm</td>
<td>ug-44.htm</td>
</tr>
<tr>
<td>find.gif</td>
<td>ug-41.htm</td>
<td>ug-45.htm</td>
</tr>
<tr>
<td>index.gif</td>
<td>ug-411.htm</td>
<td>ug-46.htm</td>
</tr>
<tr>
<td>leg-name.png</td>
<td>ug-412.htm</td>
<td>ug-47.htm</td>
</tr>
<tr>
<td>lib.gif</td>
<td>ug-413.htm</td>
<td>ug-48.htm</td>
</tr>
<tr>
<td>mod-name.png</td>
<td>ug-414.htm</td>
<td>ug-5.htm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ug-541.htm</td>
</tr>
</tbody>
</table>
A.5 Other Files

This section lists the files falling into the *others* category in the table found in Section 4.1. They are listed in the categories described there.

A.5.1 List of ACATS 4.0 Files

`acats40.lst`

A.5.2 Support Units Referenced by Many Tests

- `checkfil.ada`
- `enumchek.ada`
- `fndecl.ada`
- `impdef.a`
- `impdefc.a`
- `impdefd.a`
- `impdefe.a`
- `impdefg.a`
- `impdefh.a`
- `lencheck.ada`
- `repbody.ada`
- `repspec.ada`
- `spprt13s.tst`
- `tctouch.ada`

A.5.3 Preprocessing Tools and Data

- `macrosub.ada`
- `macro.dfs`
- `tsttests.dat`

A.5.4 Tests for Reporting Code
A.6 Tests With Special Requirements

The tests listed in this section have special processing requirements that are described in the internal test commentary.

| ba15001 | cxd2002 | cxg2006 | la14015 |
| bxh4001 | cxd2004 | cxg2008 | la14017 |
| bxh4002 | cxd2006 | cxg2009 | la14018 |
| bxh4007 | cxd3003 | cxg2014 | la14024 |
| bxh4008 | cxd4001 | cxg2015 | la14025 |
| bxh4009 | cxd4003 | cxg2016 | la14026 |
| bxh4010 | cxd4004 | cxg2017 | la14027 |
| bxh4011 | cxd4005 | cxg2018 | la20001 |
| bxh4012 | cxd4006 | cxg2019 | lc30001 |
| bxh4013 | cxd4007 | cxg2020 | lc30002 |
| ca11023 | cxd4008 | cxg2021 | lc30003 |
| ca12001 | cxd4009 | cxg2022 | lx7009  |
| cd30005 | cxd4010 | cxg2023 | lxe3001 |
| cxb3004 | cxa003  | cxg2024 | lxe3002 |
| cxb3006 | cxa004  | cxh1001 | lxh4001 |
| cxb3008 | cxe1001 | cxh3001 | lxh4002 |
| cxb3013 | cxe2001 | cxh3003 | lxh4003 |
| cxb3017 | cxe2002 | la14001 | lhx4004 |
| cxb3018 | cxe4001 | la14002 | lhx4005 |
| cxb4009 | cxe4002 | la14003 | lhx4006 |
| cxb5004 | cxe4003 | la14004 | lhx4007 |
| cxb5005 | cxe4004 | la14005 | lhx4008 |
| cxc3001 | cxe4005 | la14006 | lhx4009 |
| cxc3003 | cxe4006 | la14007 | lhx4010 |
| cxc3004 | cxe5002 | la14008 | lhx4011 |
| cxc3006 | cxe5003 | la14009 | lhx4012 |
| cxc3008 | cxg1002 | la14010 | lhx4013 |
| cxc3010 | cxg1005 | la14011 | lhx4014 |
| cxd1004 | cxg2002 | la14012 |
| cxd1005 | cxg2003 | la14013 |
| cxd2001 | cxg2004 | la14014 |

A.7 Test Files Added Since ACATS 3.1

The following test files are new in ACATS 4.0:

| b324001.a | b324002.a | b324003.a | b3a1006.a |
A.8 Documentation Files Added Since ACATS 3.1

The following documentation files are new in ACATS 4.0:

- cover03a.pdf
- cover03b.pdf
- cover04a.pdf
- cover04b.pdf
- cover05.pdf
- cover06.pdf
- cover07.pdf
- cover08.pdf
- cover09.pdf
- cover10.pdf
- cover13.pdf
- ug-ac.htm

A.9 Test Files Modified Since ACATS 3.1

The following test files have been modified from their ACATS 3.1 versions:

- b393005.a
- b3a1003.a
- b490001.a
- b954005.a
- b950001.a
- b954005.a
- ba110141.a
- ba110142.a
- bc54002.a
A.10 Support Files Modified Since ACATS 3.1

The following support files have been modified from their ACATS 3.1 versions:

impdef.a repbody.ada tctouch.a

A.11 Documentation Files Modified Since ACATS 3.1

The following documentation files have been modified from their ACATS 3.1 versions:

acats-ug.pdf ug-431.htm ug-553.htm ug-ab.htm
coverkey.pdf ug-432.htm ug-554.htm ug-ac.htm
coversum.pdf ug-433.htm ug-555.htm ug-b.htm
dirs.png ug-44.htm ug-5551.htm ug-b1.htm
ug-01.htm ug-45.htm ug-5552.htm ug-b2.htm
ug-02.htm ug-46.htm ug-5553.htm ug-b3.htm
ug-1.htm ug-47.htm ug-5554.htm ug-c.htm
ug-12.htm ug-5.htm ug-556.htm ug-c2.htm
ug-2.htm ug-51.htm ug-56.htm ug-c3.htm
ug-3.htm ug-511.htm ug-561.htm ug-c4.htm
ug-31.htm ug-512.htm ug-562.htm ug-c41.htm
ug-32.htm ug-5121.htm ug-563.htm ug-c42.htm
ug-321.htm ug-5122.htm ug-564.htm ug-d.htm
ug-322.htm ug-513.htm ug-565.htm ug-d1.htm
ug-323.htm ug-5131.htm ug-57.htm ug-d11.htm
ug-4.htm ug-5132.htm ug-571.htm ug-d12.htm
ug-41.htm ug-5133.htm ug-572.htm ug-d13.htm
ug-412.htm ug-52.htm ug-a.htm ug-d15.htm
ug-413.htm ug-521.htm ug-a1.htm ug-d16.htm
ug-414.htm ug-522.htm ug-a2.htm ug-d17.htm
ug-415.htm ug-523.htm ug-a3.htm ug-d18.htm
ug-416.htm ug-524.htm ug-a4.htm ug-d19.htm
ug-42.htm ug-525.htm ug-a5.htm ug-d1a.htm
ug-421.htm ug-53.htm ug-a51.htm ug-d1b.htm
ug-422.htm ug-531.htm ug-a52.htm ug-d2.htm
ug-423.htm ug-532.htm ug-a53.htm ug-d21.htm
ug-424.htm ug-54.htm ug-a54.htm ug-d22.htm
ug-425.htm ug-541.htm ug-a6.htm ug-d23.htm
ug-426.htm ug-542.htm ug-a7.htm ug-d24.htm
ug-427.htm ug-55.htm ug-a8.htm ug-d25.htm
ug-43.htm ug-552.htm ug-aa.htm ug-d27.htm
A.11 Documentation Files Modified Since ACATS 3.1

ug-d28.htm      ug-d2c.htm      ug-d2g.htm      ug-f.htm
ug-d29.htm      ug-d2d.htm      ug-d2h.htm      ug-toc.htm
ug-d2a.htm      ug-d2e.htm      ug-d2i.htm      ug-ttl.htm
ug-d2b.htm      ug-d2f.htm      ug-e.htm

A.12 Test Files Deleted Since ACATS 3.1

The following test files were present in ACATS 3.1 but do not appear in ACATS 4.0:

b38105a.ada      bc1202c.ada      c750b01.a      f750b00.a
b65001a.ada      c62003a.ada      cc3017c.ada

A.13 Documentation Files Deleted Since ACATS 3.1

The following documentation files were present in ACATS 3.1 but do not appear in ACATS 4.0:

cover-03.pdf      cover-06.pdf      cover-08.pdf
cover-04.pdf      cover-07.pdf      cover-10.pdf
Annex B: Parameterization Files

In ACATS 4.0, two methods are used to account for the use of implementation-dependent values in the tests.

For legacy tests, a "macro" substitution technique is used. Legacy tests requiring implementation-specific values contain symbols beginning with the '$' character; for example, the symbol $INTEGER_LAST is used where the code expects the implementation-specific integer literal representing the largest integer. For each implementation, these symbols must be systematically replaced with the appropriate values. A data file, MACRO.DFS, and an Ada program, Macrosub, are provided to facilitate this substitution.

For modern tests, a hierarchy of packages is provided that contain constants and functions that provide the desired implementation-specific values. These packages ("ImpDef" and its children) should be modified for each implementation to provide the needed values.

Information regarding the macro substitution technique is presented in Sections B.1 and B.2. Section B.3 describes the ImpDef package hierarchy.

B.1 Macro Substitution File

The support file MACRO.DFS provides substitutions for special symbols that appear in certain ACATS 4.0 tests (indicated by the three-letter file type (extension) "TST" and listed in Section B.2). The support program Macrosub may be used to insert these implementation-specific values in place of the special symbols in the test. The following excerpt from the file describes the file and its use.

```
-- MACRO.DFS
-- THIS FILE CONTAINS THE MACRO DEFINITIONS USED IN THE ACVC TESTS.
-- THESE DEFINITIONS ARE USED BY THE ACVC TEST PRE-PROCESSOR,
-- MACROSUB. MACROSUB WILL CALCULATE VALUES FOR THOSE MACRO SYMBOLS
-- WHOSE DEFINITIONS DEPEND ON THE VALUE OF MAX_IN_LEN (NAMELY, THE
-- VALUES OF THE MACRO SYMBOLS BIG_ID1, BIG_ID2, BIG_ID3, BIG_ID4,
-- BIG_STRING1, BIG_STRING2, MAX_STRING_LITERAL, BIG_INT_LIT, BIG_REAL_LIT,
-- AND BLANKS). THEREFORE, ANY VALUES GIVEN IN THIS FILE FOR THOSE
-- MACRO SYMBOLS WILL BE IGNORED BY MACROSUB.
-- NOTE: AS REQUIRED BY THE MACROSUB PROGRAM, THE FIRST MACRO DEFINED
-- IN THIS FILE IS $MAX_IN_LEN. THE NEXT 5 MACRO DEFINITIONS
-- ARE FOR THOSE MACRO SYMBOLS THAT DEPEND ON THE VALUE OF
-- MAX_IN_LEN. THESE ARE IN ALPHABETIC ORDER. FOLLOWING THESE
-- ARE 36 MORE DEFINITIONS, ALSO IN ALPHABETIC ORDER.
-- EACH DEFINITION IS ACCORDING TO THE FOLLOWING FORMAT:
-- A. A NUMBER OF LINES PRECEDED BY THE ADA COMMENT DELIMITER, --.
-- THE FIRST OF THESE LINES CONTAINS THE MACRO SYMBOL AS IT APPEARS
-- IN THE TEST FILES (WITH THE DOLLAR SIGN). THE NEXT FEW "COMMENT"
-- LINES CONTAIN A DESCRIPTION OF THE VALUE TO BE SUBSTITUTED,
-- THE REMAINING "COMMENT" LINES, THE FIRST OF WHICH BEGINS WITH THE
-- WORDS "USED IN: " (NO QUOTES), CONTAIN A LIST OF THE TEST FILES
-- (WITHOUT THE .TST EXTENSION) IN WHICH THE MACRO SYMBOL APPEARS.
-- EACH TEST FILE NAME IS PRECEDED BY ONE OR MORE BLANKS.
-- B. A LINE, WITHOUT THE COMMENT DELIMITER, CONSISTING OF THE
-- IDENTIFIER (WITHOUT THE DOLLAR SIGN) OF THE MACRO SYMBOL,
-- FOLLOWED BY A SPACE OR TAB, FOLLOWED BY THE VALUE TO BE
-- SUBSTITUTED. IN THE DISTRIBUTION FILE, A SAMPLE VALUE IS
-- PROVIDED; THIS VALUE MUST BE REPLACED BY A VALUE APPROPRIATE TO
-- THE IMPLEMENTATION.
-- DEFINITIONS ARE SEPARATED BY ONE OR MORE EMPTY LINES.
-- THE LIST OF DEFINITIONS BEGINS AFTER THE FOLLOWING EMPTY LINE.
```
B.1 Macro Substitution File

-- $MAX_IN_LEN
-- AN INTEGER LITERAL GIVING THE MAXIMUM LENGTH PERMITTED BY THE
-- COMPILER FOR A LINE OF ADA SOURCE CODE (NOT INCLUDING AN END-OF-LINE
-- CHARACTER).
-- USED IN: A26007A
MAX_IN_LEN  60

...

B.2 Macro Substitution Tests

The following test files contain the special symbols used for substituting implementation-specific values, as described in Section B.1. This list also appears in the ACATS 4.0 "support" directory as TSTTESTS.DAT.

A26007A.TST   BD2A02A.TST   C35502F.TST   CD2C11D.TST
AD8011A.TST   BD2C01D.TST   C35503D.TST   CD4041A.TST
B22001A.TST   BD2C02A.TST   C35503F.TST   CD7101G.TST
B22001B.TST   BD2C03A.TST   C45231D.TST   CE2102C.TST
B22001C.TST   BD4006A.TST   C4A007A.TST   CE2102H.TST
B22001D.TST   BD8001A.TST   C87B62D.TST   CE2103A.TST
B22001E.TST   BD8002A.TST   C96005B.TST   CE2103B.TST
B22001F.TST   BD8003A.TST   C1225A.TST    CE2203A.TST
B22001G.TST   BD8004A.TST   CD1009K.TST   CE2403A.TST
B22001H.TST   BD8004B.TST   CD1009T.TST   CE3002B.TST
B22001I.TST   BD8004C.TST   CD1009U.TST   CE3002C.TST
B22001K.TST   C23003A.TST   CD1C03E.TST   CE3102B.TST
B22001L.TST   C23003B.TST   CD1C06A.TST   CE3107A.TST
B22001M.TST   C23003G.TST   CD2A83C.TST   CE3304A.TST
B22001N.TST   C23003I.TST   CD2A91C.TST   SPPRT13S.TST
B54B01B.TST   C35502D.TST   CD2C11A.TST   

B.3 Package ImpDef and Its Children

The package ImpDef (for "Implementation Definitions") provides constants and functions for producing implementation-specific values required by certain test programs. This package resides in the file ImpDef.a in the "support" directory. Four child packages are also included in the "support" directory, each providing the means for producing implementation-specific values required by certain test programs for a particular Specialized Needs Annex. These packages have names of the form ImpDef.Annex_X, and reside in files with names of the form ImpDefX.a, where 'X' is replaced by the letter designating the relevant Annex.

The ImpDef package and each of its children should be modified for each implementation as described in the source code. The following excerpt from the "ImpDef.a" file illustrates how these modification points are indicated in the packages.

package ImpDef is
-- The following boolean constants indicate whether this validation will
-- include any of annexes C-H. The values of these booleans affect the
-- behavior of the test result reporting software.
--
-- True means the associated annex IS included in the validation.
-- False means the associated annex is NOT included.
Validating_Annex_C : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_D : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_E : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_F : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_G : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_H : constant Boolean := False;  
-- ^^^^^ --- MODIFY HERE AS NEEDED

-- This is the minimum time required to allow another task to get 
-- control. It is expected that the task is on the Ready queue.  
-- A duration of 0.0 would normally be sufficient but some number 
-- greater than that is expected.
Minimum_Task_Switch : constant Duration := 0.1;  
-- ^^^ --- MODIFY HERE AS NEEDED

...
Annex C: Results of CZ Tests

The "CZ" tests are executed before any other ACATS tests to ensure that the ImpDef packages have been properly customized and that certain support units are working properly. These tests are not considered as Passed or Failed. If they do not perform as expected, the problems must be identified and resolved before conformity testing can continue.

This Appendix presents sample results from executing the "CZ" tests. The actual output will differ, at least in the time-stamp information.

C.1 Sample Output From CZ0004

The following is the output from an execution of CZ0004 with a specific implementation. Note that it contains failure messages (indicated by '*') that are expected, as is the final FAILED report. Note also that certain report lines depend on the customization of the ImpDef package, and will vary with the implementation.

```
..., CZ00004 ACATS 4.0 14-08-31 09:53:13
---- CZ00004 Check that Impdef values have been supplied for the special needs annexes. Check that the routines in TCTouch work correctly.
- CZ00004 TCTouch ACATS 4.0.
* CZ00004 Assertion failed: Assertion Failed is expected.
* CZ00004 Assertion failed: Assertion Failed is expected.
* CZ00004 z should not equal Z Expecting: z Got: Z.
- CZ00004 Three failure messages should have occurred so far.
* CZ00004 Trace Overflow:
  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- CZ00004 A Trace Overflow message should have just occurred.
- CZ00004 <<<><><><>< Annex VALIDATION STATUS >>>><><><><.
+ CZ00004 Annex C validation: Annex C not supported.
+ CZ00004 Annex D validation: Annex D not supported.
+ CZ00004 Annex E validation: Annex E not supported.
+ CZ00004 Annex F validation: Annex F not supported.
+ CZ00004 Annex G validation: Annex G not supported.
+ CZ00004 Annex H validation: Annex H not supported.
- CZ000004 <<<><><><>< IMPDEF <<<><><><><.
- CZ00004 Validating_Annex_C : FALSE.
- CZ00004 Validating_Annex_D : FALSE.
- CZ00004 Validating_Annex_E : FALSE.
- CZ00004 Validating_Annex_F : FALSE.
- CZ00004 Validating_Annex_G : FALSE.
- CZ00004 Validating_Annex_H : FALSE.
- CZ00004 Minimum_Task_Switch: 0.100000000
- CZ00004 Switch_To_New_Task: 1.000000000
- CZ00004 Clear_Ready_Queue: 5.000000000
- CZ00004 Delay_For_Time_Past: 0.100000000
- CZ00004 Time_Dependent_Reset: 0.300000000
- CZ00004 Delay_Per_Random_Test: 1.000000000
- CZ00004 Exceed_Time_Slice.
- CZ00004 Non_State_String: By No Means A State.
- CZ00004 External_Tag_Value: implementation_defined.
- CZ00004 CD30005_1_Foreign_Address: present.
- CZ00004 CD30005_1_External_Name: CD30005_1.
- CZ00004 Max_Default_Alignment: 1.
- CZ00004 Max_Linker_Alignment: 1.
- CZ00004 CXB30130_External_Name: CXB30130.
- CZ00004 CXB30131_External_Name: CXB30131.
**** CZ00004 FAILED ******************************************.
```
C.2 Sample Output From CZ1101A

The following is the output of CZ1101A from a particular implementation. The output of this test should have the same messages (except for the time-stamp) and reported results as indicated here (including failure reports), and the format of the output lines should be as described in the output text.

- NO_NAME (CZ1101A) CHECK REPORT ROUTINES.
- NO_NAME INITIAL VALUES SHOULD BE 'NO_NAME' AND 'FAILED'.
**** NO_NAME FAILED ************************************.

**NO_NAME** FAILED ****************************.

- PASS_TEST ACATS 4.0 14-08-31 09:57:42
--- PASS_TEST CHECKING 'TEST' AND 'RESULT' FOR 'PASSED'.
- PASS_TEST THIS LINE IS EXACTLY 'MAX_LEN' LONG. ...
- PASS_TEST THIS COMMENT HAS A WORD THAT SPANS THE FOLD POINT. THIS
  COMMENT FITS EXACTLY ON TWO LINES. ...
- PASS_TEST THIS COMMENT IS ONE VERY LONG WORD AND SO IT SHOULD BE
  SPLIT AT THE FOLD POINT.

==== PASS_TEST PASSED ================.

- NO_NAME CHECK THAT 'RESULT' RESETS VALUES TO 'NO_NAME' AND
  'FAILED'.
**** NO_NAME FAILED ************************************.

FAIL_TEST ACATS 4.0 14-08-31 09:57:42
--- FAIL_TEST CHECKING 'FAILED' AND 'RESULT' FOR 'FAILED'.
* FAIL_TEST 'RESULT' SHOULD NOW BE 'FAILED'.
**** FAIL_TEST FAILED ************************************.

NA_TEST ACATS 4.0 14-08-31 09:57:42
--- NA_TEST CHECKING 'NOT-APPLICABLE'.
  + NA_TEST 'RESULT' SHOULD NOW BE 'NOT-APPLICABLE'.
  +++ NA_TEST NOT-APPLICABLE ++++++++++++++++++++.

FAIL_NA_TEST ACATS 4.0 14-08-31 09:57:42
--- FAIL_NA_TEST CHECKING 'NOT_APPLICABLE', 'FAILED', 'NOT-APPLICABLE'.
  + FAIL_NA_TEST 'RESULT' BECOMES 'NOT-APPLICABLE'.
* FAIL_NA_TEST 'RESULT' BECOMES 'FAILED'.
  + FAIL_NA_TEST CALLING 'NOT-APPLICABLE' DOESN'T CHANGE 'RESULT'.
**** FAIL_NA_TEST FAILED ************************************.

SPEC_NA_TEST ACATS 4.0 14-08-31 09:57:42
--- SPEC_NA_TEST CHECKING 'SPEC_ACT', 'NOT-APPLICABLE', 'SPEC_ACT'.
  ! SPEC_NA_TEST 'RESULT' BECOMES 'TENTATIVELY PASSED'.
  + SPEC_NA_TEST 'RESULT' BECOMES 'NOT APPLICABLE'.
  ! SPEC_NA_TEST CALLING 'SPECIAL_ACTION' DOESN'T CHANGE 'RESULT'.
  +++ SPEC_NA_TEST NOT-APPLICABLE ++++++++++++++++++++.

SPEC_FAIL_NA_TEST ACATS 4.0 14-08-31 09:57:42
--- SPEC_FAIL_NA_TEST CHECKING 'SPEC_ACT', 'FAILED', 'SPEC_ACT'.
  ! SPEC_FAIL_NA_TEST 'RESULT' BECOMES 'TENTATIVELY PASSED'.
  * SPEC_FAIL_NA_TEST 'RESULT' BECOMES 'FAILED'.
  ! SPEC_FAIL_NA_TEST CALLING 'SPECIAL_ACTION' DOESN'T CHANGE 'RESULT'.
**** SPEC_FAIL_NA_TEST FAILED ************************************.

CZ1101A ACATS 4.0 14-08-31 09:57:42
--- CZ1101A CHECKING 'SPECIAL_ACTION' ALONE.
  ! CZ1101A 'RESULT' BECOMES 'TENTATIVELY PASSED'.
!!! CZ1101A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!! SEE '! ' COMMENTS FOR SPECIAL NOTES!!!
C.3 Sample Output From CZ1102A

Test CZ1102A should execute and report PASSED as illustrated below. Only the time-stamp should differ.

,,,.  CZ1102A  ACATS 4.0  14-08-31  09:57:57
----  CZ1102A  CHECK THAT THE DYNAMIC VALUE ROUTINES OF THE REPORT PACKAGE
         WORK CORRECTLY.
====  CZ1102A  PASSED ============================.

C.4 Sample Output From CZ1103A

Test CZ1103A may produce two different forms of output, depending on whether the implementation supports external files.

C.4.1 Output When External Files Are Supported

If the implementation under test supports the creation and use of external text files, then test CZ1103A should produce the following report (except for differences in the time stamp). Note that failure messages are expected.

,,,.  CZ1103A  ACATS 4.0  14-08-31  09:58:10
----  CZ1103A  CHECK THAT PROCEDURE CHECK_FILE WORKS.
-  CZ1103A  BEGIN TEST WITH AN EMPTY FILE.
-  CZ1103A  BEGIN TEST WITH A FILE WITH BLANK LINES.
-  CZ1103A  BEGIN TEST WITH A FILE WITH BLANK LINES AND PAGES.
-  CZ1103A  BEGIN TEST WITH A FILE WITH TRAILING BLANKS.
-  CZ1103A  FROM CHECK_FILE: THIS IMPLEMENTATION PADS LINES WITH
         BLANKS.
-  CZ1103A  BEGIN TEST WITH A FILE WITHOUT TRAILING BLANKS.
-  CZ1103A  BEGIN TEST WITH A FILE WITH AN END OF LINE ERROR.
*  CZ1103A  FROM CHECK_FILE: END OF LINE EXPECTED - E ENCOUNTERED.
-  CZ1103A  FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
         REVEALED ERROR: THIS LINE WILL CONTAIN AN #.
-  CZ1103A  BEGIN TEST WITH FILE WITH END OF PAGE ERROR.
*  CZ1103A  FROM CHECK_FILE: END_OF_PAGE NOT WHERE EXPECTED.
-  CZ1103A  FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
         REVEALED ERROR: THIS LINE WILL CONTAIN AN @.
-  CZ1103A  BEGIN TEST WITH FILE WITH INCORRECT DATA.
*  CZ1103A  FROM CHECK_FILE: FILE DOES NOT CONTAIN CORRECT OUTPUT -
         EXPECTED C - GOT I.
-  CZ1103A  FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
         REVEALED ERROR: LINE WITH C.
****  CZ1103A  FAILED *****************************.
,,,.  CZ1103A  ACATS 4.0  14-08-31  09:58:10
----  CZ1103A  THE LINE ABOVE SHOULD REPORT FAILURE.
!  CZ1103A  COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!!  CZ1103A  TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!  SEE '!' COMMENTS FOR SPECIAL NOTES!!
C.4.2 Output When External Files Are Not Supported

If the implementation under test does not support external text files, then CZ1103A produces different output, as illustrated below.

,.,. CZ1103A ACATS 4.0 14-08-31 09:59:00
--- CZ1103A CHECK THAT PROCEDURE CHECK_FILE WORKS.
- CZ1103A BEGIN TEST WITH AN EMPTY FILE.
  * CZ1103A TEST WITH EMPTY FILE INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES.
  * CZ1103A TEST WITH FILE WITH BLANK LINES INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES AND PAGES.
  * CZ1103A TEST WITH FILE WITH BLANK PAGES INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITH TRAILING BLANKS.
  * CZ1103A TEST WITH FILE WITH TRAILING BLANKS INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITHOUT TRAILING BLANKS.
  * CZ1103A TEST WITH FILE WITHOUT TRAILING BLANKS INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITH AN END OF LINE ERROR.
  * CZ1103A TEST WITH END_OF_LINE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH END OF PAGE ERROR.
  * CZ1103A TEST WITH END_OF_PAGE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH END OF FILE ERROR.
  * CZ1103A TEST WITH END_OF_FILE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH INCORRECT DATA.
  * CZ1103A TEST WITH INCORRECT DATA INCOMPLETE.
*** CZ1103A FAILED ********************************************.

,.,. CZ1103A ACATS 4.0 14-08-31 09:59:00
--- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!!!!!.
!!!! SEE '!' COMMENTS FOR SPECIAL NOTES!!
Annex D: Test Applicability Criteria

Certain tests in the suite may be considered inapplicable to an implementation depending on the way the implementation treats the implementation-dependent features of the language. A brief summary of these implementation-dependent features and the tests they affect are listed in this appendix.

Note that the applicability of each one of these tests is based on the criteria listed in the test file. During validation, all the implementation-dependent tests are submitted for compilation and (if compiled successfully) are executed, with the following exceptions:

- Tests which require a floating point digits value that exceeds System.Max_Digits need not be submitted to the compiler. (The testing laboratory may require pre-validation evidence that the tests are properly rejected.)
- If file I/O is not supported, then the tests listed in Section D.2.13 will not be part of the customized test suite for bare target validations and will not be run during witness testing.

D.1 Compile-Time Inapplicability

The first part of this appendix is concerned with tests for which the applicability is determined at compile time. Class B tests that are inapplicable should be successfully compiled, or, in a few cases, should report an error on the line marked "--N/A => ERROR". Executable tests that are inapplicable based on their compile-time behavior must be rejected as a result of the unsupported feature. Lines containing the implementation-dependent features are marked "--N/A => ERROR". In every case, tests may be graded NOT-APPLICABLE only if all the following conditions are met:

- The implementation's treatment of the test is consistent with the applicability criteria given in the test comments;
- All other tests having the same applicability criteria exhibit the same behavior; and
- The behavior is consistent with the implementation's documentation.

D.1.1 Type Short_Integer

If there is no predefined type Short_Integer, then the tests contained in the following files are not applicable:

- B36105C.DEP
- B52004E.DEP
- B55B09D.DEP
- C45231B.DEP
- C45304B.DEP
- C45411B.DEP
- C45611B.DEP
- C45613B.DEP
- C45614B.DEP
- C45631B.DEP
- C45632B.DEP
- C55B07B.DEP
- CD7101E.DEP

D.1.2 Type Long_Integer

If there is no predefined type Long_Integer, then the tests contained in the following files are not applicable:

- B52004D.DEP
- B55B09C.DEP
- C45231C.DEP
- C45304C.DEP
- C45502C.DEP
- C45503C.DEP
- C45504C.DEP
- C45504F.DEP
- C45611C.DEP
- C45613C.DEP
- C45614C.DEP
- C45631C.DEP
- C45632C.DEP
- C55B07A.DEP
- CD7101F.DEP
D.1.3 Other Predefined Integer Types

If there are no predefined integer types with names other than Integer, Short_Integer, and Long_Integer, then the tests contained in the following files are not applicable.

C45231D.TST          CD7101G.TST

D.1.4 Fixed Point Restrictions

If System.Max_Mantissa is less than 47 or System.Fine_Delta is greater than 2.0**-47, then the tests contained in the following files are not applicable:

C45531M.DEP          C45531O.DEP          C45532M.DEP          C45532O.DEP
C45531N.DEP          C45531P.DEP          C45532N.DEP          C45532P.DEP

D.1.5 Non-binary Values of 'Small

If 'Small representation clauses which are not powers of two are not supported, then the tests contained in the following files are not applicable:

C45536A.DEP          CD2A53A.ADA

D.1.6 Compiler Rejection of Supposedly Static Expression

Consider the following declarations:

```plaintext
type F is digits System.Max_Digits;
N : constant := 2.0 * F'Machine_Radix ** F'Machine_EMax;
```

If the declaration of N is rejected on the grounds that evaluation of the expression will raise an exception, then the following test is not applicable:

C4A013B.ADA

D.1.7 Machine Code Insertions

If machine code insertions are not supported, then the tests contained in the following files are not applicable:

AD8011A.TST          BD8002A.TST          BD8004A.TST          BD8004C.TST
BD8001A.TST          BD8003A.TST          BD8004B.TST

D.1.8 Illegal External File Names

If there are no strings which are illegal as external file names, then the tests contained in the following files are not applicable:

CE2102C.TST          CE2102H.TST          CE3102B.TST          CE3107A.TST
D.1.9 Decimal Types
If decimal types are not supported, then the tests contained in the following files are not applicable: (Note that implementations testing Annex F must support decimal types.)
C460011.A CXAA010.A

D.1.10 Instantiation of Sequential_IO with indefinite types
If Sequential_IO does not support indefinite types, then the test contained in the following files are not applicable:
CE2201D.DEP CE2201E.DEP

D.1.11 Special Handling Tests
Test requiring special handling may also be not applicable. See section 5.5.5, “Tests with Special Processing Requirements”, for details.

D.2 Reported Inapplicability
This section is concerned with tests that can detect, at runtime, certain implementation characteristics that render the objective meaningless or prevent testing of the objective. These tests must compile and execute, reporting "NOT_APPLICABLE" as the result. This behavior must be consistent with other tests for related objective and with the implementation's documentation.

D.2.1 Value of Machine_Overflows is False
If Machine_Overflows is False for floating point types, then the tests contained in the following files should report NOT_APPLICABLE:
C45322A.ADA C45523A.ADA C4A012B.ADA

D.2.2 System.Max_Digits
If the value of System.Max_Digits is greater than 35, then the test contained in the following file should report NOT_APPLICABLE:
C4A011A.ADA

D.2.3 Floating Point Overflow
Consider the declaration
\[
\text{type } F \text{ is digits System.Max_Digits;}
\]
If \( F'\text{Machine_Overflows} = \text{False} \) and \( 2.0*F'\text{Machine_Radix}**F'\text{Machine_EMax} \leq F'\text{Base}'\text{Last} \) then the test contained in the following file should report NOT_APPLICABLE (if it compiles without error):
C4A013B.ADA
D.2.4 Type Duration

If Duration\'First = Duration\'Base\'First or Duration\'Last = Duration\'Base\'Last then the tests contained in the following file should report NOT_APPLICABLE:

C96005B.TST

D.2.5 Text Files (Non-supported Features)

If Use_Error or Name_Error is raised by every attempt to create or open a text file (this is the appropriate behavior for an implementation which does not support text files other than standard input and output), then the tests contained in the following files should report NOT_APPLICABLE:

CE2109C.ADA  CE3403E.ADA  CE3602C.ADA  CE3804G.ADA
CE3102A.ADA  CE3403F.ADA  CE3602D.ADA  CE3804H.ADA
CE3102B.TST  CE3404B.ADA  CE3603A.ADA  CE3804I.ADA
CE3102F.ADA  CE3404C.ADA  CE3604A.ADA  CE3804J.ADA
CE3102G.ADA  CE3404D.ADA  CE3604B.ADA  CE3804M.ADA
CE3102H.ADA  CE3405A.ADA  CE3605A.ADA  CE3804O.ADA
CE3102I.ADA  CE3405C.ADA  CE3605B.ADA  CE3804P.ADA
CE3102K.ADA  CE3405D.ADA  CE3605C.ADA  CE3805A.ADA
CE3103A.ADA  CE3406A.ADA  CE3605D.ADA  CE3805B.ADA
CE3104A.ADA  CE3406B.ADA  CE3605E.ADA  CE3806A.ADA
CE3104B.ADA  CE3406C.ADA  CE3606A.ADA  CE3806B.ADA
CE3104C.ADA  CE3406D.ADA  CE3606B.ADA  CE3806D.ADA
CE3106A.ADA  CE3407A.ADA  CE3704A.ADA  CE3806E.ADA
CE3106B.ADA  CE3407B.ADA  CE3704B.ADA  CE3806G.ADA
CE3107A.TST  CE3407C.ADA  CE3704C.ADA  CE3806H.ADA
CE3107B.ADA  CE3408A.ADA  CE3704D.ADA  CE3902B.ADA
CE3108A.ADA  CE3408B.ADA  CE3704E.ADA  CE3904A.ADA
CE3108B.ADA  CE3408C.ADA  CE3704F.ADA  CE3904B.ADA
CE3110A.ADA  CE3409A.ADA  CE3704M.ADA  CE3905A.ADA
CE3112C.ADA  CE3409C.ADA  CE3704N.ADA  CE3905B.ADA
CE3112D.ADA  CE3409D.ADA  CE3704O.ADA  CE3905C.ADA
CE3114A.ADA  CE3409E.ADA  CE3705A.ADA  CE3905L.ADA
CE3115A.ADA  CE3410A.ADA  CE3705B.ADA  CE3906A.ADA
CE3207A.ADA  CE3410C.ADA  CE3705C.ADA  CE3906B.ADA
CE3301A.ADA  CE3410D.ADA  CE3705D.ADA  CE3906C.ADA
CE3302A.ADA  CE3410E.ADA  CE3705E.ADA  CE3906E.ADA
CE3304A.TST  CE3411A.ADA  CE3706D.ADA  CE3906F.ADA
CE3305A.ADA  CE3411C.ADA  CE3706F.ADA  CXAA001.A
CE3401A.ADA  CE3412A.ADA  CE3706G.ADA  CXAA002.A
CE3402A.ADA  CE3413A.ADA  CE3804A.ADA  CXAA003.A
CE3402C.ADA  CE3413B.ADA  CE3804B.ADA  CXAA004.A
CE3402D.ADA  CE3413C.ADA  CE3804C.ADA  CXAA005.A
CE3403A.ADA  CE3414A.ADA  CE3804D.ADA  CXAA006.A
CE3403B.ADA  CE3602A.ADA  CE3804E.ADA  CXAA007.A
CE3403C.ADA  CE3602B.ADA  CE3804F.ADA  CXAA008.A
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a text file with mode `In_File`, then the tests contained in the following files should report NOT_APPLICABLE:

- CE3102F.ADA
- CE3407C.ADA
- CE3605C.ADA
- CE3804H.ADA
- CE3102H.ADA
- CE3408A.ADA
- CE3704A.ADA
- CE3804I.ADA
- CE3104B.ADA
- CE3408C.ADA
- CE3704C.ADA
- CE3804J.ADA
- CE3106A.ADA
- CE3409C.ADA
- CE3704D.ADA
- CE3804M.ADA
- CE3108A.ADA
- CE3409D.ADA
- CE3704E.ADA
- CE3804P.ADA
- CE3108B.ADA
- CE3409E.ADA
- CE3704F.ADA
- CE3805A.ADA
- CE3112D.ADA
- CE3410C.ADA
- CE3704M.ADA
- CE3805B.ADA
- CE3301A.ADA
- CE3410D.ADA
- CE3704N.ADA
- CE3806A.ADA
- CE3302A.ADA
- CE3410E.ADA
- CE3704O.ADA
- CE3806B.ADA
- CE3402A.ADA
- CE3411A.ADA
- CE3705B.ADA
- CE3806D.ADA
- CE3403B.ADA
- CE3411C.ADA
- CE3705C.ADA
- CE3806G.ADA
- CE3403C.ADA
- CE3412A.ADA
- CE3705D.ADA
- CE3902B.ADA
- CE3403E.ADA
- CE3413A.ADA
- CE3705E.ADA
- CE3904B.ADA
- CE3403F.ADA
- CE3413B.ADA
- CE3706D.ADA
- CE3905A.ADA
- CE3404B.ADA
- CE3413C.ADA
- CE3706G.ADA
- CE3905C.ADA
- CE3404C.ADA
- CE3602A.ADA
- CE3804A.ADA
- CE3905L.ADA
- CE3404D.ADA
- CE3602B.ADA
- CE3804B.ADA
- CE3906B.ADA
- CE3406A.ADA
- CE3602D.ADA
- CE3804D.ADA
- CE3906C.ADA
- CE3406C.ADA
- CE3603A.ADA
- CE3804E.ADA
- CE3903A.ADA
- CE3406D.ADA
- CE3604A.ADA
- CE3804F.ADA
- CE3902A.ADA
- CE3407A.ADA
- CE3604B.ADA
- CE3804G.ADA
- CE3412C.ADA

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a text file with mode `Out_File`, then the tests contained in the following files should report NOT_APPLICABLE:

- CE2109C.ADA
- CE3110A.ADA
- CE3403F.ADA
- CE3409C.ADA
- CE3102A.ADA
- CE3110C.ADA
- CE3404B.ADA
- CE3409D.ADA
- CE3102B.TST
- CE3112D.ADA
- CE3404C.ADA
- CE3409D.ADA
- CE3102F.ADA
- CE3114A.ADA
- CE3404D.ADA
- CE3410A.ADA
- CE3102G.ADA
- CE3115A.ADA
- CE3405A.ADA
- CE3410C.ADA
- CE3102H.ADA
- CE3207A.ADA
- CE3405C.ADA
- CE3410D.ADA
- CE3102J.ADA
- CE3301A.ADA
- CE3405D.ADA
- CE3410E.ADA
- CE3102K.ADA
- CE3302A.ADA
- CE3406A.ADA
- CE3411A.ADA
- CE3103A.ADA
- CE3304A.TST
- CE3406B.ADA
- CE3411C.ADA
- CE3104A.ADA
- CE3305A.ADA
- CE3406C.ADA
- CE3412A.ADA
- CE3104B.ADA
- CE3401A.ADA
- CE3406D.ADA
- CE3413A.ADA
- CE3104C.ADA
- CE3402A.ADA
- CE3407A.ADA
- CE3413B.ADA
- CE3106A.ADA
- CE3402C.ADA
- CE3407B.ADA
- CE3413C.ADA
- CE3106B.ADA
- CE3402D.ADA
- CE3407C.ADA
- CE3414A.ADA
- CE3107A.TST
- CE3403A.ADA
- CE3408A.ADA
- CE3602A.ADA
- CE3107B.ADA
- CE3403B.ADA
- CE3408B.ADA
- CE3602B.ADA
- CE3108A.ADA
- CE3403C.ADA
- CE3408C.ADA
- CE3602C.ADA
- CE3108B.ADA
- CE3403E.ADA
- CE3409A.ADA
- CE3602D.ADA
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a text file with mode `Append_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXAA001.A</td>
<td>CXAA006.A</td>
<td>CXAA008.A</td>
<td>CXAA015.A</td>
</tr>
<tr>
<td>CXAA002.A</td>
<td>CXAA007.A</td>
<td>CXAA013.A</td>
<td></td>
</tr>
</tbody>
</table>

If `Reset` is not supported for text files, then the following tests should report `NOT_APPLICABLE`:

- CE3104C.ADA
- CE3115A.ADA
- CXAA020.A

If `Delete` is not supported for text files, then the following tests should report `NOT_APPLICABLE`:

- CE3110A.ADA
- CE3114A.ADA

If association of multiple internal text files (opened for reading and writing) to a single external file is not supported, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE3115A.ADA

If there are no inappropriate values for either line length or page length, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE3304A.TST

If the value of `Count'Last` is greater than 150_000, then the following test should report `NOT_APPLICABLE`:

- CE3413B.ADA
D.2.6 Text Files (Supported Features)

If Create with mode In_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:

CE3102E.ADA

If Open with mode In_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:

CE3102J.ADA

If Create with mode Out_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:

CE3102I.ADA

If Open with mode Out_File is supported for text files, then the following test should report NOT_APPLICABLE:

CE3102K.ADA

If Reset is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:

CE3102F.ADA

If Delete for text files is supported, then the test contained in the following file should report NOT_APPLICABLE:

CE3102G.ADA

D.2.7 Sequential Files (Non-supported Features)

If Use_Error or Name_Error is raised by every attempt to create or open a sequential file (this is appropriate behavior for an implementation which does not support sequential files), then the tests contained in the following files should report NOT_APPLICABLE:

CE2102A.ADA  CE2106A.ADA  CE2201D.DEP  CE2204A.ADA
CE2102C.TST  CE2108E.ADA  CE2201E.DEP  CE2204B.ADA
CE2102G.ADA  CE2108F.ADA  CE2201F.ADA  CE2204C.ADA
CE2102N.ADA  CE2109A.ADA  CE2201G.ADA  CE2204D.ADA
CE2102O.ADA  CE2110A.ADA  CE2201H.ADA  CE2205A.ADA
CE2102P.ADA  CE2111A.ADA  CE2201I.ADA  CE2206A.ADA
CE2102Q.ADA  CE2111C.ADA  CE2201J.ADA  CE2208B.ADA
CE2102X.ADA  CE2111F.ADA  CE2201K.ADA  CXA8001.A
CE2103A.TST  CE2111L.ADA  CE2201L.ADA  CXA8002.A
CE2103C.ADA  CE2201A.ADA  CE2201M.ADA
CE2104A.ADA  CE2201B.ADA  CE2201N.ADA
CE2104B.ADA  CE2201C.ADA  CE2203A.TST

If Use_Error or Name_Error is raised by every attempt to create or open a sequential file with mode In_File, then the tests contained in the following files should report NOT_APPLICABLE:
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a sequential file with mode `Out_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2102A.ADA  CE2104B.ADA  CE2201B.ADA  CE2201M.ADA
CE2102C.TST  CE2106A.ADA  CE2201C.ADA  CE2201N.ADA
CE2102G.ADA  CE2108E.ADA  CE2201D.DEP  CE2203A.TST
CE2102N.ADA  CE2108F.ADA  CE2201E.DEP  CE2204A.ADA
CE2102O.ADA  CE2109A.ADA  CE2201F.ADA  CE2204B.ADA
CE2102P.ADA  CE2110A.ADA  CE2201G.ADA  CE2204C.ADA
CE2102Q.ADA  CE2111A.ADA  CE2201H.ADA  CE2204D.ADA
CE2102X.ADA  CE2111C.ADA  CE2201I.ADA  CE2205A.ADA
CE2103A.TST  CE2111F.ADA  CE2201J.ADA  CE2206A.ADA
CE2103C.ADA  CE2111L.ADA  CE2201K.ADA  CE2208B.ADA
CE2104A.ADA  CE2201A.ADA  CE2201L.ADA  CXA8001.A

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a sequential file with mode `APPEND_FILE`, then the tests contained in the following files should report `NOT_APPLICABLE`:

CXA8001.A

If `reset` to mode `Out_File` is not supported for sequential files, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2111C.ADA  CE2111F.ADA  CE2111L.ADA

If `reset` to mode `In_File` is not supported for sequential files, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2111F.ADA  CE2111L.ADA  CE2204C.ADA

If `Delete` for sequential files is not supported, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2106A.ADA  CE2110A.ADA

If the implementation cannot restrict the file capacity for a sequential file, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2203A.TST

### D.2.8 Sequential Files (Supported Features)

If `Create` with mode `In_File` is supported for sequential files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102D.ADA
If Open with mode In_File is supported for sequential files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102N.ADA

If Reset to mode In_File is supported for sequential files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102O.ADA

If Create with mode Out_File is supported, then the test contained in the following file should report NOT_APPLICABLE:

CE2102E.ADA

If Open with mode Out_File is supported, then the test contained in the following file should report NOT_APPLICABLE:

CE2102P.ADA

If Reset to mode Out_File is supported for sequential files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102Q.ADA

D.2.9 Direct Files (Non-supported Features)

If Use_Error or Name_Error is raised by every attempt to create or open a direct access file (this is appropriate behavior for an implementation which does not support direct access files), then the tests contained in the following files should report NOT_APPLICABLE:

CE2102B.ADA    CE2104D.ADA    CE2401F.ADA    CE2408A.ADA
CE2102H.TST    CE2106B.ADA    CE2401H.ADA    CE2408B.ADA
CE2102K.ADA    CE2108G.ADA    CE2401I.ADA    CE2409A.ADA
CE2102R.ADA    CE2108H.ADA    CE2401J.ADA    CE2409B.ADA
CE2102S.ADA    CE2109B.ADA    CE2401K.ADA    CE2410A.ADA
CE2102T.ADA    CE2110C.ADA    CE2401L.ADA    CE2410B.ADA
CE2102U.ADA    CE2111B.ADA    CE2403A.TST    CE2411A.ADA
CE2102V.ADA    CE2111E.ADA    CE2404A.ADA    CXA8003.A
CE2102W.ADA    CE2111G.ADA    CE2404B.ADA    CXA9001.A
CE2102Y.ADA    CE2401A.ADA    CE2405B.ADA    CXA9002.A
CE2103B.TST    CE2401B.ADA    CE2406A.ADA
CE2103D.ADA    CE2401C.ADA    CE2407A.ADA
CE2104C.ADA    CE2401E.ADA    CE2407B.ADA

If Use_Error or Name_Error is raised by every attempt to create or open a direct access file with mode In_File, then the tests contained in the following files should report NOT_APPLICABLE:

CE2102K.ADA    CE2111B.ADA    CE2401E.ADA    CE2406A.ADA
CE2102U.ADA    CE2111E.ADA    CE2401F.ADA    CE2407A.ADA
CE2104C.ADA    CE2401A.ADA    CE2401H.ADA    CE2411A.ADA
CE2104D.ADA    CE2401B.ADA    CE2401I.ADA
CE2108H.ADA    CE2401C.ADA    CE2405B.ADA
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file with mode `Out_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

- CE2102B.ADA
- CE2102K.ADA
- CE2102R.ADA
- CE2102W.ADA
- CE2102Y.ADA
- CE2103B.TST
- CE2103D.ADA
- CE2104C.ADA

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file with mode `InOut_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

- CE2102H.TST
- CE2102K.ADA
- CE2102S.ADA
- CE2102T.ADA
- CE2102U.ADA
- CE2102V.ADA
- CE2109B.ADA

If `Delete` for direct access files is not supported, then the following tests should report `NOT_APPLICABLE`:

- CE2106B.ADA
- CE2110C.ADA

If the implementation cannot restrict the file capacity for a direct file, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE2403A.TST

### D.2.10 Direct Files (Supported Features)

If `Create` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE2102I.ADA

If `Open` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE2102T.ADA

If `Reset` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE2102U.ADA

If `Create` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE2102J.ADA
If `Open` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

`CE2102V.ADA`

If `Reset` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

`CE2102W.ADA`

If `Create` with mode `InOut_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

`CE2102F.ADA`

If `Open` with mode `InOut_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

`CE2102R.ADA`

If `Reset` to mode `InOut_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

`CE2102S.ADA`

**D.2.11 Stream Files (Non-supported Features)**

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a stream file (this is the appropriate behavior for an implementation which does not support stream files), then the tests contained in the following files should report `NOT_APPLICABLE`:

- `CXAA019.A`
- `CXAC001.A`
- `CXAC002.A`
- `CXAC003.A`
- `CXAC004.A`
- `CXAC005.A`
- `CXAC006.A`
- `CXAC007.A`
- `CXACA01.A`
- `CXACA02.A`
- `CXACB01.A`
- `CXACB02.A`

**D.2.12 Wide Text Files (Non-supported Features)**

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a wide text file (this is the appropriate behavior for an implementation which does not support wide text files), then the tests contained in the following files should report `NOT_APPLICABLE`:

- `CXAA019.A`
- `CXAB001.A`

**D.2.13 File I/O Tests**

If sequential, text, wide text, wide wide text, direct access, and stream files are not supported, then the tests contained in the following files should report `NOT_APPLICABLE`:

- `CE2102A.ADA`
- `CE2102B.ADA`
- `CE2102C.TST`
- `CE2102G.ADA`
- `CE2102H.TST`
- `CE2102K.ADA`
- `CE2102N.ADA`
- `CE2102O.ADA`
- `CE2102P.ADA`
- `CE2102Q.ADA`
- `CE2102R.ADA`
- `CE2102S.ADA`
CE2102T.ADA  CE2204C.ADA  CE3114A.ADA  CE3602C.ADA
CE2102U.ADA  CE2204D.ADA  CE3115A.ADA  CE3602D.ADA
CE2102V.ADA  CE2205A.ADA  CE3207A.ADA  CE3603A.ADA
CE2102W.ADA  CE2206A.ADA  CE3301A.ADA  CE3604A.ADA
CE2102X.ADA  CE2208B.ADA  CE3302A.ADA  CE3604B.ADA
CE2102Y.ADA  CE2401A.ADA  CE3304A.TST  CE3605A.ADA
CE2103A.TST  CE2401B.ADA  CE3305A.ADA  CE3605B.ADA
CE2103B.TST  CE2401C.ADA  CE3401A.ADA  CE3605C.ADA
CE2103C.ADA  CE2401E.ADA  CE3402A.ADA  CE3605D.ADA
CE2103D.ADA  CE2401F.ADA  CE3402C.ADA  CE3605E.ADA
CE2104A.ADA  CE2401H.ADA  CE3402D.ADA  CE3606A.ADA
CE2104B.ADA  CE2401I.ADA  CE3403A.ADA  CE3606B.ADA
CE2104C.ADA  CE2401J.ADA  CE3403B.ADA  CE3704A.ADA
CE2104D.ADA  CE2401K.ADA  CE3403C.ADA  CE3704B.ADA
CE2106A.ADA  CE2401L.ADA  CE3403E.ADA  CE3704C.ADA
CE2106B.ADA  CE2403A.TST  CE3403F.ADA  CE3704D.ADA
CE2108E.ADA  CE2404A.ADA  CE3404B.ADA  CE3704E.ADA
CE2108F.ADA  CE2404B.ADA  CE3404C.ADA  CE3704F.ADA
CE2108G.ADA  CE2405B.ADA  CE3404D.ADA  CE3704M.ADA
CE2108H.ADA  CE2406A.ADA  CE3405A.ADA  CE3704N.ADA
CE2109A.ADA  CE2407A.ADA  CE3405C.ADA  CE3704O.ADA
CE2109B.ADA  CE2407B.ADA  CE3405D.ADA  CE3705A.ADA
CE2109C.ADA  CE2408A.ADA  CE3406A.ADA  CE3705B.ADA
CE2110A.ADA  CE2408B.ADA  CE3406B.ADA  CE3705C.ADA
CE2110C.ADA  CE2409A.ADA  CE3406C.ADA  CE3705D.ADA
CE2111A.ADA  CE2409B.ADA  CE3406D.ADA  CE3705E.ADA
CE2111B.ADA  CE2410A.ADA  CE3407A.ADA  CE3706D.ADA
CE2111C.ADA  CE2410B.ADA  CE3407B.ADA  CE3706F.ADA
CE2111E.ADA  CE2411A.ADA  CE3407C.ADA  CE3706G.ADA
CE2111F.ADA  CE3102A.ADA  CE3408A.ADA  CE3804A.ADA
CE2111G.ADA  CE3102B.TST  CE3408B.ADA  CE3804B.ADA
CE2111H.ADA  CE3102F.ADA  CE3408C.ADA  CE3804C.ADA
CE2201A.ADA  CE3102G.ADA  CE3409A.ADA  CE3804D.ADA
CE2201B.ADA  CE3102H.ADA  CE3409C.ADA  CE3804E.ADA
CE2201C.ADA  CE3102J.ADA  CE3409D.ADA  CE3804F.ADA
CE2201D.DEP  CE3102K.ADA  CE3409E.ADA  CE3804G.ADA
CE2201E.DEP  CE3103A.ADA  CE3410A.ADA  CE3804H.ADA
CE2201F.ADA  CE3104A.ADA  CE3410C.ADA  CE3804I.ADA
CE2201G.ADA  CE3104B.ADA  CE3410D.ADA  CE3804J.ADA
CE2201H.ADA  CE3104C.ADA  CE3410E.ADA  CE3804M.ADA
CE2201I.ADA  CE3106A.ADA  CE3411A.ADA  CE3804O.ADA
CE2201J.ADA  CE3106B.ADA  CE3411C.ADA  CE3804P.ADA
CE2201K.ADA  CE3107A.TST  CE3412A.ADA  CE3805A.ADA
CE2201L.ADA  CE3107B.ADA  CE3413A.ADA  CE3805B.ADA
CE2201M.ADA  CE3108A.ADA  CE3413B.ADA  CE3806A.ADA
CE2201N.ADA  CE3108B.ADA  CE3413C.ADA  CE3806B.ADA
CE2203A.TST  CE3110A.ADA  CE3414A.ADA  CE3806D.ADA
CE2204A.ADA  CE3112C.ADA  CE3602A.ADA  CE3806E.ADA
CE2204B.ADA  CE3112D.ADA  CE3602B.ADA  CE3806G.ADA
D.2.14 Memory for Allocated Objects

If a large amount of memory (more than 32 megabytes for a typical implementation) is available for allocated objects (those created by `new`), then the test contained in the following file should report NOT_APPLICABLE:

CB10002.A

D.2.15 Task Attributes

If Annex C (Systems Programming) is tested and the size of a task attribute is limited such that an attribute of a controlled type is not supported, then the test contained in the following file should report NOT_APPLICABLE:

CXC7003.A

D.2.16 Reserved Interrupts

If Annex C (Systems Programming) is tested and no interrupts are reserved, then the tests contained in the following files should report NOT_APPLICABLE:

CXC3002.A  CXC3005.A

D.2.17 Multiprocessor Systems

If Annex D (Real-Time Systems) is tested and the target is a multiprocessor, then the tests contained in the following files should report NOT_APPLICABLE:

D.2.18 Non-binary Machine Radix

If Annex G (Numerics) is tested and the machine radix is not a power of two, then the test contained in the following file should report NOT_APPLICABLE:

CXG2010.A
Annex E: Guidelines for Test Development

The guidelines used for developing recent ACATS tests are summarized in this Annex. Developers of potential ACATS tests should follow these guidelines closely. Tests that deviate extensively from these guidelines are far less likely to be added to the ACATS than those that follow them carefully.

- Tests should follow the test structure and organization of existing ACATS tests. Many details of existing tests are described elsewhere in this document. Important topics include:
  - Test classes (see 4.2);
  - Test naming conventions (see 4.3.2 and 4.3.3);
  - Test layout and prologue (see 4.4);
  - Identifier and reserved word conventions (see 4.5);
  - Library unit naming within tests (see 4.3.3);
  - Executable test structure (see 4.6); and
  - Indication of errors in B and L tests (see 4.6).

- Submitted tests should be include the standard ACAA "Grant of Unlimited Rights" (found in most newer tests issued since 2010) or an equivalent release allowing the test to be freely used by anyone. Under no circumstances should submitted tests contain proprietary information or code. Submitted tests without such a grant cannot be included in the ACATS or even the submitted tests directory.

- The test number (character positions 6 and 7) should be assigned letters rather than numbers by submitting test authors. Letters that are likely to be unique (such as the author's initials) are preferred. (Of course, different tests for the same clause with the same author should have different names.) The ACAA Technical Agent will perform the final naming of tests in order to ensure that the names are unique and appropriate. Following this guideline reduces the chance of test conflicts between authors.

- If a single test file contains multiple compilation units, they should be given in an order such that any dependent units follow the units they depend on (so an implementation can process the units sequentially in order). However, it should be assumed that all units in a file will be presented to the compiler simultaneously. If a test requires units to be presented in a specific order (as some separate compilation tests do), the units with ordering requirements should be in separate files, and the required order should be documented in the test prologue.

- When possible for B-Tests, only the last unit in a file should contain errors (or even better, units with errors should be in separate files from those without errors). This avoids penalizing implementations that process units in a file sequentially and stop on the first bad unit. (Multiple units in a single file are fairly rare in real user code; the ACATS shouldn't require work in areas not useful to typical users.) This guideline should be violated only when the number of units with errors would be prohibitive to have in separate files. (The maximum number of separate files in a test is 36 plus any foundations, because the naming conventions for tests only leave a single character for sequence numbers.)

- C-Tests (especially those testing rules that are not runtime checks) should be written in a usage-oriented style. That means that the tests should reflect the way the features are typically used in practice. Using a feature with no context is discouraged. For instance, C-Tests for limited with clauses should use them to declare mutually dependent types (the reason that limited with clauses were added to the language) rather than just using them to replace regular with clauses.

- Tests should avoid the use of Text_IO (unless required by the test objective). In particular, C-Tests should not create messages with Text_IO; all messages should be generated via the
subprograms in the Report package. Messages in C-Tests should be written in mixed case, not all UPPER CASE. Failure messages should be unique, so that the exact failure can be pinpointed. This is often accomplished by including a subtest identifier in the messages.

- Executable tests should not call routines in Report (directly, or indirectly via another package like TCTouch or Check_File) before the call to Report.Test or after the call to Report.Result. Doing so could cause an individual report (Comment, Failure, and so on) to not properly identify the test, or for the overall test result to be incorrectly reported. This does not apply to the various functions defined in Report with the exception of Legal_File_Name; these can be used freely in a test. It also does not apply to TCTouch.Touch.

- Tests can combine multiple objectives if a test for a single objective is very short. However, the objectives should be related, and the number of objectives in a test should be limited (to avoid creating gigantic tests that are hard to understand and use). In particular, objectives should not be combined if in doing so, the test will exceed 500 lines. Objectives from different clauses should never be combined, as that makes it hard to find the associated test (it will necessarily be filed in the wrong clause for one of the objectives).

- When possible, tests should define (and thus share) foundation code (see 4.1.4). Foundation packages are a better alternative than creating large tests with many objectives when the primary reason for combining the objectives is to avoid writing set-up code multiple times. Foundation code is specific to tests for a particular clause, however, so this technique cannot be used to combine objectives from multiple clauses.

- When a rule includes a term defined elsewhere, testing of the rule should include testing all of the combinations implied by the term. For instance, if we have a definition like "something is either this, that, or fuzzy", then a test for a test objective involving something should test cases where something is this and that and fuzzy. If multiple layers of definitions make this impractical, then a wide selection of combinations (as different as possible) should be tried. The only exception to this rule is if separate tests of the definition exist or should exist (that is, there is a test objective to test that the definition is appropriately implemented).

- When appropriate, tests should try a variety of things. For instance, when testing subprograms, both procedures and functions should be tested, with varying numbers of parameters, and with different modes and types. Similarly, types should be more than just Integer – tagged types, tasks, protected types, and anonymous access types should be tried. However, adding variety should not be used as an excuse to create multiple tests for an objective when one will do. That is, variety is a secondary goal; exhaustive coverage of possibilities isn't needed (unless the testing includes testing a term defined elsewhere, as described in the previous guideline). Remember that the goal isn't to test combinations of features; the point of using variety is to ensure that the objective being tested works in more than just the simplest cases.

- Tests should generally use only the 7-bit ASCII characters. However, some tests will need to use other characters in order to test Wide_Wide_Character support, Unicode characters in identifiers, and the like. Such tests should be encoded in UTF-8 and start with a UTF-8 Byte Order Mark. Tests should only use the code points that were assigned in version 4.0 of the Unicode standard; when possible, using only commonly used characters such as Greek and Cyrillic characters is preferred. (Such characters are more likely to be present in the fonts available to ACATS users; users will be more likely to be able to view the tests as intended if common characters are used.)

- When constructing tests that check to see that run-time checks are made, take special care that 11.6 permissions don't render the test impotent. 11.6(5) allows language-defined checks to be optimized away if the result of the operation is not used (even if the exception is handled). That means it is critical that the values that fail checks are used in some way afterwards (even though a correct program will never execute that code). Failure to do that could allow a compiler to optimize the entire test away, and that would require the test to be corrected later.
When creating a B-Test for which different parts test different errors, each error should identify the intended failure. The standard error indication includes a colon; any needed text can follow that colon. If the error identification will not fit on one line, place it somewhere else and index to it. One common way to do this is to put a list of intended errors into the header, labeling each with a letter. Then each error comment can just identify the letter of the intended error.

Tests that cover test objectives that are documented as untested are especially welcome as test submissions. Tests that cover previously tested objectives are less likely to be included in the test suite.

Submitted tests can be sent to the ACAA Technical Agent (agent@ada-auth.org) via e-mail; include a subject of "Submitted ACATS Test" in your e-mail. The agent will attempt to work with all submitters to correct critical deficiencies, and will attempt to publicly add tests to the submitted tests index in a timely fashion. However, there can be no promise that submitted tests (or a similar test for a similar objective) will ever be added to the ACATS.
Annex F: Definitions

ACATS Modification List. (Abbreviated AML) A list maintained by the ACAA documenting the currently modified and withdrawn tests. It also documents any new tests that have been or will be added to the test suite. The ACATS modification list is updated from time to time as challenges from implementers are received and processed, new tests are created, or as other technical information is received.

Acceptable result. The result of processing an ACATS test program that meets the explicit grading criteria for a grade of "passed" or inapplicable.

ACVC Implementer's Guide. (Abbreviated AIG) A document describing the test objectives used to produce test programs for [Ada83] ACVC versions (1.1-1.11). AIG section references are embedded in [Ada83] test naming conventions.

Ada. Short for Ada programming language. The term Ada by itself always refers to the most current ISO/IEC standard document(s); if a specific version of the language standards is meant it will always be referred to explicitly (for instance, as [Ada83] or [Ada95]).

Ada Conformity Assessment Authority. (Abbreviated ACAA) The part of the certification body that provides technical guidance for operations of the Ada certification system.

Ada Conformity Assessment Laboratory. (Abbreviated ACAL) The part of the certification body that carries out the procedures required to perform conformity assessment of an Ada implementation. (Formerly AVF)

Ada Conformity Assessment Test Report. (Abbreviated ACATR) A report summarizing the results of formal ACATS testing. Test Reports are issued only after witness testing is completed, and contain a summary of the testing (including which Specialized Needs Annexes were tested, any test modifications needed, and the values used in customizing the support files). Recent test reports can be found on-line at http://www.ada-auth.org/cpl.html, linked from the Certified Processors List.

Ada implementation. An Ada compilation system, including any required run-time support software, together with its host and target computer systems.

Ada Joint Program Office. (Abbreviated AJPO) An organization within the U.S. Department of Defense that sponsored the development of the ACVC and formerly provided policy and guidance for an Ada certification system.

Ada programming language. The language defined by the current Ada Standard documents.


Ada Standard documents. The document(s) that define the Ada programming language, currently the Ada Standard [Ada2012]. Future corrigendum documents are also included (corrigenda fix bugs in a Standard).

Ada Validation Facility. (Abbreviated AVF) Former designation of an Ada Conformity Assessment Laboratory.

Ada Validation Organization. (Abbreviated AVO) Organization that formerly performed the functions of the Ada Conformity Assessment Authority.
Certification Body. The organizations (ACAA and ACALs) collectively responsible for defining and implementing Ada conformity assessments, including production and maintenance of the ACATS tests, and award of Ada Conformity Assessment Certificates.

Certified Processors List. (Abbreviated CPL) A published list identifying all certified Ada implementations. The CPL is available on the ACAA Internet site (www.ada-auth.org).

Challenge. A documented disagreement with the test objective, test code, test grading criteria, or result of processing an ACATS test program when the result is not PASSED or INAPPLICABLE according to the established grading criteria. A challenge is submitted to the ACAA.

Conforming implementation. An implementation that produces an acceptable result for every applicable test. Any deviation constitutes a non-conformity.

Core language. Sections 2-13 and Annexes A, B, and J of [Ada95]. All implementations are required to implement the core language. The tests for core language features are required of all implementations.

Coverage documents. Documents containing an analysis of every paragraph of the Ada Standard documents. Each paragraph has an indication of whether it contains a testable Ada requirement, and if so, suggested test objectives to cover the requirements of the paragraphs. Paragraphs that include objectives also indicate what ACATS test(s) specifically test those objectives.

Deviation. Failure of an Ada implementation to produce an acceptable result when processing an ACATS test program.

Foundation Code. Code used by multiple tests; foundation code is designed to be reusable. Generally a foundation is a package containing types, variables, and subprograms that are applicable and useful to a series of related tests. Foundation code is never expected to cause compile time errors. It may be compiled once for all tests that use it or recompiled for each test that uses it; it must be bound with each test that uses it.

Legacy Tests. Tests that were included in ACVC 1.12 that have been incorporated into later ACVC and ACATS versions. The vast majority of these tests check for language features that are upwardly compatible from [Ada83] to later versions of Ada. Some of these tests have been modified from the ACVC 1.12 versions to ensure that Ada rules are properly implemented in cases where there were extensions or incompatibilities from [Ada83] to later versions of Ada.

Modern Tests. Tests that have been constructed and added to the ACATS since the release of ACVC 1.12. These tests usually test features added to Ada since [Ada83]. Modern tests have a coding style more like that used by typical programmers than the Legacy tests, and have a different naming convention.

Specialized Needs Annex. (Abbreviated SNA) One of annexes C through H of [Ada95]. Conformity testing against one or more Specialized Needs Annexes is optional. There are tests that apply to each of the Specialized Needs Annexes. Results of processing these tests (if processed during a conformity assessment) are reported on the certificate and in the Certified Processors List.

Test Objective. The intended purpose of an ACATS test. A test objective ought to be relatable to rules given in the Standards that define Ada.

Test Objectives Document. A document containing the test objectives used for Modern ACATS tests. Information on Legacy tests is not included.

Validated Compilers List. (Abbreviated VCL) Former designation of the Certified Processors List.

Validated Implementation. Informally used to mean Conforming Implementation.

Validation. Informally used to mean conformity assessment.
**Withdrawn Test.** A test found to be incorrect and not used in conformity testing. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains erroneous or illegal use of the Ada programming language. Withdrawn tests are not applicable to any implementation. Withdrawn tests are often modified and restored to subsequent ACATS releases.

**Witness Testing.** Conformity assessment testing performed in the presence of ACAL personnel. Witness testing adds the assurance that the test procedures were followed and that the results were verified.
References


Index

A
ACAA
See Ada Conformity Assessment Authority
ACAL
See Ada Conformity Assessment Laboratory
ACATR
See Ada Conformity Assessment Test Report
ACATS
See Ada Conformity Assessment Test Suite
ACATS Modification List 5.1.1, F
acceptable result F
ACVC
See Ada Compiler Validation Capability (ACVC)
ACVC Implementer's Guide 4.3.1, F
Ada 1, F
Ada Compiler Validation Capability 1
Ada Conformity Assessment Authority 1, F
Ada Conformity Assessment Laboratory 1, F
Ada Conformity Assessment Test Report 5.4.1, F
Ada Conformity Assessment Test Suite 5.4.1, F
Ada implementation 1, F
Ada Joint Program Office 1, F
Ada programming language 1, F
Ada Resource Association 1, F
Ada Standard documents 1.1, F
Ada Validation Facility 1, F
Ada Validation Organization 1, F
AIG
See ACVC Implementer's Guide
AJPO
See Ada Joint Program Office
AML 5.1.1
See also ACATS Modification List
ARA
See Ada Resource Association
AVF
See Ada Validation Facility
AVO
See Ada Validation Organization
B
batch file 5.4.1
C
certification body F
certified processors list F
deviation F
core language 5.5.1, F
coverage documents F
CPL
See certified processors list
coverage documents F
D
development of implementation 5.5.1, F
deviation F
D
deviation F
E
efficient implementation 5.5.4
foundation code 4.1.4, F
F
inapplicable test 5.6.4
I
inapplicable test 5.6.4
implementation-dependent values B
J
legacy tests 4.1.3, F
L
macro substitution B
M
modern tests 4.1.3, F
P
processing
foregn language tests 5.5.4
inapplicable tests 5.5.4
special handling tests 5.5.5
withdrawn tests 5.5.1
R
rule
Ada language 3.1
S
SNA
See specialized needs annex, specialized needs annex
special handling test 5.5.5
specialized needs annex F
support files 5.3.1
T
test
allowed modification AML
classification 5.2.5
inapplicable 5.6.4
modified AML classification 5.2.5
new AML classification 5.5.4
pending new AML classification 5.5.4
special handling 5.5.5
withdrawn 5.6.4
valid test objective 3, F
valid objectives document 3, F
V
validated compilers list F
validated implementation F
validation F
VCL
See validated compilers list
W
withdrawn test 5.6.5, F
witness testing F