THE ADA CONFORMITY ASSESSMENT TEST SUITE (ACATS) VERSION 4.1 USER'S GUIDE

June 29, 2016

Prepared by:
Ada Conformity Assessment Authority
Randall L. Brukardt, Technical Agent
621 N. Sherman Ave., Suite B6
Madison, WI 53704

Table of Contents

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 8652:2012 and later corrigenda, including ANSI/ISO/IEC 8652:2012/COR 1:2016). 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].

In 2014, the ACAA developed ACATS version 4.0 to test the enhancements and changes of the third edition of the Ada Standard, ISO/IEC 8652:2012 ([Ada2012]). This version of the ACATS, version 4.1, was released to improve the coverage and correctness of the ACATS for features defined in [Ada2012], including features originally defined in earlier versions of the Ada Standard. Subsequent maintenance or enhancement versions of the suite, if they are required, will be numbered 4.2, 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 4.1 summarizes the changes between ACATS 4.0 and ACATS 4.1. 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 4.1 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 4.1 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], [TC1-2012] 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 4.1 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 4.1

Version 4.1 of the ACATS updates version 4.0 with additional tests for features defined in [Ada2012]. It includes 134 new tests to check features including type invariants, subtype predicates, extended return statements, new iterator forms, array aggregates, generalized indexing, conditional expressions, aspect specifications, unchecked unions, Ada.Directories, Ada.Generic_Dispatching_Constructor, Ada.Environment_Variables, Ada.Text_IO.Bounded_IO and Unbounded_IO, and Ada.Containers.

In addition, 16 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.

Finally, ACATS 4.1 introduces a tool to simplify and mostly automate test grading (see 6, "ACATS Grading using the Grading Tool"). This tool and a supporting tool are provided in 6 Ada source files.

See Annex A, "Version Description" for lists of added, deleted and modified tests, documentation, and support files.

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 unfamilar 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 4.1, the following subclauses 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
6.1.1
6.5 through 6.8
7.3.2 through 7.6.1
8.3.1 through 8.5.3
Clause 10 (10 through 10.2.1)
13.1.1
B.3.3
```

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 value. 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 incorrecct) 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 4.1 is an update of ACATS 4.0, and has a similar delivery structure. The existing support tools are unchanged, except for updating header comments and version identification. There are some additional and changed files to support the new test grading tool (see 6).

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

4.1 Structure

The ACATS 4.1 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.

	Total	Core Tests	SNA Tests	Found-ations	Docs	Other
Number of Files	5002	4457	250	60	208	27
Number of Tests	4033	3841	192	0	0	0

Others consists of:

1	List of all files
13	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)
6	Code for test grading tools

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

4.1.1 Physical Organization

The preceeding table summarizes the number of files that compose ACATS 4.1. In addition to files containing test code proper, the ACATS 4.1 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., enumchek 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:

http://www.ada-auth.org/acats.html.

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

Clause 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 Clause 13.")

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

CD10001	CD30003	CD30008	CD72A02
CD20001	CD30004	CD30009	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 Clause 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, CXB3018, CXB3023, and CXB3024

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.

All of these test markings can be followed by an optional range indicator (see 6.3.2). These describe the expected locations of an error message for the error, and are primarily used by the automated Grading Tool (see 6).

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 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 4.1, 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.

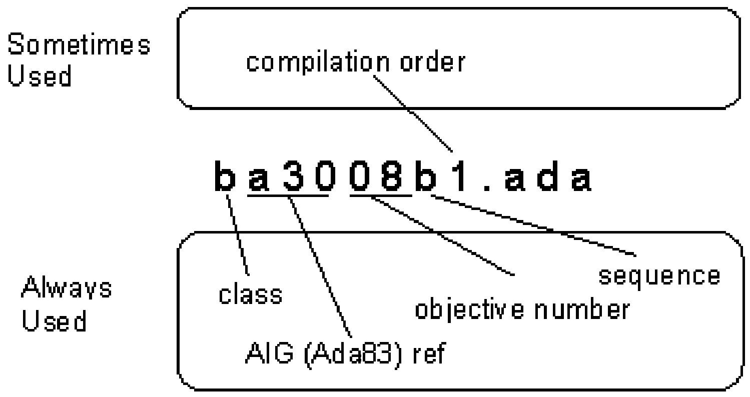
There are two different but similar naming conventions used in ACATS 4.1 Legacy tests use the naming conventions of early ACVC versions. Tests new since ACVC 1.12 use the modern convention. The conventions are consistently distinguishable at the 7th character of the name: legacy names have a letter in 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:

Position	**Kind**	**Meaning**
1	Letter	Test class (see Section 4.2)
2	Hexadecimal	AIG chapter containing the test objective
3	Hexadecimal	Section within the above AIG chapter
4	Alphanumeric	Sub-section of the above AIG section
5-6	Decimal	Number of the test objective within the above sub-section
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

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.

AD7001C0.ADA	B83E01E0.ADA	BA1010G0.ADA	BA1011C0.ADA
AD7001D0.ADA	B83E01F0.ADA	BA1010H0.ADA	BA1020A0.ADA
B38103C3.ADA	B86001A1.ADA	BA1010I0.ADA	BA1020B6.ADA
B38103E0.ADA	B95020B2.ADA	BA1010J0.ADA	BA1020C0.ADA
B63009C3.ADA	BA1001A0.ADA	BA1010K0.ADA	BA1020F2.ADA
B73004B0.ADA	BA1010A0.ADA	BA1010L0.ADA	BA1101B0.ADA
B83003B0.ADA	BA1010B0.ADA	BA1010M0.ADA	BA1101C2.ADA
B83004B0.ADA	BA1010C0.ADA	BA1010N0.ADA	BA1109A2.ADA
B83004C2.ADA	BA1010D0.ADA	BA1010P0.ADA	BA1110A1.ADA
B83004D0.ADA	BA1010E0.ADA	BA1010Q0.ADA	BA2001F0.ADA
B83024F0.ADA	BA1010F0.ADA	BA1011B0.ADA	BA2003B0.ADA

BA2011A1.ADA	C83F01C2.ADA	CA2002A0.ADA	LA5007B1.ADA
BA3001A0.ADA	C83F01D0.ADA	CA2003A0.ADA	LA5007C1.ADA
BA3001B0.ADA	C83F03C2.ADA	CA2004A0.ADA	LA5007D1.ADA
BA3001C0.ADA	C83F03D0.ADA	CA2007A0.ADA	LA5007E1.ADA
BA3001E0.ADA	C86004B2.ADA	CA2008A0.ADA	LA5007F1.ADA
BA3001F0.ADA	C86004C2.ADA	CA2009C0.ADA	LA5007G1.ADA
BA3006A6.ADA	CA1011A6.ADA	CA2009F0.ADA	LA5008A1.ADA
BA3006B4.ADA	CA1012A4.ADA	CA3011A4.ADA	LA5008B1.ADA
C38108C1.ADA	CA1012B4.ADA	CA5003A6.ADA	LA5008C1.ADA
C38108D0.ADA	CA1013A6.ADA	CA5003B5.ADA	LA5008D1.ADA
C39006C0.ADA	CA1014A0.ADA	CA5004B2.ADA	LA5008E1.ADA
C39006F3.ADA	CA1020E3.ADA	CC3019B2.ADA	LA5008F1.ADA
C64005D0.ADA	CA1022A6.ADA	CC3019C2.ADA	LA5008G1.ADA
C83022G0.ADA	CA1102A2.ADA	LA5001A7.ADA	
C83024E1.ADA	CA2001H3.ADA	LA5007A1.ADA	

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 4.1. 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

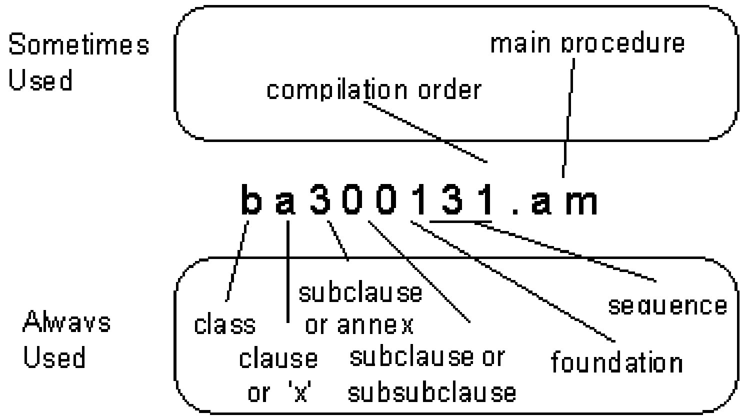
The name of a modern ACATS test is composed of seven or eight characters. Foundation code has a name composed of seven characters. The use of each character position is described below. The first column indicates the character position(s) starting from the left, and the second column indicates the kind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

Position	Kind	Meaning
1	Letter	Test class; foundations are marked 'F'.

2	Alphanumeric	If other than an 'x', the clause 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.
3	Alphanumeric	Core subclause or annex letter identifier (either core or Specialized Needs Annex); clauses are a hexadecimal value.
4	Alphanumeric	Sub-subclause (if a core test), or subclause (if an annex test); a number if less than 10, otherwise a letter with 10='A', 11='B', and so on.
5	Alphanumeric	Foundation identifier (alphabetic, unless no foundation is required, in which case a '0').
6-7	Decimal	Sequence number of this test in a series of tests for the same clause; foundation code will have "00".
8	Alphanumeric	*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.

(Note: Formally groupings for all levels below the top-level grouping are known as subclauses; here we use subclause to specifically refer to the second level and sub-subclause to refer to the third level.)

The convention is illustrated below.



Modern File Name Convention

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 4.1, 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, C**123A**xx depends on F**123A**00.

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 `C390006F3M`. 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:

```
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:

```
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.

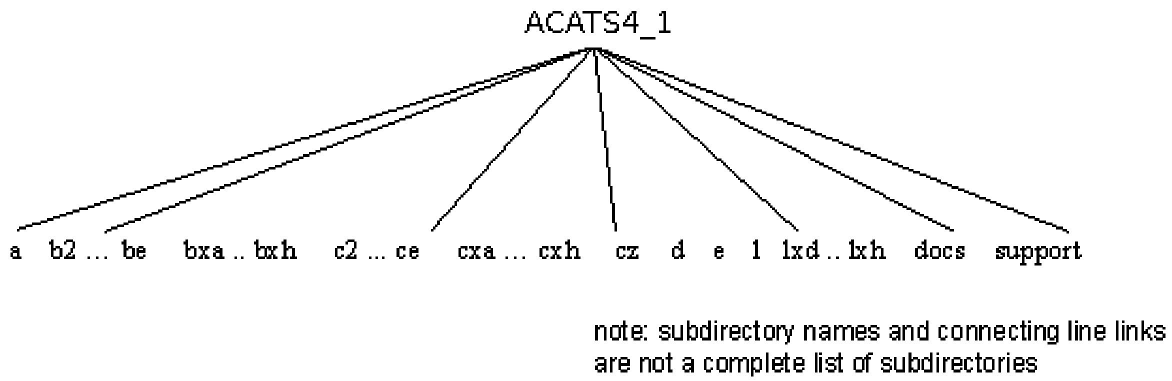
4.7 Delivery Directory Structure

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.


ACATS4_1
a b2 … be bxa .. bxh c2 … ce cxa … cxh cz d e l lxd .. lxh docs support
note: subdirectory names and connecting line links
are not a complete list of subdirectories

Delivery Directory Structure

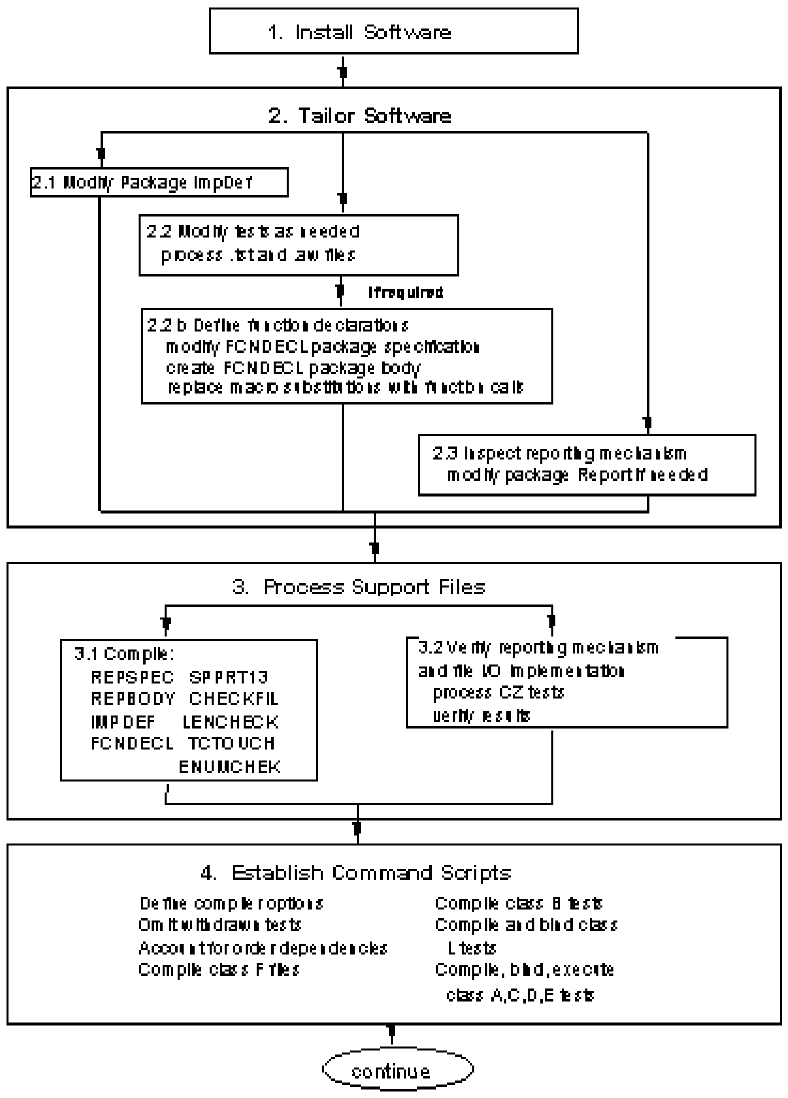
4.8 File Format

To conserve space and ease downloading, all files in the delivered ACATS 4.1 (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.

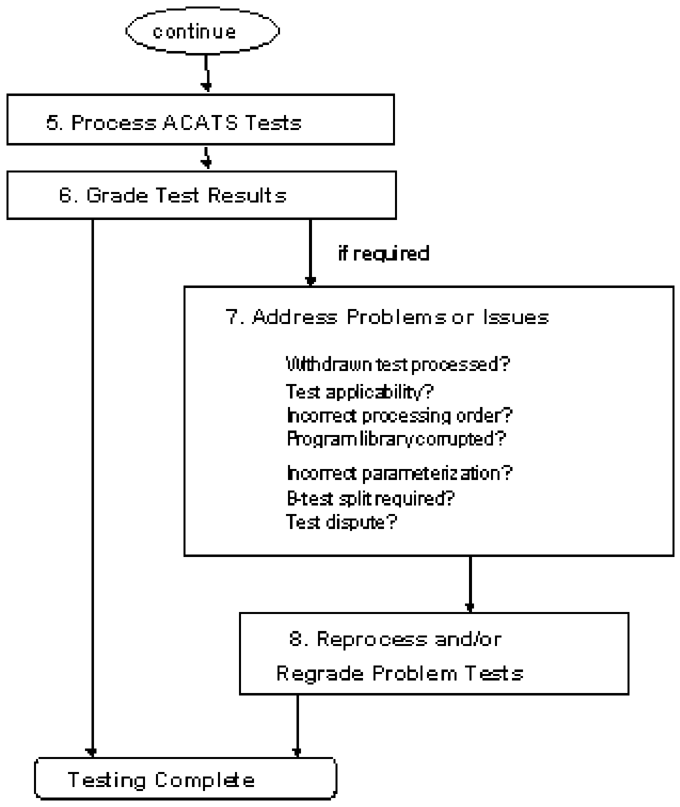
Files with the .zip extension have been compressed using a DOS zip utility; files with the .Z extension have been first put in Unix tar format and then compressed with Unix compress.

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.


1. Install Software
2. Tailor Software
2.1 Modify Package ImpDef
2.2 Modify tests as needed process .tst and .aw files
If required
2.2 b Define function declarations modify FCNDECL package specification create FCNDECL package body replace macro substitutions with function calls
2.3 Inspect reporting mechanism modify package Report if needed
3. Process Support Files
3.1 Compile: REPSPEC SPPRT13 REPBODY CHECKFIL IMPDEF LENCHECK FCNDECL TCTOUCH ENUMCHEK
3.2 Verify reporting mechanism and file I/O implementation process CZ tests verify results
4. Establish Command Scripts
Define compiler options
Omit withdrawn tests
Account for order dependencies
Compile class F files
Compile class B tests
Compile and bind class L tests
Compile, bind, execute class A,C,D,E tests
continue

Using the ACATS


continue
5. Process ACATS Tests
6. Grade Test Results
if required
7. Address Problems or Issues
Withdrawn test processed?
Test applicability?
Incorrect processing order?
Program library corrupted?
Incorrect parameterization?
B-test split required?
Test dispute?
8. Reprocess and/or Regrade Problem Tests
Testing Complete

Using the ACATS (cont.)

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_1x, where 'x' represents the suffix letter for the AML version.

These files can be found on the ACAA's web site:

www.ada-auth.org/acats.html.

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_1/a        ./acats4_1/be       ./acats4_1/cd       ./acats4_1/cxc
./acats4_1/b2       ./acats4_1/bxa      ./acats4_1/ce       ./acats4_1/cxd
./acats4_1/b3       ./acats4_1/c2       ./acats4_1/cxa      ./acats4_1/cxe
./acats4_1/b4       ./acats4_1/c3       ./acats4_1/cxb      ./acats4_1/cxf
./acats4_1/b5       ./acats4_1/c4       ./acats4_1/cz       ./acats4_1/cxg
./acats4_1/b6       ./acats4_1/c5       ./acats4_1/d        ./acats4_1/cxh
./acats4_1/b7       ./acats4_1/c6       ./acats4_1/e        ./acats4_1/lxd
./acats4_1/b8       ./acats4_1/c7       ./acats4_1/l        ./acats4_1/lxe
./acats4_1/b9       ./acats4_1/c8       ./acats4_1/bxc      ./acats4_1/lxh
./acats4_1/ba       ./acats4_1/c9       ./acats4_1/bxd      ./acats4_1/docs
./acats4_1/bb       ./acats4_1/ca       ./acats4_1/bxe      ./acats4_1/support
./acats4_1/bc       ./acats4_1/cb       ./acats4_1/bxf
./acats4_1/bd       ./acats4_1/cc       ./acats4_1/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 4.1 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_1`, 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_1. For `winzip`, this simply means selecting acats4_1 as the extract path.

For example, using unzip, and assuming that the archive name is ACATS41.zip, type

```
cd acats4_1
```

to set the proper directory, and

```
unzip ACATS41
```

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_41.tar.Z from the distribution source to a hard drive. Uncompress the file with the Unix command

```
uncompress acats_41.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_1 directory to be created and then untar the ACATS files

```
tar -xvf <path>/acats_41.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:

Common Name	Ada Name	Decimal Value	Hex Value
Carriage return	ASCII.CR	13	0D
Carriage return	ASCII.CR	13	0D
Vertical tab	ASCII.VT	11	0B
Line feed	ASCII.LF	10	0A
Line feed	ASCII.LF	10	0A
Form feed	ASCII.FF	12	0C

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:

Common Name	Ada Name	Decimal Value	Hex Value
Horizontal tab	ASCII.HT	9	09
Vertical tab	ASCII.VT	11	0B
Carriage return	ASCII.CR	13	0D
Line feed	ASCII.LF	10	0A
Form feed	ASCII.FF	12	0C

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 4.1 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 (fcndecl.ada) must be provided if necessary. Finally, Package Report (report.a) 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 4.1, 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 4.1, there are seven new parameters in Impdef. Other than those parameters, and the removal of two obsolete, unused parameters, there was no change to Impdef or any of its children from ACATS 4.0 to ACATS 4.1. A version of any of these packages that was tailored for ACATS 4.0 should be valid for ACATS 4.1 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 4.0 to ACATS 4.1. A version of `macro.dfs` that was tailored for ACATS 4.0 should be valid for ACATS 4.1 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 Clause 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 **with**s it and **use**s 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 ACAA) 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 package Report is in the file `report.a`; this includes both the specification and body.

The specification of package Report may be modified to change the setting of Generate_Event_Trace_File. (This setting controls whether Report writes an event trace file for the grading tool, see 6.2 for more information.) No other modifications of the specification of package Report are allowed.

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 the body of Report must receive advance approval from the ACAL (and, if necessary, the ACAA) before use in a conformity assessment. No approval is needed to change the setting of Generate_Event_Trace_File, but the setting used should be reported to the ACAL performing 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:

`report.a`	
`impdef.a`	`impdefc.a` *(if testing Annex C)*
`fcndecl.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 Comformity 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 **with**s 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.

c3a1003	ca11023	cc51010
c3a1004	ca12001	

5.5.5.2 Foreign Language Interface Tests

Annex B, Interface to Other Languages, is part of the Ada core language. However, most of the features contained in it are optional. In particular, if an implementation does not support an optional feature in Annex B, 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 B(2/3).)

ACATS tests expect that an implementation that provides one or more of the packages Interfaces.C, Interfaces.COBOL, or Interfaces.Fortran implements the entire interface as defined by Annex B. As such, ACATS testing procedures expect that the implementation will correctly process, and pass, all of the tests for interfaces to C, COBOL, and/or Fortran code respectively, with the possible exception of tests containing actual foreign code. If an implementation provides a partial implementation of one of the foreign language interfaces, special handling of the test results will be needed. In particular, some or all tests may fail by reporting an error on a line other than on marked with N/A => ERROR or may fail at runtime. Such cases need to be described to the ACAL and ACAA and will be handled on a case-by-case basis.

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 (see D.1.15 (C), D.1.18 (COBOL), and D.1.19 (Fortran) for lists of tests that use the packages). Similarly, if a convention identifier for one of these languages is not supported, then the corresponding tests may be graded as inapplicable, provided they reject the corresponding aspect or **pragma** (see D.1.12 (C), D.1.13 (COBOL), and D.1.14 (Fortran) for lists of tests containing each convention).

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

The following tests check the C language interface; the ACATS expects that all of the tests identified below will be satisfactorily processed as described above if the C language interface is supported.

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.

cd30005*	cxb3003	cxb3010	cxb3017*
bxb3001	cxb3004*	cxb3011	cxb3018
bxb3002	cxb3005	cxb3012	cxb3019
bxb3003	cxb3006*	cxb3013*	cxb3020
bxb3004	cxb3007	cxb3014	cxb3021
cxb3001	cxb3008	cxb3015	cxb3022
cxb3002	cxb3009	cxb3016	

COBOL Language Interface

The following tests check the COBOL language interface; the ACATS expects that all of the tests identified below will be satisfactorily processed as described above if the COBOL language interface is supported.

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.

cxb4001	cxb4004	cxb4007
cxb4002	cxb4005	cxb4008
cxb4003	cxb4006	cxb4009*

Fortran Language Interface

The following tests check the Fortran language interface; the ACATS expects that all of the tests identified below will be satisfactorily processed as described above if the Fortran language interface is supported.

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.

cxb5001	cxb5002	cxb5003	cxb5004*

cxb5005*

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:

bxe2009	bxe4001	cxe4002	cxe4006
bxe2010	cxe2001	cxe4003	cxe5002
bxe2011	cxe2002	cxe4004	cxe5003
bxe2013	cxe4001	cxe4005	lxe3001

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	cxe4004	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	cxg2009	cxg2014	cxg2019
cxg2004	cxg2010	cxg2015	cxg2020
cxg2006	cxg2011	cxg2016	cxg2021
cxg2007	cxg2012	cxg2017	
cxg2008	cxg2013	cxg2018	

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	cxd2001	cxd4008	lxd7008
bxc5001	cxd2002	cxd4009	lxd7009
bxh4001	cxd2003	cxd4010	lxh4001
bxh4002	cxd2004	cxd5002	lxh4002
bxh4003	cxd2005	cxd6002	lxh4003
bxh4004	cxd2006	cxd6003	lxh4004
bxh4005	cxd2007	cxda003	lxh4005
bxh4006	cxd2008	cxdb005	lxh4006
bxh4007	cxd3001	cxh1001	lxh4007
bxh4008	cxd3002	cxh3001	lxh4008
bxh4009	cxd3003	cxh3003	lxh4009
bxh4010	cxd4001	lxd7001	lxh4010
bxh4011	cxd4003	lxd7003	lxh4011
bxh4012	cxd4004	lxd7004	lxh4012
bxh4013	cxd4005	lxd7005	lxh4013
cxd1004	cxd4006	lxd7006	
cxd1005	cxd4007	lxd7007	

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 4.1 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.

The ACATS provides a tool that can be used to automate much of the test grading process; see 6. This tool enforces the expected results as described in the following sections, as well as checking for processing errors (compiling test files in the wrong order, for instance) and omissions (failing to compile required test files).

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. The actual text of error messages is not used in determining whether an error is properly detected; only the location of the reported error is used. (The ACATS must not prevent innovation in error handling.)

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.

Section 6: ACATS Grading using the Grading Tool

ACATS 4.1 introduces an optional tool to (mostly) automate grading of ACATS tests.

When the ACATS was designed (as the ACVC in the early 1980s), the intention always was that running it would give a simple and clear Pass or Fail result. However, grading of tests (particularly of B and L Tests) is somewhat subjective and very time-consuming. (Test grading by formal testers typically involves poring over compiler listings of the entire ACATS with a large highlighter.)

The grading tool greatly reduces this effort and enforces both the processing rules (as outlined in 5.4.2) and expected results for a test (see 5.6).

Use of the grading tool is optional for ACATS 4.1. Manual test grading is acceptable for formal conformity assessments; whether to use the grading tool will be left to implementers and their ACAL. The ACAA will use experience with the tool to inform whether using the tool should be required for future ACATS versions.

6.1 Using the Grading Tool

The ACATS Grading Tool, "Grade" takes three files as input and produces a grading report, with details for each test followed by a summary and an overall Passed or Failed result.

The three files are:

Event Trace File
: Includes each interesting event that occurs during the processing of an ACATS test. Each ACATS User (or their implementor) needs to provide a method of producing an event trace file for their implementation in order to use the Grading Tools. See clause 6.2 for more information.

Test Summary File
: A machine-readable distillation of one or more ACATS tests. These are created by a tool included with the ACATS. See clause 6.3 for more information.

Manual Grading File
: A list of tests that may require manual grading. This file can be created with any plain text editor, and may be empty. See clause 6.4 for more information.

An example using the grading tool is given in subclause 6.1.2, “Annotated Grading Tool Example”.

6.1.1 Workflow using the Grading Tool

The workflow using the Grading Tool is similar to using the ACATS manually. The steps needed are outlined below.

1. Install and configure the ACATS in the normal way, as outlined in clauses 5.1, 5.2, and 5.3. It is particular important that Macro Defs customization 5.2.2 is accomplished *before* generating any test summaries, as the summary program is unaware of the macro syntax. Also, do not use the grading tool on the support tests in the CZ directory, as some of these include intentional failure messages that the grading tool is not prepared to handle.
2. Compile the Grading and Test Summary Tools, as described in 6.1.3.
3. Determine how Event Traces are going to be constructed. If the implementation provides direct writing of an event trace (as described in 6.2.2), then go to step 4a. Otherwise, acquire or create a listing convertion tool as described in 6.2.3, and go to step 4b.

4a. Create command scripts (as described in 5.4) to process the ACATS. Include in those the appropriate option to create event traces. Also, modify Report.A so that the constant Generate_Event_Trace_File has the value True. When complete, go to step 5.

    The command scripts could generate one giant event trace for the entire ACATS, but it probably is more manageable to create several smaller event traces for portions of the ACATS. One obvious way to do that is to create a single event trace for each subdirectory that contains ACATS tests in its default delivery structure. (Such event traces can be combined later, if desired.)

4b. Create command scripts (as described in 5.4) to process the ACATS. Include in those use of the listing conversion tool to make event traces. Then go to step 5.

5. Create test summaries for each grading segment. If, for instance, you will grade each directory individually, then you will need a test summary file for each directory. These files can be generated by running the Test Summary tool on each source file in the directory using a single summary file as output. On most operating systems, this is easily accomplished with a script command. (Some possibilities are discussed in 6.1.5).

   The Test Summary Files will only need to be regenerated if the ACATS tests change in some way (typically, when an ACATS Modification List is issued). It's probably easiest to make a script to regenerate the entire set of summaries so that it can be used when the suite changes. Once the entire set of test summaries has been created, move to step 6.

6. Create an empty manual grading request file. This is just an empty text file. (See 6.4 for more information.)

7. Process the ACATS tests, creating event traces. The event traces should contain the same tests as the test summary files. This process is described in 5.5.

8. Run the Grading Tool (GRADE) on the pairs of event traces and summaries, using the current manual grading request file. Typically, the default options are sufficient, but some implementations or event traces may need options. The options are described in 6.1.4.

9. If all of the grading reports display Passed, you're done. But most likely, some tests will be reported as failed. The grading tool will report the first failure reason for each test, but there may be additional failure reasons for each test.

   A. If the failure reason is a process failure or a missing compilation, most likely there is a problem with the scripts that process the ACATS. Make sure that the test files are compiled in the appropriate order and no test files are missing. A missing compilation might also mean that the test needs to be split. See item B.

   B. If the failure reason is extra or missing errors, grade the test manually (see 5.6) to see if the problem is with the implementation or the Grading Tool being too strict about error locations. If manual grading indicates the test passed, add the test to your Manual Grading Request file - again, see 6.4 (preferably with a comment explaining why it was added). Note that it is not necessary to remove tests from this list: if the grading tool determines that the test grades as Passed or Not Applicable, it will not request manual grading for the test even if it appears in this list.

      If the manual grading indicates that the test needs to be split, do the following. First, add the test to your Manual Grading Request file - the ACATS requires processing the original test in this case. (Be sure to put in a comment that the test is split, since it won't be necessary to manually grade the original test in that case.) Then, split the test following the guidelines in 5.2.5, and add the split tests to a processing script and the test summary script. The Test Summary tool can create summaries for split tests, so the grading tool can be used to grade them.

   C. For other failure reasons, most likely the implementation is at fault. Fixing the implementation is likely the only way to meaningfully change the result. In the unlikely event that there is a problem with a test, the procedure for test challenges are outlined in 5.7.2.

You'll also need to handle any special handling tests, including any tests that require manual grading. (This is one good reason to keep the manual grading list as short as possible.)

Then return to step 7 and repeat the test run.

Using this procedure, the vast majority of tests will not require hand grading. Future ACATS updates may improve tests that are particulary difficult to grade automatically. The ACAA is interested in which tests need manual grading for your implementation - see 6.4.

6.1.2 Annotated Grading Tool Example

Following is an example of using the grading tool on Chapter 5 C-Tests for an implementation that has not yet implemented most of Ada 2012. For this example, file ChapC5.csv is an event trace containing the events generated by processing the 102 ACATS C-Tests for chapter 5. The file C5-Sum.csv is the test summary file for the 102 ACATS C-Tests for chapter 5. Man.Txt is an empty file (there are no manual grading requests for this run).

In the following, *annotations are given in italics*, and are not part of the output of the Grading Tool.

The command line used for the example is:

```
Grade ChapC5.csv C5-Sum.csv Man.Txt "C-Tests for Chapter 5" - Use_Timestamps -
Check_All_Compiles -No_Positions
```

This gives the following expected results:

```
ACATS Grading Tool - version 1.0
  Compile Checks: CHECK_ALL
  Make checks of time stamps in event log
  Use only line numbers of error messages when grading
  Report verbosity: NORMAL
  779 event trace records read from ChapC5.csv
```

The non-comment records in the event trace file.

```
  117 test summary trace records read from C5-Sum.csv
```

The non-comment records in the test summary file. All tests in this file are graded, whether or not they appear in the event trace. Extra tests in the event trace are ignored.

```
  0 manual grading requests read from New-Man.Txt
```

The number of tests for which manual grading was requested, excluding comments.

```
-- Test C51004A passed execution
```

The result for an individual test. The passed results can be suppressed with the -quiet option rather than -normal.

```
-- Test C52005A passed execution
-- Test C52005B passed execution
-- Test C52005C passed execution
-- Test C52005D passed execution
-- Test C52005E passed execution
-- Test C52005F passed execution
-- Test C52008A passed execution
-- Test C52008B passed execution
-- Test C52009A passed execution
-- Test C52009B passed execution
-- Test C52010A passed execution
-- Test C52011A passed execution
-- Test C52011B passed execution
-- Test C52101A passed execution
-- Test C52102A passed execution
-- Test C52102B passed execution
-- Test C52102C passed execution
-- Test C52102D passed execution
-- Test C52103A passed execution
-- Test C52103B passed execution
-- Test C52103C passed execution
-- Test C52103F passed execution
-- Test C52103G passed execution
-- Test C52103H passed execution
-- Test C52103K passed execution
-- Test C52103L passed execution
-- Test C52103M passed execution
-- Test C52103P passed execution
-- Test C52103Q passed execution
-- Test C52103R passed execution
-- Test C52103X passed execution
-- Test C52104A passed execution
-- Test C52104B passed execution
-- Test C52104C passed execution
-- Test C52104F passed execution
-- Test C52104G passed execution
-- Test C52104H passed execution
-- Test C52104K passed execution
-- Test C52104L passed execution
-- Test C52104M passed execution
-- Test C52104P passed execution
-- Test C52104Q passed execution
-- Test C52104R passed execution
-- Test C52104X passed execution
-- Test C52104Y passed execution
-- Test C53007A passed execution
** Test C540001 has unexpected compile error at line 144 in file C540001.A
   because: Feature not implemented
```

The test uses an Ada 2012 feature not implemented in the test implementation. Thus, this test (and several others) fail. The message is not part of the grading (and is suppressed if -quiet is used), but should be helpful in determining the reason a test has failed.

```
** Test C540002 has unexpected compile error at line 112 in file C540002.A
   because: Unable to resolve expression
** Test C540003 has unexpected compile error at line 69 in file C540003.A
```

```
   because: Feature not implemented
-- Test C54A03A passed execution
-- Test C54A04A passed execution
-- Test C54A07A passed execution
-- Test C54A13A passed execution
-- Test C54A13B passed execution
-- Test C54A13C passed execution
-- Test C54A13D passed execution
-- Test C54A22A passed execution
-- Test C54A23A passed execution
-- Test C54A24A passed execution
-- Test C54A24B passed execution
-- Test C54A42A passed execution
-- Test C54A42B passed execution
-- Test C54A42C passed execution
-- Test C54A42D passed execution
-- Test C54A42E passed execution
-- Test C54A42F passed execution
-- Test C54A42G passed execution
** Test C550001 has unexpected compile error at line 72 in file C550001.A
   because: Feature not implemented
** Test C552001 has unexpected compile error at line 150 in file C552001.A
   because: This must name a type
++ Test C552002 N/A because of expected error at line 59 in file C552002.A
   because: Feature not implemented
```

Note that the text of error messages does not have any effect on test grading. This test is Not Applicable because the first error occurred on a N/A => Error line, regardless of the message text.

```
** Test C552A01 has unexpected compile error at line 73 in file C552A01.A
   because: F552A00_PRIME_NUMBERS; WITHed compilation unit not found
```

Here, a foundation (which is not graded by itself) failed to compile. The failure to compile the foundation (because of an unimplemented feature) caused this test to fail.

```
** Test C552A02 has unexpected compile error at line 94 in file C552A02.A
   because: F552A00_SPARSE_ARRAYS; WITHed compilation unit not found
-- Test C55B03A passed execution
-- Test C55B04A passed execution
-- Test C55B05A passed execution
-- Test C55B06A passed execution
-- Test C55B06B passed execution
-- Test C55B07A passed execution
++ Test C55B07B N/A because of expected error at line 45 in file C55B07B.DEP
   because: Identifier is not defined
```

This test is Not Applicable because the implementation in question does not define a type Short_Integer.

```
-- Test C55B10A passed execution
-- Test C55B11A passed execution
-- Test C55B11B passed execution
-- Test C55B15A passed execution
-- Test C55B16A passed execution
-- Test C55C02A passed execution
-- Test C55C02B passed execution
-- Test C56002A passed execution
-- Test C57003A passed execution
-- Test C57004A passed execution
-- Test C57004B passed execution
-- Test C58004C passed execution
-- Test C58004D passed execution
-- Test C58004G passed execution
-- Test C58005A passed execution
-- Test C58005B passed execution
-- Test C58005H passed execution
-- Test C58006A passed execution
-- Test C58006B passed execution
-- Test C59002A passed execution
-- Test C59002B passed execution
-- Test C59002C passed execution

=======================================

Summary for C-Tests for Chapter 5
```

```
Result                        Overall  B-Tests  C-Tests  L-Tests  Other Tests

   Total Tests Graded           102        0      102        0        0
```
The total tests graded. The numbers in each column reflect the number of tests of the appropriate type for the category in question.

```
** Failed (Process)               0        0        0        0        0
```
A process failure is some problem with the test processing, such as running a test before it is compiled or compiling units out of the required order. Usually, correcting the test processing is all that is needed to eliminate this error.
```
** Failed (Compile Missing)       0        0        0        0        0
```
This means that some required unit was not compiled. This is a failure even if the test executed and passed.
```
** Failed (Compiler Crash)        0        0        0        0        0
```
This means that some compilation started but did not finish normally. This usually reflects some internal compiler error.
```
** Failed (Error in OK area)      0        0        0        0        0
```
This means that an error was reported in the range of an OK tagged line. These are counted separately as these typically indicate a significant compiler bug.
```
** Failed (Unexpected Error)      7        0        7        0        0
```
This means that an error was reported outside of the range of any test tag.
```
** Failed (Missing Error)         0        0        0        0        0
```
This means that no error was reported where one was required (for instance, for an error tag). This is only likely for a B-Test.
```
** Failed (Bind Missing)          0        0        0        0        0
```
This means that no bind operation was found for the test (and no other failure occurred first).
```
** Failed (Bind Crash)            0        0        0        0        0
```
This means that bind started but did not finish normally.
```
** Failed (Bind Error)            0        0        0        0        0
```
This means that bind reported an error.
```
** Failed (Run Missing)           0        0        0        0        0
```
This means that no test execution was found (and no other error was detected first).
```
** Failed (Run Crash)             0        0        0        0        0
```
This means that test execution started but did not complete and report a result.
```
** Failed (Runtime Message)       0        0        0        0        0
```
This means that test execution explicitly reported Failed.
```
++ Not-Applicable (Annex C)       0        0        0        0        0
```
This means that the test did not meet an Annex C Requirement; if Annex C is being tested, this should be considered a failure.
```
++ Not-Applicable (Compile)       2        0        2        0        0
```
This means that the test had an error in an N/A => Error range.
```
++ Not-Applicable (Runtime)       0        0        0        0        0
```
This means that the test execution reported that it was Not Applicable.
```
!! Special Handling (Runtime)     0        0        0        0        0
```
This means that the test execution reported that it required special handling.
```
!! Manual Grading Requested       0        0        0        0        0
```
This means that the test failed for some reason and manual test grading was requested. If the test grades as Passed or Not Applicable, the manual grading request is ignored and the test will be counted in the appropriate other category.
```
== Passed (Expected Errors)       0        0        0        0        0
```
This means that the test had compile-time errors as expected, and no extra errors.
```
== Passed (Bind Error)            0        0        0        0        0
```
This means that the test had bind errors as expected (this is most likely for an L-Test.
```
== Passed (Runtime)              93        0       93        0        0
```
This means that the test reported Passed when executed.
```
=======================================

Overall result for C-Tests for Chapter 5 is **FAILED**
```
The overall result is Failed unless all tests either pass, are not applicable, or had manual grading requested. For this implementation, the result of failed is expected. The implementer could have requested manual grading for the tests that were not expected to work, in order to change this result to Passed.

6.1.3 Compiling the Grading Tool and the Test Summary Tool

The Grading Tool and the Test Summary Tool are provided in six Ada source files. Some of the files are shared by both tools, so we present them as a single group. The six files are:

GRADE.A
: The main subprogram for the grading tool; contains option processing and test grading.

GRD_DATA.A
: A package to store grading data; contains the code to read and display all of the file information (events, test summaries, and manual grading requests).

SPECIAL.A
: A package containing the special handling for the test summary program. This package contains the data needed to handle optional units and tests with severe syntax errors that cannot be processed by normal means.

SUMMARY.A
: The main subprogram for the test summary tool; contains lexical and syntactic analysis for ACATS test files.

TRACE.A
: A package containing the data types that define an event trace.

TST_SUM.A
: A package containing the data types that define a test summary; it also contains a routine to write an individual test summary record.

The source of the grading tools is written in Ada, and only uses Ada 95 features other than the following:

- Raise statements with messages;
- Ada.Containers.Generic_Array_Sort;
- Ada.Calendar.Formatting.

Uses of the first could be replaced by calls to Ada.Exceptions.Raise_Exception; and the latter could be replaced by similar routines. We did not do this for readability and to avoid re-inventing the wheel.

The source code should be compilable by any Ada 95 or later compiler that supports the above three features; it certainly should be compilable by any Ada 2012 compiler.

As each Ada implementation uses different commands for compiling, we can only give the general direction that the six source files need to be compiled and then bound into two execuable programs: Grade (the Grading Tool) and Summary (the Test Summary Tool). Elsewhere in this documentation we assume that this has been done.

6.1.4 Grading Tool Reference

The command line for the Grading Tool is:

```
Grade <Event_Trace_File_Name> <Summary_of_Tests_File_Name>
<Manual_Grading_Request_Name> <Quoted Report Title> [options]
```

<Event_Trace_File_Name>
: The name of an event trace file (see 6.2). This can use any file name acceptable to the implementation that compiled the tool (in particular, full paths may be used on most implementations). The event trace should contain traces of the processing of at least the tests to be graded; it is acceptable to include traces for additional tests that are not being graded.

<Summary_of_Tests_File_Name>
: The name of a test summary file (see 6.3). This can use any file name acceptable to the implementation that compiled the tool (in particular, full paths may be used on most implementations). The test summary file should contain the summaries of exactly the tests that need to be graded; all of the tests summarized in the file will be graded regardless of the contents of the event trace. If the event trace does not include a test that is in the summary file, the test will be graded "Failed - Compile Missing".

<Manual_Grading_Request_Name>
: The name of a manual grading request file (see 6.4). This can use any file name acceptable to the implementation that compiled the tool (in particular, full paths may be used on most implementations). This file may include names of tests not included in the test summary file; such names will have no effect on grading.

<Quoted Report Title>
: A double quoted string containing the name of the test report. This is primarily intended so that similar-looking reports can be differentiated.

[options]
: Zero or more optional setting flags for the Grading Tool. These are case-insensitive, and can be:

-Specs_Optional
: Compiling of specifications is optional. Use for source-based compilers (such as the commonly used GNAT compiler) that don't compile specifications in normal operation. Compilation of bodies, instances, and so on are checked.

-Check_All_Compiles
: All compilations are checked and must be present (unless processing the unit is marked as optional). This is the default.

-No_Compile_Checks
: No compilation checks are made. This option is not allowed for formal conformity assessments.

-Use_Time_Stamps
: Check event trace time stamp information as part of checking, specifically, enforce that all units of a test are compiled and run in an appropriate order and reasonably close in time. This is the default.

-No_Time_Stamps
: Do not make any check of event trace time stamps. Use this option only if there is no meaningful timestamps in the event trace. This option is not allowed for formal conformity assessments.

-Use_Positions
: Use position information when checking whether errors are appropriately detected. This is the default.

-No_Positions
: Use only line information when checking whether errors are appropriately detected. Use this option if the event trace doesn't have position information. It is acceptable to use this option for formal conformity assessments.

-Quiet Produce minimal information: a list of failed tests and the summary report.

-Verbose
: Produce information about every test processed (including passed tests), along with the summary report. In this mode, the grading tool also produces a warning for multiple messages for one error tag.

-Normal
: Produce details about each failed test, along with the summary report. This is the default.

Only one of the options -Specs_Optional, -Check_All_Compiles, or -No_Compile_Checks can be given. Only one of the options -Use_Positions or -No_Positions can be given. Only one of the options -Quiet, -Verbose, or -Normal can be given. Only one of the options -Use_Time_Stamps or -No_Time_Stamps can be given.

An annotated example using the grading tool is given in subclause 6.1.2, “Annotated Grading Tool Example”. That example explains the output of the grading tool.

6.1.5 Test Summary Tool Reference

The command line for the Test Summary Tool is:

```
Summary <ACATS_Test_File_Name> <Summary_of_Tests_File_Name>
```

<ACATS_Test_File_Name>

The name of an ACATS test source file. This can use any file name acceptable to the implementation that compiled the tool (in particular, full paths may be used on most implementations). The Summary tool assumes that the simple file name of the test follows the naming conventions as described in 4.3; either the modern or legacy naming conventions are acceptable.

The summary tool can be used on split tests and other Ada code not directly part of the ACATS, so long as the ACATS naming conventions are followed (see above), and the files contain no optional units (as the tool has no way to discover optional units). This allows the Grading Tool to be used on tests being developed for submission to the ACATS (see Annex E). We also expect that the tool could be used on older ACATS versions, allowing the Grading Tool to be used with those versions.

<Summary_of_Tests_File_Name>

The name of the Test Summary File. If this file exists, the new test summary records will be appended to it. Otherwise, the file will be created.

The test summary tool needs to be run on each individual source file. Typically, it makes sense to combine all of the tests of a single ACATS directory into a single test summary file. This can easily be accomplished on Microsoft Windows with a batch file containing the following:

```
Rem Chapter 5 C-Tests
Del C5-Sum.csv
for %%i in (\My_ACATS\C5\*.A??) do Summary %%i C5-Sum.csv
for %%i in (\My_ACATS\C5\*.D??) do Summary %%i C5-Sum.csv
```

We first delete any existing summary file (so we don't accidentally double the contents), then run the summary tool on all of the ACATS source code found in the directory \My_ACATS\C5. (The second loop is needed in case there are any legacy .DEP files; all of the other extensions start with 'A'.)

A similar technique can be used on other host operating systems.

6.2 Event Trace Files

An event trace file includes each (interesting) event that occurs during the compilation, binding/linking, and execution of one or more ACATS tests.

An event trace file provides a way to present the implementation-specific format of events to the Grading Tool in a common format. In order for an ACATS user to use the Grading Tool, they will need to provide a method to get an event trace file from the implementation's processing of ACATS tests. There are a number of ways to accomplish that; several are outlined in following subclauses. No matter what method is selected, it should be possible to use the same options/tools for future ACATS tests. (Developing a method to create event trace files should be a one-time cost).

The events of an event trace file are intended to be abstract representations of the processes of an implementation. It should be possible to map the processes of any Ada implementation into an event trace file.

The event trace file was selected as the method of abstracting implementation processing in order to avoid the ACATS Grading Tool from providing an disincentive to innovation in error handling by Ada implementations. The files are not intended to be useful (directly) to a human user, so their details should have little effect on the human error handling interface for an implementation. Moreover, while the event trace files provide values for error messages and positioning, neither of these is required for formal grading (they're provided to making easier for an ACATS user to figure out why a test is reported as failing). As noted in 5.6.2, the actual text of an error message is not used to determine pass or fail for grading purposing; only the specified location of the error is used.

6.2.1 Event Trace File Reference

An event trace file is a CSV (Comma Separated Value) file of event records. See 6.5, “CSV File Reference” for the general rules for constructing a CSV file.

An event trace file record contains the following comma-separated fields on a single line:

Event	One of UNKN (Unknown), CSTART (Compilation_Start), CEND (Compilation_End), CERR (Compile_Error), CWARN (Compile_Warning), BSTART (Binder_Start), BEND (Binder_End), BERR (Binder_Error), BWARN (Binder_Warning), EXSTART (Execution_Start), EXEND (Execution_End), EXFAIL (Execution_Failure), EXNA (Execution_Not_Applicable), EXSACT (Execution_Special_Action), EVENT (see below). These values are case-insensitive. "EVENT" is treated as specifying a comment; it usually appears in column headers.
Timestamp	The timestamp, double quoted, in the format specified by Ada.Calendar.Formatting.Image.
Name	The double quoted name of the source file, main subprogram, or test. For Compilation events, this is the simple name of the source file. For Binder events, this is the name of the main subprogram. For Execution events, this is the name of the test as passed to Report.Test.
Line	For Compilation_Start, the first line of the current compilation unit. (Usually 1, unless there are multiple compilation units in a single file.) For Compile_Error or Compile_Warning, the line number where that error or warning is reported. (This is critical to the correct operation of the grading tool.) Otherwise, it is not used and can be omitted other than the comma separator.
Position	For Compile_Error or Compile_Warning, the position within the line that on which the error is reported. An implementation does not have to provide a meaningful Position for errors (use the -No_Position option on the Grading Tool - see 6.1.4 if this is true for your implementation). Otherwise, it is not used and can be omitted other than the comma separator.
Message	The double quoted message (make sure to replace any double quotes, as they are not allowed in double quoted strings). For Compile and Binder Errors and Warnings, this is the message emitted by the appropriate tool. For Execution events, this is the message passed to Report. For End events, this is an implementation-defined result of the operation (OK, with Errors, Passed, Failed, and so on). If there is no appropriate message, nothing need be written for this field (as it is last, there is no trailing comma).

The order of the event records in the event trace file is unimportant to the Grading Tool; it will sort the records appropiately before grading. (The order of the timestamps in the records does matter; running before compiling and the like indicate a test processing problem.)

The record types used by the Ada implementation of this file can be found in Trace.A.

There is an example of writing an event trace file in the file Report.A, in procedure Put_Event_Trace. Most of the code involves limiting the length of, and removing any double quotes from, the (quoted) message string. Note that Put_Event_Trace writes column headers into a new file, so that headers exist if the file is loaded into a spreadsheet or database. This is recommended for any tool that creates an event trace.

The Grading Tool treats any record that starts with EVENT as a comment; this skips any headers and allows event trace files to be concatenated together for combined processing.

For the purposes of an event trace, a "compile" is the part of an Ada implementation that processes Ada source code and provides diagnostics to diagnose Ada errors (specifically syntax errors, resolution errors, and violations of Legality Rules). This does not need to be a single phase or program; it could be several cooperating programs. Moreover, the "compile" events only need to include parts of the implementation that are involved in diagnosing errors. Code generation and optimization are part of a conventional compiler that can be omitted from the "compile" as defined for an event trace. (A failure in one of these phases not included in "compile" would probably cause a test to be graded as crashed or with a failed bind.)

Similarly, a "bind" is the part of an Ada implementation that creates an Ada partition and enforces post-compilation rules not enforced by the compile stage. (The compiler is allowed to enforce post-compilation rules y the Ada Standard.) This also does need not be a single program, and it only needs to include phases that enforce Ada errors. For instance, a system linker need not be included in the event trace for the "bind" operation.

Not all of the information in an event trace is currently used by the Grading Tool. We included additional information (like warnings) in part because future versions of the ACATS tools might need them and changing the format in the future could be very disruptive. In addition, it's possible that this compiler-independent event format could be useful to other future ACATS tools or even third-party tools having nothing to do with the ACATS. As such, we included all of the information that seemed potentially useful.

Here is part of an event trace for Chapter 5 C-Tests:

```
Event,"Timestamp","Name","Line","Position","Message"
CSTART,"2016-05-16 23:16:41.05","C51004A.ADA", 1, 1,""
CEND,"2016-05-16 23:16:41.13","C51004A.ADA",,,"OK"
BSTART,"2016-05-16 23:16:41.14","C51004A",,,""
BEND,"2016-05-16 23:16:41.27","C51004A",,,"OK"
EXSTART,"2016-05-16 23:16:41.33","C51004A",,,"CHECK THAT LABELS, LOOP IDENTIFIERS,
AND BLOCK"
EXEND,"2016-05-16 23:16:41.33","C51004A",,,"Passed"
CSTART,"2016-05-16 23:16:41.38","C52005A.ADA", 1, 1,""
CEND,"2016-05-16 23:16:41.44","C52005A.ADA",,,"OK"
BSTART,"2016-05-16 23:16:41.45","C52005A",,,""
BEND,"2016-05-16 23:16:41.56","C52005A",,,"OK"
EXSTART,"2016-05-16 23:16:41.64","C52005A",,,"CHECK THAT CONSTRAINT_ERROR EXCEPTION
IS RAISED"
EXEND,"2016-05-16 23:16:41.64","C52005A",,,"Passed"
CSTART,"2016-05-16 23:16:41.70","C52005B.ADA", 1, 1,""
CEND,"2016-05-16 23:16:41.77","C52005B.ADA",,,"OK"
BSTART,"2016-05-16 23:16:41.78","C52005B",,,""
BEND,"2016-05-16 23:16:41.89","C52005B",,,"OK"
EXSTART,"2016-05-16 23:16:41.95","C52005B",,,"CHECK THAT CONSTRAINT_ERROR EXCEPTION
IS RAISED"
EXEND,"2016-05-16 23:16:41.95","C52005B",,,"Passed"
CSTART,"2016-05-16 23:17:06.36","C55B07B.DEP", 1, 1,""
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 45, 14,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 47, 39,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 51, 27,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 52, 27,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 57, 32,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 58, 32,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 58, 52,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 83, 21,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 83, 21,"Only discrete types may b
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 99, 18,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.41","C55B07B.DEP", 107, 18,"Identifier is not defined"
CERR,"2016-05-16 23:17:06.42","C55B07B.DEP", 109, 26,"Identifier is not defined"
CEND,"2016-05-16 23:17:06.42","C55B07B.DEP",,,"Aborted by semantic errors"
BSTART,"2016-05-16 23:17:06.44","C55B07B",,,""
BERR,"2016-05-16 23:17:06.44","C55B07B",,,"Main program file not found"
BEND,"2016-05-16 23:17:06.44","C55B07B",,,"Aborted by errors"
```

6.2.2 Creating an Event Trace directly by the implementation

Creating an event trace for a particular implementation is a problem for the ACATS user and their implementer (the ACATS user need not be an implementer, of course). We can only give general guidance on approaching this problem.

One approach is the for implementer to directly add event trace creation to their tools. In this approach, options would be added to the compiler and binder that would write events to an event trace file. The ACATS user would set the Generate_Event_Trace_File constant in Report.A to True. The user would use this new compiler and binder option to write events to ETrace.csv, as will the Report package. At strategic times, the ETrace.csv file could be copied to a new name in order to keep its size manageable.

This approach is likely to be the easiest, as all of the information needed to create an event is likely to be available in a compiler error handler. Moreover, writing an event is an easy operation in Ada (see Report.Put_Event_Trace) and likely most other languages. And by directly including this capability in the compiler, it is available to all users and likely needs little modification even if the compiler error handling is changed significantly. The effort needed is small and one-time.

However, this approach may not work in some cases:

1. The ACATS user is not the implementer, so modifying the compiler is not possible;

2. An important part of the compiler or binder is a third party tool, so adding an option is not feasible;
3. The ACATS testing target does not have a file system, so having report write an event trace is not possible.

In each of these cases, the technique outlined in the next section is preferable.

Note that in case 2, not all parts of an Ada compilation system need to be able to generate an event trace. Any part that can fail only in the case of a bug elsewhere in the system (that is, does not diagnose errors in the Ada code) can be ignored for the purposes of an event trace. For instance, the use of a system-provided linker after the completion of the bind step need not be reflected in an event trace, as it has no role in the enforcement of Ada Post-Compilation Rules. For more on this topic, see 6.2.1.

6.2.3 Creating an Event Trace from Listings

If it is not possible or desirable to modify the implementation to create an event trace, an event trace can be created from a listing of test processing. Such a listing would capture all of the compile, bind, and run steps for a set of tests. The Generate_Event_Trace_File constant in Report.A would remain False.

A tool then would have to be constructed to read those listings and convert them to an event trace. This would be specific to a particular implementation, and the tool would potentially break due to future changes in an implementation.

The tool would probably have to gather information from several parts of a listing. For instance, most programs don't display a time stamp when they run, so it would be necessary to use a system tool or tiny Ada program to add those to strategic points in a listing. (Using a single timestamp for an entire compilation of a compilation unit, including any reported errors, is sufficient to meet the requirements of the Grading Tool.) Note that the initial message from Report includes a timestamp, so there is no need to add one for the execution of tests.

The listing converter tool would have to be able to extract events from the compiler, the binder, and running of ACATS tests. Each of these likely has a different format, so the tool would need many rules to extract all of the events.

While a listing converted tool is potentially more fragile (due to more dependence on the exact layout of messages for an implementation), it would have the possibility of detecting test failures after the reporting of Passed (due to a finalization or tasking failure) - such "aberrant behavior" (as defined in 5.6.1) cannot be detected when Report.A itself writes the event trace (as the failure happens outside of the control of Report).

6.3 Test Summary Files

A test summary file is a machine-readable distillation of one or more ACATS tests. The summary includes the information about the test that is needed by the grading tool, in a much more convenient format than the original source files. Test summaries are created with the Summary tool (see 6.1.1 and 6.1.5).

6.3.1 Test Summary File Reference

A test summary file is a CSV (Comma Separated Value) file of test summary records. See 6.5, “CSV File Reference” for the general rules for constructing a CSV file.

A test summary file record contains the following comma-separated fields on a single line:

Kind
: KIND (see below), ERROR (Error), PERROR (Possible_Error), OERROR (Optional_Error), NAERR (NA_Error), ACRQMT (Annex_C_Requirement), OK (OK), UPACKSPEC (Package_Specification), UFUNCSPEC (Function_Specification), UPROCSPEC (Procedure_Specification), UGENPACK (Generic_Package), UGENFUNC (Generic_Function), UGENPROC (Generic_Procedure), UPACKBODY (Package_Body), UFUNCBODY (Function_Body), UPROCBODY (Procedure_Body), UPACKINST (Package_Instantiation), UFUNCINST (Function_Instantiation), UPROCINST (Procedure_Instantiation), UPACKREN (Package_Renaming), UFUNCREN (Function_Renaming), UPROCREN (Procedure_Renaming), UGPACKREN (Generic_Package_Renaming), UGFUNCREN (Generic_Function_Renaming), UGPROCREN (Generic_Procedure_Renaming), PACKSUB (Package_Subunit), PROCSUB (Procedure_Subunit), FUNCSUB (Function_Subunit), TASKSUB (Task_Subunit), PROTSUB (Protected_Subunit), PRAGMA (Configuration_Pragma) These values are case-insensitive. "KIND" is treated as specifying a comment; it usually appears in column headers. All of the U kinds are varieties of compilation units; as are the subunits and PRAGMA.

Source_Name
: The quoted simple name of the source file.

Start_Line
: For a compilation unit, the starting line of the compilation unit. This may include leading whitespace and comments, but must be no later than the first significant text of the unit, and it may not include part of some preceding unit. For other kinds of records, this represents the first line of the range that indicates a trigger for this record. (For instance, for an ERROR record, this represents the first line in the range in which an error must be reported.)

Start_Position
: For a compilation unit, the starting position within the Start_Line for the compilation unit. This may include whitespace or comments (always setting this to 1 would be allowed), but must be no later than the first significant character of the line. (The ACATS never puts multiple compilation units on the same line.)

  For other kinds of records, the starting position within Start_Line for the trigger for this record. Whether or not positions are used for grading is selectable by an option to the Grading Tool - see 6.1.4. (Use of positions is not required.)

End_Line
: For a compilation unit, the end line of the compilation unit. This may include trailing whitespace and comments, but must be no earlier than the last significant text of the unit, and it may not include part of some succeeding unit. For other kinds of records, this represents the last line of the range that indicates a trigger for this record.

End_Position
: For a compilation unit, the end position within the Start_Line for the compilation unit. This may include whitespace or comments, but must be no earlier than the last significant character of the line. For other kinds of records, this is the ending position within End_Line for the trigger for this record.

Name_Label
: For a compilation unit, the quoted compilation unit name. For a Possible_Error, the quoted set name. For other kinds of records, this value can be omitted other than the separating comma.

Flag
: Only used for compilation units; for other kinds of records it can be completely omitted (as it is last, there is no trailing comma). For a compilation unit, it can be omitted (meaning a required, non-main subprogram unit), MAIN (a required, main subprogram unit), OPT (an optional, non-main subprogram unit), or OPTMAIN (an optional, main

subprogram unit). An optional unit is one that is not required to be processed (this is indicated by comments within the test). As with kinds, this value is case-insensitive.

These fields do not quite match the components used in the associated Ada record; a few were combined to simplify the file format (at a cost of some additional complexity in reading the records).

The order of records within a test summary file is unimportant to the Grading Tool; it will sort the records appropiately before grading.

The compilation unit start and end information is primarily used to determine to which compilation unit (and thus, which compilation) a particular ERROR or other kind of record belongs. The compilation unit records themselves are secondarily used to ensure that all required units have been processed.

The ranges for ERROR, OK, and other non-compilation unit record kinds are used to determine if an appropriate error is (or is not) present for that record. The limits can be determined by a range indicator (see 6.3.2) if present. Otherwise, the limits will contain at least all of the significant text of the line that contains the tag. (In some cases, these default limits may be expanded.)

The record types used by the Ada implementation of this file can be found in Tst_Sum.A, along with a routine (Write_Summary_Record) used to write a summary record to a test summary file. Note that Write_Summary_Record writes column headers into a new file, so that headers exist if the file is loaded into a spreadsheet or database. It is recommended to use Tst_Sum.A as part of any tool that creates a test summary file (there being little reason to reinvent this wheel).

Code to read a test summary file into an array can be found in Grd_Data.A.

Here is part of a test summary file for Chapter 5 C-Tests:

```
Kind,Source_Name,Start Line,Start Pos,End Line,End Pos,Name_Label,Flag
UPROCBODY,C51004A.ADA,38,1,261,12,C51004A,MAIN
UPROCBODY,C52005A.ADA,33,1,177,12,C52005A,MAIN
UPROCBODY,C52005B.ADA,33,1,115,12,C52005B,MAIN
UPROCBODY,C52005C.ADA,33,1,79,12,C52005C,MAIN
UPROCBODY,C52005D.ADA,32,1,182,12,C52005D,MAIN
UPROCBODY,C52005E.ADA,32,1,129,12,C52005E,MAIN
UPROCBODY,C52005F.ADA,32,1,86,12,C52005F,MAIN
UPACKSPEC,C540001.A,50,1,53,14,C540001_0,
UPACKSPEC,C540001.A,57,1,73,14,C540001_1,
UPACKBODY,C540001.A,77,1,105,14,C540001_1,
UGENPACK,C540001.A,109,1,121,14,C540001_2,
UPACKBODY,C540001.A,125,1,137,14,C540001_2,
UGENFUNC,C540001.A,141,1,149,69,C540001_3,
UFUNCBODY,C540001.A,153,1,157,14,C540001_3,
UGENPACK,C540001.A,161,1,172,14,C540001_4,
UPACKBODY,C540001.A,176,1,187,14,C540001_4,
UPACKSPEC,C540001.A,191,1,205,14,C540001_5,
UPACKBODY,C540001.A,209,1,222,14,C540001_5,
UPROCBODY,C540001.A,226,1,410,12,C540001,MAIN
UPROCBODY,C540002.A,66,1,235,12,C540002,MAIN
UPACKSPEC,C540003.A,62,1,81,14,C540003_0,
UPROCBODY,C540003.A,83,1,240,12,C540003,MAIN
NAERR,C55B07A.DEP,45,6,45,26,,
UPROCBODY,C55B07A.DEP,41,1,126,14,C55B07A,MAIN
NAERR,C55B07B.DEP,45,6,45,27,,
UPROCBODY,C55B07B.DEP,41,1,126,14,C55B07B,MAIN
```

6.3.2 Range Indicators

Optional range indicators (sometimes known as location indicators) can appear after the various markers in an ACATS test source file. These describe the exact range of the reported location of an expected error.

Range indicators are usually used to expand the range of an error beyond the same line as the ERROR: or other marker.

The format of a range indicator is:

```
{[sl:]sp[;[el:]ep]}
```

In the above, '{' and '}' are literal, while '[' and ']' indicate optionality. Each of the four values is relative, so it is one or two digits, with an optional minus sign for `el`. Omitted values are assumed to be zero. Specifically:

`sl`	Start Line – offset *before* the current line for the start of the error range.
`sp`	Start Position – position offset in the line indicated by `sl`, relative to the start of the line.
`el`	End_Line – offset *before* the current line for the end of the error range. Can be negative if the end of the error range follows the error tag. But almost always should be zero.
`ep`	End Position – position offset from the last significant character in the line indication by End_Line. (The last significant character is the last non-white-space character not including any comment.)

This compact representation was chosen because of the limited space given the ACATS line length limit (see 4.5) and a desire to avoid unnecessarily cluttering tests with extraneous information.

6.4 Manual Grading Request Files

The ACATS Grading Tool takes a file containing a list of tests that may require manual grading.

The file is just a list of ACATS test names (7 characters each), one per line. Ada comments (anywhere on a line) and blank lines are also allowed.

If the Grading Tool encounters a failure for one of the tests in the manual grading file for that grading run, it will report that the test needs manual grading rather than that it failed. The manual grading list has no effect on tests that are graded as Passed or Not Applicable.

An ACATS user can add tests to this file as needed. Some of the reasons that one might want to do this are discussed in 6.1.1.

The ACAA would like to know which tests require manual grading for your implementation. Please send manual grading files annotated with comments as to why they require manual grading to the ACAA Technical Agent, agent@ada-auth.org. The ACAA will use this information to determine which, if any tests, require repair to better provide error range information (potentially including alternative error locations.)

For formal testing, the ACAL should be aware of all tests included in any manual grading file used.

6.5 CSV File Reference

Several of the files used by the Grading Tool are .CSV (Comma Separated Value) files. This format was chosen as it:

- is a pure text format that is easy to write in Ada. Records can be appended to an existing file by opening the file with Text_IO in Append_Mode, Put_Line the line of values for the record, then closing the file. No end markers need to be moved.
- is reasonably easy to read in Ada using Text_IO. Get_Line and some string operations are sufficient.

- can be read by most spreadsheet and database programs. Thus, we don't need to provide tools to inspect, check, or edit the contents of these files.

A CSV consists of list of records, one record per line. Each line is a set of values, separated by commas. Each line should have the same number of values, so when loaded in a spreadsheet it becomes a series of columns with each Ada component associated with a column.

There are a number of varieties of CSV files, so we have adopted one of the simplest in order to have the widest applicability.

There are only a few restrictions on the values in our CSV files. Unquoted values must not contain commas, spaces, tabs, or semicolons. A value can be quoted with double quotes, in which case commas, spaces, tabs, and semicolons are allowed, but double quotes are **not** allowed in quoted strings. Line breaks are not allowed in or outside of values of a single record (they delimit the records).

We require that values are limited to 150 characters (primarily to prevent excessively long messages from making reports hard to read).

There is an example of writing a CSV file (an event trace - see 6.2) in the file Report.A, in procedure Put_Event_Trace. Most of the code involves limiting the length of, and removing any double quotes from, the (quoted) message string.

Annex A: Version Description

ACATS 4.1 includes 39nn tests in 45nn files, not including foundation and other support units. From Version 4.0 to 4.1, 134 tests were added, comprising 179 files (including six foundation files). 13 tests and one foundation (14 files) were modified. Seven support files were added and two deleted. Three tests (3 files) were removed. 36 documentation files were added, 161 documentation files were modified, and one documentation file was deleted.

The following sections present a detailed description of ACATS 4.1, as follows:

A.1 Core Test Files

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

a22006b.ada	a38106d.ada	a74105b.ada	a83c01h.ada
a22006c.ada	a38106e.ada	a74106a.ada	a83c01i.ada
a22006d.ada	a49027a.ada	a74106b.ada	a85007d.ada
a26007a.tst	a49027b.ada	a74106c.ada	a85013b.ada
a27003a.ada	a49027c.ada	a74205e.ada	a87b59a.ada
a29003a.ada	a54b01a.ada	a74205f.ada	a95001c.ada
a2a031a.ada	a54b02a.ada	a83009a.ada	a95074d.ada
a33003a.ada	a55b12a.ada	a83009b.ada	a97106a.ada
a34017c.ada	a55b13a.ada	a83a02a.ada	a99006a.ada
a35101b.ada	a55b14a.ada	a83a02b.ada	aa2010a.ada
a35402a.ada	a71004a.ada	a83a06a.ada	aa2012a.ada
a35801f.ada	a73001i.ada	a83a08a.ada	ac1015b.ada
a35902c.ada	a73001j.ada	a83c01c.ada	ac3106a.ada

ac3206a.ada	b24009b.ada	b324003.a	b34014i.ada
ac3207a.ada	b24104a.ada	b330001.a	b34014m.ada
ad7001b.ada	b24204a.ada	b33001a.ada	b34014o.ada
ad7001c0.ada	b24204b.ada	b33101a.ada	b34014q.ada
ad7001c1.ada	b24204c.ada	b33102a.ada	b34014s.ada
ad7001d0.ada	b24204d.ada	b33102b.ada	b34014v.ada
ad7001d1.ada	b24204e.ada	b33102c.ada	b34014z.ada
ad7006a.ada	b24204f.ada	b33102d.ada	b35004a.ada
ad7101a.ada	b24205a.ada	b33102e.ada	b35101a.ada
ad7101c.ada	b24206a.ada	b33201a.ada	b35103a.ada
ad7102a.ada	b24206b.ada	b33201b.ada	b35103b.ada
ad7103a.ada	b24211b.ada	b33201c.ada	b35302a.ada
ad7103c.ada	b25002a.ada	b33201d.ada	b354001.a
ad7104a.ada	b25002b.ada	b33201e.ada	b35401a.ada
ad7201a.ada	b26001a.ada	b33204a.ada	b35401b.ada
ad7203b.ada	b26002a.ada	b33205a.ada	b35403a.ada
ad7205b.ada	b26005a.ada	b33302a.ada	b35501a.ada
ad8011a.tst	b28001a.ada	b34001b.ada	b35501b.ada
ada101a.ada	b28001b.ada	b34001e.ada	b35506a.ada
ae2113a.ada	b28001c.ada	b34002b.ada	b35506b.ada
ae2113b.ada	b28001d.ada	b34003b.ada	b35506c.ada
ae3002g.ada	b28001e.ada	b34004b.ada	b35506d.ada
ae3101a.ada	b28001r.ada	b34005b.ada	b35701a.ada
ae3702a.ada	b28001s.ada	b34005e.ada	b35709a.ada
ae3709a.ada	b28001t.ada	b34005h.ada	b35901a.ada
b22001a.tst	b28001u.ada	b34005k.ada	b35901c.ada
b22001b.tst	b28001v.ada	b34005n.ada	b35901d.ada
b22001c.tst	b28001w.ada	b34005q.ada	b35a01a.ada
b22001d.tst	b29001a.ada	b34005t.ada	b35a08a.ada
b22001e.tst	b2a003a.ada	b34006b.ada	b36001a.ada
b22001f.tst	b2a003b.ada	b34006e.ada	b36002a.ada
b22001g.tst	b2a003c.ada	b34006h.ada	b36101a.ada
b22001h.ada	b2a003d.ada	b34006k.ada	b36102a.ada
b22001i.tst	b2a003e.ada	b34007b.ada	b36103a.ada
b22001j.tst	b2a003f.ada	b34007e.ada	b36105c.dep
b22001k.tst	b2a005a.ada	b34007h.ada	b36171a.ada
b22001l.tst	b2a005b.ada	b34007k.ada	b36171b.ada
b22001m.tst	b2a007a.ada	b34007n.ada	b36171c.ada
b22001n.tst	b2a010a.ada	b34007q.ada	b36171d.ada
b23002a.ada	b2a021a.ada	b34007t.ada	b36171e.ada
b23004a.ada	b32103a.ada	b34008b.ada	b36171f.ada
b23004b.ada	b32104a.ada	b34009b.ada	b36171g.ada
b24001a.ada	b32106a.ada	b34009e.ada	b36171h.ada
b24001b.ada	b32201a.ada	b34009h.ada	b36171i.ada
b24001c.ada	b32202a.ada	b34009k.ada	b36201a.ada
b24005a.ada	b32202b.ada	b34011a.ada	b36307a.ada
b24005b.ada	b32202c.ada	b34014b.ada	b370001.a
b24007a.ada	b324001.a	b34014d.ada	b370002.a
b24009a.ada	b324002.a	b34014f.ada	b37004a.ada

b37004c.ada	b391003.a	b3a10070.a	b416a01.a
b37004d.ada	b391004.a	b3a10071.a	b420001.a
b37004e.ada	b392001.a	b3a1008.a	b430001.a
b37004f.ada	b392002.a	b3a10090.a	b43002d.ada
b37004g.ada	b392003.a	b3a10091.a	b43002e.ada
b3710010.a	b392004.a	b3a1010.a	b43002f.ada
b3710011.a	b392005.a	b3a1a01.a	b43002g.ada
b3710012.a	b392006.a	b3a1a02.a	b43002h.ada
b3710013.a	b392007.a	b3a1a030.a	b43002i.ada
b3710014.am	b392008.a	b3a1a031.a	b43002j.ada
b37101a.ada	b392009.a	b3a1a04.a	b43002k.ada
b37102a.ada	b392010.a	b3a1a05.a	b43005a.ada
b37104a.ada	b392011.a	b3a2002.a	b43005b.ada
b37106a.ada	b393001.a	b3a2003.a	b43005f.ada
b37201a.ada	b393002.a	b3a2004.a	b431001.a
b37201b.ada	b393003.a	b3a2005.a	b431002.a
b37203a.ada	b393004.a	b3a2006.a	b431003.a
b37301i.ada	b393005.a	b3a2007.a	b431004.a
b37301j.ada	b393006.a	b3a2008.a	b431005.a
b37302a.ada	b393007.a	b3a2009.a	b431006.a
b37303a.ada	b393008.a	b3a2010.a	b43101a.ada
b37309b.ada	b393009.a	b3a2011.a	b43102b.ada
b37310b.ada	b393010.a	b3a2012.a	b43105c.ada
b37311a.ada	b3930110.a	b3a2013.a	b432001.a
b37401a.ada	b3930111.a	b3a2014.a	b43201a.ada
b37409b.ada	b3930112.a	b3a2015.a	b43201c.ada
b380001.a	b3930113.a	b3a2016.a	b43201d.ada
b38003a.ada	b394001.a	b3a20170.a	b43202a.ada
b38003b.ada	b394a01.a	b3a20171.a	b43202c.ada
b38003c.ada	b394a02.a	b3a20172.a	b43209b.ada
b38003d.ada	b394a03.a	b3a20173.a	b43221a.ada
b38008a.ada	b394a04.a	b3a20174.a	b43221b.ada
b38008b.ada	b394a05.a	b41101a.ada	b43223a.ada
b38009a.ada	b3a0001.a	b41101c.ada	b433001.a
b38009d.ada	b3a0002.a	b41201a.ada	b433002.a
b38103a.ada	b3a0003.a	b41201c.ada	b433003.a
b38103b.ada	b3a0004.a	b41202c.ada	b44001a.ada
b38103c0.ada	b3a0005.a	b41202d.ada	b44001b.ada
b38103c1.ada	b3a0006.a	b413001.a	b44002b.ada
b38103c2.ada	b3a0007.a	b413002.a	b44002c.ada
b38103c3.ada	b3a0008.a	b413003.a	b44004a.ada
b38103d.ada	b3a1001.a	b413004.a	b44004b.ada
b38103e0.ada	b3a1002.a	b41324b.ada	b44004c.ada
b38103e1.ada	b3a1003.a	b41325b.ada	b44004d.ada
b38105b.ada	b3a10040.a	b41327b.ada	b44004e.ada
b38203a.ada	b3a10041.a	b415001.a	b45102a.ada
b390001.a	b3a10042.a	b415002.a	b45116a.ada
b391001.a	b3a1005.a	b416001.a	b45121a.ada
b391002.a	b3a1006.a	b416002.a	b45204a.ada

b45205a.ada	b455002.a	b49004a.ada	b54b01b.tst
b45206c.ada	b45501a.ada	b49005a.ada	b54b01c.ada
b45207a.ada	b45501b.ada	b49007a.ada	b54b02b.ada
b45207b.ada	b45501c.ada	b49007b.ada	b54b02c.ada
b45207c.ada	b45522a.ada	b49008a.ada	b54b02d.ada
b45207d.ada	b45537a.ada	b49008c.ada	b54b04a.ada
b45207g.ada	b45601a.ada	b49009b.ada	b54b04b.ada
b45207h.ada	b45625a.ada	b49009c.ada	b54b05a.ada
b45207i.ada	b45661a.ada	b49010a.ada	b54b06a.ada
b45207j.ada	b457001.a	b49011a.ada	b551001.a
b45207m.ada	b457002.a	b4a010c.ada	b551002.a
b45207n.ada	b457003.a	b4a016a.ada	b552001.a
b45207o.ada	b457004.a	b51001a.ada	b552a01.a
b45207p.ada	b457005.a	b51004b.ada	b552a02.a
b45207s.ada	b457006.a	b51004c.ada	b552a03.a
b45207t.ada	b457007.a	b52002a.ada	b552a04.a
b45207u.ada	b460001.a	b52002b.ada	b552a05.a
b45207v.ada	b460002.a	b52002c.ada	b55a01a.ada
b45208a.ada	b460004.a	b52002d.ada	b55a01d.ada
b45208b.ada	b460005.a	b52002e.ada	b55a01e.ada
b45208c.ada	b460006.a	b52002f.ada	b55a01j.ada
b45208g.ada	b46002a.ada	b52002g.ada	b55a01k.ada
b45208h.ada	b46003a.ada	b52004a.ada	b55a01l.ada
b45208i.ada	b46004a.ada	b52004b.ada	b55a01n.ada
b45208m.ada	b46004b.ada	b52004c.ada	b55a01o.ada
b45208n.ada	b46004c.ada	b52004d.dep	b55a01t.ada
b45208s.ada	b46004d.ada	b52004e.dep	b55a01u.ada
b45208t.ada	b46004e.ada	b53001a.ada	b55a01v.ada
b45209a.ada	b46005a.ada	b53001b.ada	b55b01a.ada
b45209b.ada	b47001a.ada	b53002a.ada	b55b01b.ada
b45209c.ada	b480001.a	b53002b.ada	b55b09b.ada
b45209d.ada	b480002.a	b53009a.ada	b55b09c.dep
b45209e.ada	b480003.a	b53009b.ada	b55b09d.dep
b45209f.ada	b48001a.ada	b53009c.ada	b55b12b.ada
b45209g.ada	b48001b.ada	b540001.a	b55b12c.ada
b45209h.ada	b48002a.ada	b540002.a	b55b17a.ada
b45209i.ada	b48002b.ada	b54a01b.ada	b55b17b.ada
b45209j.ada	b48002c.ada	b54a01f.ada	b55b17c.ada
b45209k.ada	b48002d.ada	b54a01g.ada	b55b18a.ada
b45221a.ada	b48002e.ada	b54a01l.ada	b56001a.ada
b45261a.ada	b48002g.ada	b54a05a.ada	b56001d.ada
b45261b.ada	b48003a.ada	b54a05b.ada	b56001e.ada
b45261c.ada	b48003b.ada	b54a10a.ada	b56001f.ada
b45261d.ada	b48003d.ada	b54a12a.ada	b56001g.ada
b45301a.ada	b48003e.ada	b54a20a.ada	b56001h.ada
b45301b.ada	b490001.a	b54a21a.ada	b57001a.ada
b45301c.ada	b490002.a	b54a25a.ada	b57001b.ada
b45302a.ada	b490003.a	b54a60a.ada	b57001c.ada
b45341a.ada	b49002a.ada	b54a60b.ada	b57001d.ada

b58003a.ada	b650005.a	b7200016.a	b74103e.ada
b58003b.ada	b650006.a	b730001.a	b74103g.ada
b59001a.ada	b65002a.ada	b730002.a	b74103i.ada
b59001c.ada	b65002b.ada	b730003.a	b74104a.ada
b59001d.ada	b660001.a	b730004.a	b74105a.ada
b59001e.ada	b660002.a	b730005.a	b74105c.ada
b59001f.ada	b660003.a	b7300060.a	b74201a.ada
b59001g.ada	b66001a.ada	b7300061.a	b74202a.ada
b59001h.ada	b66001b.ada	b7300062.a	b74202b.ada
b59001i.ada	b66001c.ada	b7300063.am	b74202c.ada
b610001.a	b66001d.ada	b730007.a	b74202d.ada
b610002.a	b6700010.a	b730008.a	b74203b.ada
b61001f.ada	b6700011.am	b730009.a	b74203c.ada
b61005a.ada	b6700012.a	b730010.a	b74203d.ada
b61006a.ada	b6700013.a	b73001a.ada	b74203e.ada
b61011a.ada	b67001a.ada	b73001b.ada	b74205a.ada
b62001a.ada	b67001b.ada	b73001c.ada	b74207a.ada
b62001b.ada	b67001c.ada	b73001d.ada	b74304a.ada
b62001c.ada	b67001d.ada	b73001e.ada	b74304b.ada
b62001d.ada	b67001h.ada	b73001f.ada	b74304c.ada
b63001a.ada	b67001i.ada	b73001g.ada	b74404a.ada
b63001b.ada	b67001j.ada	b73001h.ada	b74404b.ada
b63005a.ada	b67001k.ada	b73004a.ada	b74409a.ada
b63005b.ada	b67004a.ada	b73004b0.ada	b750a01.a
b63006a.ada	b680001.a	b73004b1.ada	b750a02.a
b63009a.ada	b71001a.ada	b73004b2.ada	b750a03.a
b63009b.ada	b71001b.ada	b7310010.a	b750a04.a
b63009c0.ada	b71001c.ada	b7310011.a	b750a05.a
b63009c1.ada	b71001d.ada	b7310012.a	b750a06.a
b63009c2.ada	b71001f.ada	b7310013.a	b750a08.a
b63009c3.ada	b71001g.ada	b7310014.a	b750a09.a
b63103a.ada	b71001h.ada	b7310015.a	b750a10.a
b640001.a	b71001i.ada	b7310016.am	b750a11.a
b64002a.ada	b71001j.ada	b731a01.a	b750a12.a
b64002c.ada	b71001l.ada	b731a02.a	b750a13.a
b64003a.ada	b71001m.ada	b732001.a	b810001.a
b64004a.ada	b71001n.ada	b732c01.a	b830001.a
b64004b.ada	b71001o.ada	b732c02.a	b8300020.a
b64004c.ada	b71001p.ada	b740001.a	b8300021.a
b64004d.ada	b71001r.ada	b740002.a	b8300022.a
b64004e.ada	b71001t.ada	b7400030.a	b8300023.a
b64004f.ada	b71001u.ada	b7400031.a	b8300024.a
b641001.a	b71001v.ada	b7400032.am	b8300025.am
b641002.a	b7200010.a	b74001a.ada	b83001a.ada
b64101a.ada	b7200011.a	b74001b.ada	b83003a.ada
b650001.a	b7200012.a	b74101a.ada	b83003b0.ada
b650002.a	b7200013.a	b74101b.ada	b83003b1.ada
b650003.a	b7200014.a	b74103a.ada	b83003b2.ada
b650004.a	b7200015.a	b74103d.ada	b83003b3.ada

b83003b4.ada	b8310052.a	b85001a.ada	b91002f.ada
b83003c.ada	b8310053.a	b85001b.ada	b91002g.ada
b83004a.ada	b83a01a.ada	b85001c.ada	b91002h.ada
b83004b0.ada	b83a01b.ada	b85001d.ada	b91002i.ada
b83004b1.ada	b83a01c.ada	b85001e.ada	b91002j.ada
b83004b2.ada	b83a05a.ada	b85001f.ada	b91002k.ada
b83004b3.ada	b83a06b.ada	b85001g.ada	b91002l.ada
b83004c0.ada	b83a06h.ada	b85001h.ada	b91003a.ada
b83004c1.ada	b83a07a.ada	b85001i.ada	b91003b.ada
b83004c2.ada	b83a07b.ada	b85001j.ada	b91003c.ada
b83004d0.ada	b83a07c.ada	b85001k.ada	b91003d.ada
b83004d1.ada	b83a08b.ada	b85001l.ada	b91003e.ada
b83004d2.ada	b83a09a.ada	b85002a.ada	b91004a.ada
b83004d3.ada	b83b01a.ada	b85003a.ada	b91005a.ada
b83006a.ada	b83b02c.ada	b85003b.ada	b92001a.ada
b83006b.ada	b83e01a.ada	b85004a.ada	b92001b.ada
b83008a.ada	b83e01b.ada	b85008f.ada	b940001.a
b83008b.ada	b83e01c.ada	b85008g.ada	b940002.a
b83011a.ada	b83e01d.ada	b85008h.ada	b940003.a
b83023b.ada	b83e01e0.ada	b85010a.ada	b940004.a
b83024b.ada	b83e01e1.ada	b85010b.ada	b940005.a
b83024f0.ada	b83e01e2.ada	b85012a.ada	b940006.a
b83024f1.ada	b83e01e3.ada	b85013c.ada	b940007.a
b83024f2.ada	b83e01f0.ada	b85013d.ada	b950001.a
b83024f3.ada	b83e01f1.ada	b85015a.ada	b950002.a
b83026b.ada	b83e01f2.ada	b8510010.a	b95001a.ada
b83027b.ada	b83e01f3.ada	b8510011.a	b95001b.ada
b83027d.ada	b83e01f4.ada	b8510012.am	b95001d.ada
b83028b.ada	b83e01f5.ada	b851002.a	b95002a.ada
b83029b.ada	b83e01f6.ada	b851003.a	b95003a.ada
b83030b.ada	b83e11a.ada	b851004.a	b95004a.ada
b83030d.ada	b83f02a.ada	b854001.a	b95004b.ada
b83031b.ada	b83f02b.ada	b860001.a	b95006a.ada
b83031d.ada	b83f02c.ada	b86001a0.ada	b95006b.ada
b83031f.ada	b840001.a	b86001a1.ada	b95006c.ada
b83032b.ada	b8400020.a	b87b23b.ada	b95006d.ada
b83033b.ada	b8400021.a	b87b26a.ada	b95007a.ada
b83041e.ada	b8400022.a	b87b48c.ada	b95007b.ada
b8310010.a	b8400023.a	b91001b.ada	b95020a.ada
b8310011.a	b8400024.a	b91001c.ada	b95020b0.ada
b8310012.a	b8400025.a	b91001d.ada	b95020b1.ada
b831002.a	b840003.a	b91001e.ada	b95020b2.ada
b8310030.a	b84001a.ada	b91001f.ada	b95030a.ada
b8310031.a	b84002b.ada	b91001g.ada	b95031a.ada
b8310032.a	b84004a.ada	b91002a.ada	b95032a.ada
b8310033.a	b84005b.ada	b91002b.ada	b95061a.ada
b831004.a	b84006a.ada	b91002c.ada	b95061b.ada
b8310050.a	b84007a.ada	b91002d.ada	b95061c.ada
b8310051.a	b84008b.ada	b91002e.ada	b95061d.ada

b95061e.ada	b97104d.ada	ba1010e2.ada	ba1010m4.ada
b95061f.ada	b97104e.ada	ba1010e3.ada	ba1010m5.ada
b95061g.ada	b97104f.ada	ba1010e4.ada	ba1010m6.ada
b95062a.ada	b97104g.ada	ba1010e5.ada	ba1010m7.ada
b95063a.ada	b97107a.ada	ba1010e6.ada	ba1010m8.ada
b95064a.ada	b97108a.ada	ba1010f0.ada	ba1010n0.ada
b95068a.ada	b97108b.ada	ba1010f1.ada	ba1010n2.ada
b95070a.ada	b97109a.ada	ba1010f3.ada	ba1010n3.ada
b95080a.ada	b97110a.ada	ba1010f4.ada	ba1010n4.ada
b95080c.ada	b97110b.ada	ba1010f5.ada	ba1010n5.ada
b95081a.ada	b97111a.ada	ba1010f6.ada	ba1010p0.ada
b95082a.ada	b99001a.ada	ba1010f7.ada	ba1010p2.ada
b95082b.ada	b99001b.ada	ba1010f8.ada	ba1010q0.ada
b95082c.ada	b99002a.ada	ba1010g0.ada	ba1010q1.ada
b95082d.ada	b99002b.ada	ba1010g2.ada	ba1010q3.ada
b95082e.ada	b99002c.ada	ba1010g3.ada	ba1010q4.ada
b95082f.ada	b99003a.ada	ba1010g4.ada	ba1011b0.ada
b95083a.ada	b9a001a.ada	ba1010g5.ada	ba1011b1.ada
b95094a.ada	b9a001b.ada	ba1010h0.ada	ba1011b2.ada
b95094b.ada	ba1001a0.ada	ba1010h2.ada	ba1011b3.ada
b95094c.ada	ba1001a1.ada	ba1010i0.ada	ba1011b4.ada
b951001.a	ba1001a4.ada	ba1010i1.ada	ba1011b5.ada
b951002.a	ba1001ac.ada	ba1010i3.ada	ba1011b6.ada
b952001.a	ba1001d.ada	ba1010i4.ada	ba1011b7.ada
b952002.a	ba1010a0.ada	ba1010j0.ada	ba1011b8.ada
b952003.a	ba1010a1.ada	ba1010j1.ada	ba1011c0.ada
b952004.a	ba1010a2.ada	ba1010j2.ada	ba1011c1.ada
b954001.a	ba1010a3.ada	ba1010j4.ada	ba1011c2.ada
b954003.a	ba1010b0.ada	ba1010j5.ada	ba1011c3.ada
b954004.a	ba1010b1.ada	ba1010j6.ada	ba1011c4.ada
b954005.a	ba1010b2.ada	ba1010j7.ada	ba1011c5.ada
b960001.a	ba1010b4.ada	ba1010j8.ada	ba1011c6.ada
b96002a.ada	ba1010b5.ada	ba1010k0.ada	ba1011c7.ada
b97102b.ada	ba1010b6.ada	ba1010k1.ada	ba1011c8.ada
b97102c.ada	ba1010b7.ada	ba1010k2.ada	ba1020a0.ada
b97102d.ada	ba1010b8.ada	ba1010k3.ada	ba1020a1.ada
b97102f.ada	ba1010c0.ada	ba1010k4.ada	ba1020a2.ada
b97102g.ada	ba1010c1.ada	ba1010k5.ada	ba1020a3.ada
b97102h.ada	ba1010c2.ada	ba1010k6.ada	ba1020a4.ada
b97102i.ada	ba1010c3.ada	ba1010l0.ada	ba1020a5.ada
b97103a.ada	ba1010c4.ada	ba1010l1.ada	ba1020a6.ada
b97103b.ada	ba1010c5.ada	ba1010l2.ada	ba1020a7.ada
b97103d.ada	ba1010c6.ada	ba1010l3.ada	ba1020a8.ada
b97103e.ada	ba1010d0.ada	ba1010l4.ada	ba1020b0.ada
b97103f.ada	ba1010d1.ada	ba1010l5.ada	ba1020b1.ada
b97103g.ada	ba1010d2.ada	ba1010l6.ada	ba1020b2.ada
b97104a.ada	ba1010d3.ada	ba1010m0.ada	ba1020b3.ada
b97104b.ada	ba1010e0.ada	ba1010m1.ada	ba1020b4.ada
b97104c.ada	ba1010e1.ada	ba1010m3.ada	ba1020b5.ada

ba1020b6.ada	ba1110a0.ada	ba120134.a	ba150035.a
ba1020c0.ada	ba1110a1.ada	ba120135.a	ba150036.a
ba1020c1.ada	ba1110a2.ada	ba120136.a	ba150037.a
ba1020c2.ada	ba1110a3.ada	ba120137.a	ba150038.a
ba1020c3.ada	ba1110a4.ada	ba120138.a	ba150039.a
ba1020c4.ada	ba1110a5.ada	ba120139.a	ba15003a.a
ba1020c5.ada	ba12001.a	ba12013a.a	ba15003b.am
ba1020f0.ada	ba12002.a	ba120140.a	ba16001.a
ba1020f1.ada	ba12003.a	ba120141.a	ba16002.a
ba1020f2.ada	ba12004.a	ba120142.a	ba2001a.ada
ba11001.a	ba12005.a	ba120143.a	ba2001b.ada
ba11002.a	ba12007.a	ba120144.a	ba2001c.ada
ba11003.a	ba12008.a	ba120145.a	ba2001d.ada
ba11004.a	ba120090.a	ba120146.a	ba2001f0.ada
ba11005.a	ba120091.a	ba120147.a	ba2001f1.ada
ba11007.a	ba120092.a	ba120150.a	ba2001f2.ada
ba11008.a	ba120093.a	ba120151.a	ba2003b0.ada
ba11009.a	ba120094.a	ba120152.a	ba2003b1.ada
ba11010.a	ba120095.a	ba120153.a	ba2011a0.ada
ba11011.a	ba120096.a	ba120154.a	ba2011a1.ada
ba11012.a	ba120097.a	ba120160.a	ba2011a2.ada
ba110130.a	ba120098.a	ba120161.a	ba2011a3.ada
ba110131.a	ba120100.a	ba120162.a	ba2011a4.ada
ba110132.a	ba120101.a	ba120163.a	ba2011a5.ada
ba110133.a	ba120102.a	ba120164.a	ba2011a6.ada
ba110134.am	ba120103.a	ba120165.a	ba2011a7.ada
ba110140.a	ba120104.a	ba120166.a	ba2011a8.ada
ba110141.a	ba120105.a	ba120170.a	ba2011a9.ada
ba110142.a	ba120110.a	ba120171.a	ba2013a.ada
ba1101a.ada	ba120111.a	ba120172.a	ba2013b.ada
ba1101b0.ada	ba120112.a	ba120173.a	ba21002.a
ba1101b1.ada	ba120113.a	ba13b01.a	ba210030.a
ba1101b2.ada	ba120114.a	ba13b02.a	ba210031.a
ba1101b3.ada	ba120115.a	ba15001.a	ba210032.a
ba1101b4.ada	ba120120.a	ba150020.a	ba210033.a
ba1101c0.ada	ba120121.a	ba150021.a	ba210034.a
ba1101c1.ada	ba120122.a	ba150022.a	ba210035.a
ba1101c2.ada	ba120123.a	ba150023.a	ba210040.a
ba1101c3.ada	ba120124.a	ba150024.a	ba210041.a
ba1101c4.ada	ba120125.a	ba150025.a	ba210042.a
ba1101c5.ada	ba120126.a	ba150026.a	ba210043.a
ba1101c6.ada	ba120127.a	ba150027.a	ba210044.a
ba1101e0.ada	ba120128.a	ba150028.a	ba210045.am
ba1101e1.ada	ba120129.a	ba150029.am	ba21a01.a
ba1101f.ada	ba12012a.a	ba150030.a	ba21a02.a
ba1101g.ada	ba120130.a	ba150031.a	ba21a03.a
ba1109a0.ada	ba120131.a	ba150032.a	ba3001a0.ada
ba1109a1.ada	ba120132.a	ba150033.a	ba3001a1.ada
ba1109a2.ada	ba120133.a	ba150034.a	ba3001a2.ada

ba3001a3.ada	bc1016b.ada	bc3001a.ada	bc3405b.ada
ba3001b0.ada	bc1101a.ada	bc3002a.ada	bc3405d.ada
ba3001b1.ada	bc1102a.ada	bc3002b.ada	bc3405e.ada
ba3001c0.ada	bc1103a.ada	bc3002c.ada	bc3405f.ada
ba3001c1.ada	bc1106a.ada	bc3002d.ada	bc3501a.ada
ba3001e0.ada	bc1107a.ada	bc3002e.ada	bc3501b.ada
ba3001e1.ada	bc1109a.ada	bc3005a.ada	bc3501c.ada
ba3001e2.ada	bc1109b.ada	bc3005b.ada	bc3501d.ada
ba3001e3.ada	bc1109c.ada	bc3005c.ada	bc3501e.ada
ba3001f0.ada	bc1109d.ada	bc3006a.ada	bc3501f.ada
ba3001f1.ada	bc1110a.ada	bc3009c.ada	bc3501g.ada
ba3001f2.ada	bc1201a.ada	bc3011b.ada	bc3501h.ada
ba3001f3.ada	bc1201b.ada	bc3013a.ada	bc3501i.ada
ba3006a0.ada	bc1201c.ada	bc3016g.ada	bc3501j.ada
ba3006a1.ada	bc1201d.ada	bc3018a.ada	bc3501k.ada
ba3006a2.ada	bc1201e.ada	bc3101a.ada	bc3502a.ada
ba3006a3.ada	bc1201f.ada	bc3101b.ada	bc3502b.ada
ba3006a4.ada	bc1201g.ada	bc3102a.ada	bc3502c.ada
ba3006a5.ada	bc1201h.ada	bc3102b.ada	bc3502d.ada
ba3006a6.ada	bc1201i.ada	bc3103b.ada	bc3502e.ada
ba3006b0.ada	bc1201j.ada	bc3123c.ada	bc3502f.ada
ba3006b1.ada	bc1201k.ada	bc3201a.ada	bc3502g.ada
ba3006b2.ada	bc1201l.ada	bc3201b.ada	bc3502h.ada
ba3006b3.ada	bc1202a.ada	bc3201c.ada	bc3502i.ada
ba3006b4.ada	bc1202e.ada	bc3202a.ada	bc3502j.ada
bb10001.a	bc1202f.ada	bc3202b.ada	bc3502k.ada
bb20001.a	bc1202g.ada	bc3202c.ada	bc3502l.ada
bb2001a.ada	bc1203a.ada	bc3202d.ada	bc3502m.ada
bb2002a.ada	bc1205a.ada	bc3205c.ada	bc3502n.ada
bb2003a.ada	bc1206a.ada	bc3301a.ada	bc3502o.ada
bb2003b.ada	bc1207a.ada	bc3301b.ada	bc3503a.ada
bb2003c.ada	bc1208a.ada	bc3302a.ada	bc3503c.ada
bb3001a.ada	bc1226a.ada	bc3302b.ada	bc3503d.ada
bb3002a.ada	bc1230a.ada	bc3303a.ada	bc3503e.ada
bc1001a.ada	bc1303a.ada	bc3304a.ada	bc3503f.ada
bc1002a.ada	bc1303b.ada	bc3401a.ada	bc3604a.ada
bc1005a.ada	bc1303c.ada	bc3401b.ada	bc3604b.ada
bc1008a.ada	bc1303d.ada	bc3402a.ada	bc3607a.ada
bc1008b.ada	bc1303e.ada	bc3402b.ada	bc40002.a
bc1008c.ada	bc1303f.ada	bc3403a.ada	bc50001.a
bc1009a.ada	bc1303g.ada	bc3403b.ada	bc50002.a
bc1011a.ada	bc1306a.ada	bc3403c.ada	bc50003.a
bc1011b.ada	bc2001b.ada	bc3404a.ada	bc50004.a
bc1011c.ada	bc2001c.ada	bc3404b.ada	bc51002.a
bc1012a.ada	bc2001d.ada	bc3404c.ada	bc51003.a
bc1013a.ada	bc2001e.ada	bc3404d.ada	bc51004.a
bc1014a.ada	bc2004a.ada	bc3404e.ada	bc51005.a
bc1014b.ada	bc2004b.ada	bc3404f.ada	bc51006.a
bc1016a.ada	bc30001.a	bc3405a.ada	bc51007.a

bc51011.a	bd1b02b.ada	bd5001a.ada	be3703a.ada
bc51012.a	bd1b03c.ada	bd5005a.ada	be3802a.ada
bc51013.a	bd1b05e.ada	bd5005d.ada	be3803a.ada
bc51015.a	bd1b06j.ada	bd5102a.ada	be3902a.ada
bc51016.a	bd2001b.ada	bd5102b.ada	be3903a.ada
bc51017.a	bd2a01h.ada	bd5103a.ada	bxa8001.a
bc51018.a	bd2a02a.tst	bd5104a.ada	bxac001.a
bc51019.a	bd2a03a.ada	bd7001a.ada	bxac002.a
bc51020.a	bd2a03b.ada	bd7101h.ada	bxac003.a
bc510210.a	bd2a06a.ada	bd7201c.ada	bxac005.a
bc510211.a	bd2a25a.ada	bd7203a.ada	bxai001.a
bc510212.a	bd2a35a.ada	bd7204a.ada	bxai002.a
bc510213.am	bd2a45a.ada	bd7205a.ada	bxai003.a
bc510220.a	bd2a55a.ada	bd7301a.ada	bxai004.a
bc510221.a	bd2a55b.ada	bd7302a.ada	bxai005.a
bc510222.a	bd2a67a.ada	bd8001a.tst	bxai006.a
bc510223.am	bd2a77a.ada	bd8002a.tst	bxai007.a
bc51b01.a	bd2a85a.ada	bd8003a.tst	bxai008.a
bc51b02.a	bd2a85b.ada	bd8004a.tst	bxaia01.a
bc51c01.a	bd2b01c.ada	bd8004b.tst	bxaia02.a
bc51c02.a	bd2b02a.ada	bd8004c.tst	bxaia03.a
bc53001.a	bd2b03a.ada	bdb0a01.a	bxaia04.a
bc53002.a	bd2b03b.ada	bdb3a01.a	bxb3001.a
bc54001.a	bd2b03c.ada	bdd2001.a	bxb3002.a
bc54002.a	bd2c01d.tst	bdd2002.a	bxb3003.a
bc54003.a	bd2c02a.tst	bdd2003.a	bxb3004.a
bc54a01.a	bd2c03a.tst	bdd2004.a	c23001a.ada
bc54a02.a	bd2d01c.ada	bdd2005.a	c23003a.tst
bc54a03.a	bd2d01d.ada	bde0001.a	c23003b.tst
bc54a04.a	bd2d02a.ada	bde0002.a	c23003g.tst
bc54a05.a	bd2d03a.ada	bde0003.a	c23003i.tst
bc54a06.a	bd2d03b.ada	bde0004.a	c23006a.ada
bc60001.a	bd3001a.ada	bde0005.a	c23006b.ada
bc60002.a	bd3001b.ada	bde0006.a	c23006c.ada
bc60003.a	bd3001c.ada	bde0007.a	c23006d.ada
bc60004.a	bd3002a.ada	bde0008.a	c23006e.ada
bc70001.a	bd3003a.ada	bde0009.a	c23006f.ada
bc70002.a	bd3003b.ada	bde0010.a	c23006g.ada
bc70003.a	bd3012a.ada	bde0011.a	c24002d.ada
bc70004.a	bd3013a.ada	be2101e.ada	c24003a.ada
bc70005.a	bd4001a.ada	be2101j.ada	c24003b.ada
bc70006.a	bd4002a.ada	be2114a.ada	c24003c.ada
bc70007.a	bd4003a.ada	be2116a.ada	c24106a.ada
bc70008.a	bd4003b.ada	be2208a.ada	c24202d.ada
bc70009.a	bd4003c.ada	be3002a.ada	c24203a.ada
bc70010.a	bd4006a.tst	be3002e.ada	c24203b.ada
bd11001.a	bd4007a.ada	be3205a.ada	c24207a.ada
bd11002.a	bd4009a.ada	be3301c.ada	c24211a.ada
bd1b01a.ada	bd4011a.ada	be3606c.ada	c250001.au

c250002.au	c34005g.ada	c340a01.a	c35507e.ada
c25001a.ada	c34005i.ada	c340a02.a	c35507g.ada
c25001b.ada	c34005j.ada	c341a01.a	c35507h.ada
c26006a.ada	c34005l.ada	c341a02.a	c35507i.ada
c26008a.ada	c34005m.ada	c341a03.a	c35507j.ada
c2a001a.ada	c34005o.ada	c341a04.a	c35507k.ada
c2a001b.ada	c34005p.ada	c35003a.ada	c35507l.ada
c2a001c.ada	c34005r.ada	c35003b.ada	c35507m.ada
c2a002a.ada	c34005s.ada	c35003d.ada	c35507n.ada
c2a008a.ada	c34005u.ada	c35102a.ada	c35507o.ada
c2a021b.ada	c34005v.ada	c352001.a	c35507p.ada
c32001a.ada	c34006a.ada	c354002.a	c35508a.ada
c32001b.ada	c34006d.ada	c354003.a	c35508b.ada
c32001c.ada	c34006f.ada	c35502a.ada	c35508c.ada
c32001d.ada	c34006g.ada	c35502b.ada	c35508e.ada
c32001e.ada	c34006j.ada	c35502c.ada	c35508g.ada
c32107a.ada	c34006l.ada	c35502d.tst	c35508h.ada
c32107c.ada	c34007a.ada	c35502e.ada	c35508k.ada
c32108a.ada	c34007d.ada	c35502f.tst	c35508l.ada
c32108b.ada	c34007f.ada	c35502g.ada	c35508o.ada
c32111a.ada	c34007g.ada	c35502h.ada	c35508p.ada
c32111b.ada	c34007i.ada	c35502i.ada	c35703a.ada
c32112b.ada	c34007j.ada	c35502j.ada	c35704a.ada
c32113a.ada	c34007m.ada	c35502k.ada	c35704b.ada
c32115a.ada	c34007p.ada	c35502l.ada	c35704c.ada
c32115b.ada	c34007r.ada	c35502m.ada	c35704d.ada
c324001.a	c34007s.ada	c35502n.ada	c35801d.ada
c324002.a	c34007u.ada	c35502o.ada	c35902d.ada
c324003.a	c34007v.ada	c35502p.ada	c35904a.ada
c324004.a	c34008a.ada	c35503a.ada	c35904b.ada
c324005.a	c34009a.ada	c35503b.ada	c35a02a.ada
c330001.a	c34009d.ada	c35503c.ada	c35a05a.ada
c330002.a	c34009f.ada	c35503d.tst	c35a05d.ada
c332001.a	c34009g.ada	c35503e.ada	c35a05n.ada
c340001.a	c34009j.ada	c35503f.tst	c35a05q.ada
c34001a.ada	c34009l.ada	c35503g.ada	c35a07a.ada
c34001c.ada	c34011b.ada	c35503h.ada	c35a07d.ada
c34001d.ada	c34012a.ada	c35503k.ada	c35a08b.ada
c34001f.ada	c34014a.ada	c35503l.ada	c360002.a
c34002a.ada	c34014c.ada	c35503o.ada	c36104a.ada
c34002c.ada	c34014e.ada	c35503p.ada	c36104b.ada
c34003a.ada	c34014g.ada	c35504a.ada	c36172a.ada
c34003c.ada	c34014h.ada	c35504b.ada	c36172b.ada
c34004a.ada	c34014n.ada	c35505c.ada	c36172c.ada
c34004c.ada	c34014p.ada	c35505e.ada	c36174a.ada
c34005a.ada	c34014r.ada	c35505f.ada	c36180a.ada
c34005c.ada	c34014t.ada	c35507a.ada	c36202c.ada
c34005d.ada	c34014u.ada	c35507b.ada	c36203a.ada
c34005f.ada	c34018a.ada	c35507c.ada	c36204a.ada

c36204b.ada	c37211e.ada	c38108d0.ada	c392008.a
c36204c.ada	c37213b.ada	c38108d1.ada	c392010.a
c36204d.ada	c37213d.ada	c3900010.a	c392011.a
c36205a.ada	c37213f.ada	c3900011.am	c392013.a
c36205b.ada	c37213h.ada	c390002.a	c392014.a
c36205c.ada	c37213j.ada	c390003.a	c392015.a
c36205d.ada	c37213k.ada	c390004.a	c392a01.a
c36205e.ada	c37213l.ada	c3900050.a	c392c05.a
c36205f.ada	c37215b.ada	c3900051.a	c392c07.a
c36205g.ada	c37215d.ada	c3900052.a	c392d01.a
c36205h.ada	c37215f.ada	c3900053.am	c392d02.a
c36205i.ada	c37215h.ada	c3900060.a	c392d03.a
c36205j.ada	c37217a.ada	c3900061.a	c393001.a
c36205k.ada	c37217b.ada	c3900062.a	c393007.a
c36205l.ada	c37217c.ada	c3900063.am	c393008.a
c36301a.ada	c37304a.ada	c390007.a	c393009.a
c36301b.ada	c37305a.ada	c390010.a	c393010.a
c36302a.ada	c37306a.ada	c390011.a	c393011.a
c36304a.ada	c37309a.ada	c390012.a	c393012.a
c36305a.ada	c37310a.ada	c39006a.ada	c393013.a
c37002a.ada	c37312a.ada	c39006b.ada	c393a02.a
c37003a.ada	c37402a.ada	c39006c0.ada	c393a03.a
c37003b.ada	c37403a.ada	c39006c1.ada	c393a05.a
c37005a.ada	c37404a.ada	c39006d.ada	c393a06.a
c37006a.ada	c37404b.ada	c39006e.ada	c393b12.a
c37008a.ada	c37405a.ada	c39006f0.ada	c393b13.a
c37008b.ada	c37411a.ada	c39006f1.ada	c393b14.a
c37009a.ada	c380001.a	c39006f2.ada	c394001.a
c37010a.ada	c380002.a	c39006f3.ada	c394002.a
c37010b.ada	c380003.a	c39006g.ada	c394003.a
c371001.a	c380004.a	c39007a.ada	c3a0001.a
c371002.a	c38002a.ada	c39007b.ada	c3a0002.a
c371003.a	c38002b.ada	c39008a.ada	c3a0003.a
c37102b.ada	c38005a.ada	c39008b.ada	c3a0004.a
c37103a.ada	c38005b.ada	c39008c.ada	c3a0005.a
c37105a.ada	c38005c.ada	c390a010.a	c3a0006.a
c37107a.ada	c38102a.ada	c390a011.am	c3a0007.a
c37108b.ada	c38102b.ada	c390a020.a	c3a0008.a
c37206a.ada	c38102c.ada	c390a021.a	c3a0009.a
c37207a.ada	c38102d.ada	c390a022.am	c3a0010.a
c37208a.ada	c38102e.ada	c390a030.a	c3a0011.a
c37208b.ada	c38104a.ada	c390a031.am	c3a00120.a
c37209a.ada	c38107a.ada	c391001.a	c3a00121.a
c37209b.ada	c38107b.ada	c391002.a	c3a00122.am
c37210a.ada	c38108a.ada	c391003.a	c3a0013.a
c37211a.ada	c38108b.ada	c392002.a	c3a0014.a
c37211b.ada	c38108c0.ada	c392003.a	c3a0015.a
c37211c.ada	c38108c1.ada	c392004.a	c3a0016.a
c37211d.ada	c38108c2.ada	c392005.a	c3a0017.a

c3a0018.a	c41303c.ada	c43105b.ada	c433001.a
c3a0019.a	c41303e.ada	c43106a.ada	c433002.a
c3a0020.a	c41303f.ada	c43107a.ada	c433003.a
c3a0021.a	c41303g.ada	c43108a.ada	c433004.a
c3a0022.a	c41303i.ada	c431a01.a	c433005.a
c3a0023.a	c41303j.ada	c431a02.a	c433006.a
c3a0024.a	c41303k.ada	c431a03.a	c433a01.a
c3a0025.a	c41303m.ada	c432001.a	c433a02.a
c3a0026.a	c41303n.ada	c432002.a	c433a03.a
c3a0027.a	c41303o.ada	c432003.a	c433a04.a
c3a0028.a	c41303q.ada	c432004.a	c44003d.ada
c3a0029.a	c41303r.ada	c432005.a	c44003f.ada
c3a1001.a	c41303s.ada	c43204a.ada	c44003g.ada
c3a1002.a	c41303u.ada	c43204c.ada	c450001.a
c3a10030.a	c41303v.ada	c43204e.ada	c45112a.ada
c3a10031.a	c41303w.ada	c43204f.ada	c45112b.ada
c3a10032.am	c41304a.ada	c43204g.ada	c45113a.ada
c3a10040.a	c41304b.ada	c43204h.ada	c45114b.ada
c3a10041.a	c41306b.ada	c43204i.ada	c452001.a
c3a10042.am	c41306c.ada	c43205a.ada	c45201a.ada
c3a2001.a	c41307d.ada	c43205b.ada	c45201b.ada
c3a2002.a	c41309a.ada	c43205c.ada	c45202b.ada
c3a2003.a	c41320a.ada	c43205d.ada	c45210a.ada
c3a2004.a	c41321a.ada	c43205e.ada	c45211a.ada
c3a2a01.a	c41322a.ada	c43205g.ada	c45220a.ada
c3a2a02.a	c41323a.ada	c43205h.ada	c45220b.ada
c410001.a	c41324a.ada	c43205i.ada	c45220c.ada
c41101d.ada	c41325a.ada	c43205j.ada	c45220d.ada
c41103a.ada	c41326a.ada	c43205k.ada	c45220e.ada
c41103b.ada	c41327a.ada	c43206a.ada	c45220f.ada
c41104a.ada	c41328a.ada	c43207b.ada	c45231a.ada
c41105a.ada	c41401a.ada	c43207d.ada	c45231b.dep
c41107a.ada	c41402a.ada	c43208a.ada	c45231c.dep
c41201d.ada	c41404a.ada	c43208b.ada	c45231d.tst
c41203a.ada	c415001.a	c43209a.ada	c45232b.ada
c41203b.ada	c416a01.a	c43210a.ada	c45242b.ada
c41204a.ada	c416a02.a	c43211a.ada	c45251a.ada
c41205a.ada	c420001.a	c43212a.ada	c45252a.ada
c41206a.ada	c42006a.ada	c43212c.ada	c45252b.ada
c41207a.ada	c42007e.ada	c43214a.ada	c45253a.ada
c413001.a	c43003a.ada	c43214b.ada	c45262a.ada
c413002.a	c43004a.ada	c43214c.ada	c45262b.ada
c413003.a	c43004c.ada	c43214d.ada	c45262c.ada
c413004.a	c431001.a	c43214e.ada	c45262d.ada
c413005.a	c431002.a	c43214f.ada	c45264a.ada
c413006.a	c43103a.ada	c43215a.ada	c45264b.ada
c41301a.ada	c43103b.ada	c43215b.ada	c45264c.ada
c41303a.ada	c43104a.ada	c43222a.ada	c45265a.ada
c41303b.ada	c43105a.ada	c43224a.ada	c45271a.ada

c45272a.ada	c45531f.ada	c457001.a	c47008a.ada
c45273a.ada	c45531g.ada	c457002.a	c47009a.ada
c45274a.ada	c45531h.ada	c457003.a	c47009b.ada
c45274b.ada	c45531i.ada	c457004.a	c48004a.ada
c45274c.ada	c45531j.ada	c457005.a	c48004b.ada
c45281a.ada	c45531k.ada	c457006.a	c48004c.ada
c45282a.ada	c45531l.ada	c457007.a	c48004d.ada
c45282b.ada	c45531m.dep	c458001.a	c48004e.ada
c45291a.ada	c45531n.dep	c460001.a	c48004f.ada
c45303a.ada	c45531o.dep	c460002.a	c48005a.ada
c45304a.ada	c45531p.dep	c460004.a	c48005b.ada
c45304b.dep	c45532a.ada	c460005.a	c48006a.ada
c45304c.dep	c45532b.ada	c460006.a	c48006b.ada
c45322a.ada	c45532c.ada	c460007.a	c48007a.ada
c45323a.ada	c45532d.ada	c460008.a	c48007b.ada
c45331a.ada	c45532e.ada	c460009.a	c48007c.ada
c45342a.ada	c45532f.ada	c460010.a	c48008a.ada
c45343a.ada	c45532g.ada	c460011.a	c48008c.ada
c45344a.ada	c45532h.ada	c460012.a	c48009a.ada
c45345b.ada	c45532i.ada	c460013.a	c48009b.ada
c45347a.ada	c45532j.ada	c46011a.ada	c48009c.ada
c45347b.ada	c45532k.ada	c46013a.ada	c48009d.ada
c45347c.ada	c45532l.ada	c46014a.ada	c48009e.ada
c45347d.ada	c45532m.dep	c46021a.ada	c48009f.ada
c45411a.ada	c45532n.dep	c46024a.ada	c48009g.ada
c45411b.dep	c45532o.dep	c46031a.ada	c48009h.ada
c45411c.dep	c45532p.dep	c46032a.ada	c48009i.ada
c45411d.ada	c45534b.ada	c46033a.ada	c48009j.ada
c45413a.ada	c45536a.dep	c46041a.ada	c48010a.ada
c45431a.ada	c456001.a	c46042a.ada	c48011a.ada
c455001.a	c45611a.ada	c46043b.ada	c48012a.ada
c45502b.dep	c45611b.dep	c46044b.ada	c490001.a
c45502c.dep	c45611c.dep	c46051a.ada	c490002.a
c45503a.ada	c45613a.ada	c46051b.ada	c490003.a
c45503b.dep	c45613b.dep	c46051c.ada	c49020a.ada
c45503c.dep	c45613c.dep	c46052a.ada	c49021a.ada
c45504a.ada	c45614a.ada	c46053a.ada	c49022a.ada
c45504b.dep	c45614b.dep	c46054a.ada	c49022b.ada
c45504c.dep	c45614c.dep	c460a01.a	c49022c.ada
c45504d.ada	c45631a.ada	c460a02.a	c49023a.ada
c45504e.dep	c45631b.dep	c47002a.ada	c49024a.ada
c45504f.dep	c45631c.dep	c47002b.ada	c49025a.ada
c45505a.ada	c45632a.ada	c47002c.ada	c49026a.ada
c45523a.ada	c45632b.dep	c47002d.ada	c4a005b.ada
c45531a.ada	c45632c.dep	c47003a.ada	c4a006a.ada
c45531b.ada	c45651a.ada	c47004a.ada	c4a007a.tst
c45531c.ada	c45662a.ada	c47005a.ada	c4a010a.ada
c45531d.ada	c45662b.ada	c47006a.ada	c4a010b.ada
c45531e.ada	c45672a.ada	c47007a.ada	c4a011a.ada

c4a012b.ada	c53007a.ada	c58005b.ada	c64105a.ada
c4a013a.ada	c540001.a	c58005h.ada	c64105b.ada
c4a014a.ada	c540002.a	c58006a.ada	c64105c.ada
c51004a.ada	c540003.a	c58006b.ada	c64105d.ada
c52005a.ada	c54a03a.ada	c59002a.ada	c64106a.ada
c52005b.ada	c54a04a.ada	c59002b.ada	c64106b.ada
c52005c.ada	c54a07a.ada	c59002c.ada	c64106c.ada
c52005d.ada	c54a13a.ada	c61008a.ada	c64106d.ada
c52005e.ada	c54a13b.ada	c61009a.ada	c64107a.ada
c52005f.ada	c54a13c.ada	c61010a.ada	c64108a.ada
c52008a.ada	c54a13d.ada	c611001.a	c64109a.ada
c52008b.ada	c54a22a.ada	c620001.a	c64109b.ada
c52009a.ada	c54a23a.ada	c62002a.ada	c64109c.ada
c52009b.ada	c54a24a.ada	c62003b.ada	c64109d.ada
c52010a.ada	c54a24b.ada	c62004a.ada	c64109e.ada
c52011a.ada	c54a42a.ada	c62006a.ada	c64109f.ada
c52011b.ada	c54a42b.ada	c631001.a	c64109g.ada
c52101a.ada	c54a42c.ada	c640001.a	c64109h.ada
c52102a.ada	c54a42d.ada	c640002.a	c64109i.ada
c52102b.ada	c54a42e.ada	c64002b.ada	c64109j.ada
c52102c.ada	c54a42f.ada	c64004g.ada	c64109k.ada
c52102d.ada	c54a42g.ada	c64005a.ada	c64109l.ada
c52103a.ada	c550001.a	c64005b.ada	c64201b.ada
c52103b.ada	c552001.a	c64005c.ada	c64201c.ada
c52103c.ada	c552002.a	c64005d0.ada	c64202a.ada
c52103f.ada	c552a01.a	c64005da.ada	c650002.a
c52103g.ada	c552a02.a	c64005db.ada	c650003.a
c52103h.ada	c55b03a.ada	c64005dc.ada	c65003a.ada
c52103k.ada	c55b04a.ada	c641001.a	c65003b.ada
c52103l.ada	c55b05a.ada	c64103b.ada	c650a01.a
c52103m.ada	c55b06a.ada	c64103c.ada	c650a02.a
c52103p.ada	c55b06b.ada	c64103d.ada	c650b01.a
c52103q.ada	c55b07a.dep	c64103e.ada	c650b02.a
c52103r.ada	c55b07b.dep	c64103f.ada	c650b03.a
c52103x.ada	c55b10a.ada	c64104a.ada	c660001.a
c52104a.ada	c55b11a.ada	c64104b.ada	c66002a.ada
c52104b.ada	c55b11b.ada	c64104c.ada	c66002c.ada
c52104c.ada	c55b15a.ada	c64104d.ada	c66002d.ada
c52104f.ada	c55b16a.ada	c64104e.ada	c66002e.ada
c52104g.ada	c55c02a.ada	c64104f.ada	c66002f.ada
c52104h.ada	c55c02b.ada	c64104g.ada	c66002g.ada
c52104k.ada	c56002a.ada	c64104h.ada	c67002a.ada
c52104l.ada	c57003a.ada	c64104i.ada	c67002b.ada
c52104m.ada	c57004a.ada	c64104j.ada	c67002c.ada
c52104p.ada	c57004b.ada	c64104k.ada	c67002d.ada
c52104q.ada	c58004c.ada	c64104l.ada	c67002e.ada
c52104r.ada	c58004d.ada	c64104m.ada	c67003f.ada
c52104x.ada	c58004g.ada	c64104n.ada	c67005a.ada
c52104y.ada	c58005a.ada	c64104o.ada	c67005b.ada

c67005c.ada	c760007.a	c83f01a.ada	c854001.a
c67005d.ada	c760009.a	c83f01b.ada	c854002.a
c680001.a	c760010.a	c83f01c0.ada	c854003.a
c72001b.ada	c760011.a	c83f01c1.ada	c86003a.ada
c72002a.ada	c760012.a	c83f01c2.ada	c86004a.ada
c730001.a	c760013.a	c83f01d0.ada	c86004b0.ada
c730002.a	c760014.a	c83f01d1.ada	c86004b1.ada
c730003.a	c760015.a	c83f03a.ada	c86004b2.ada
c730004.a	c760a01.a	c83f03b.ada	c86004c0.ada
c73002a.ada	c761001.a	c83f03c0.ada	c86004c1.ada
c730a01.a	c761002.a	c83f03c1.ada	c86004c2.ada
c730a02.a	c761003.a	c83f03c2.ada	c86006i.ada
c731001.a	c761004.a	c83f03d0.ada	c86007a.ada
c7320010.a	c761005.a	c83f03d1.ada	c87a05a.ada
c7320011.a	c761006.a	c840001.a	c87a05b.ada
c7320012.am	c761007.a	c840002.a	c87b02a.ada
c732002.a	c761010.a	c84002a.ada	c87b02b.ada
c732a01.a	c761011.a	c84005a.ada	c87b03a.ada
c732a02.a	c761012.a	c84008a.ada	c87b04a.ada
c732b01.a	c761013.a	c84009a.ada	c87b04b.ada
c732b02.a	c83007a.ada	c85004b.ada	c87b04c.ada
c732c01.a	c83012d.ada	c85005a.ada	c87b05a.ada
c74004a.ada	c83022a.ada	c85005b.ada	c87b06a.ada
c74203a.ada	c83022g0.ada	c85005c.ada	c87b07a.ada
c74206a.ada	c83022g1.ada	c85005d.ada	c87b07b.ada
c74207b.ada	c83023a.ada	c85005e.ada	c87b07c.ada
c74208a.ada	c83024a.ada	c85005f.ada	c87b07d.ada
c74208b.ada	c83024e0.ada	c85005g.ada	c87b07e.ada
c74209a.ada	c83024e1.ada	c85006a.ada	c87b08a.ada
c74210a.ada	c83025a.ada	c85006b.ada	c87b09a.ada
c74211a.ada	c83025c.ada	c85006c.ada	c87b09c.ada
c74211b.ada	c83027a.ada	c85006d.ada	c87b10a.ada
c74302a.ada	c83027c.ada	c85006e.ada	c87b11a.ada
c74302b.ada	c83028a.ada	c85006f.ada	c87b11b.ada
c74305a.ada	c83029a.ada	c85006g.ada	c87b13a.ada
c74305b.ada	c83030a.ada	c85007a.ada	c87b14a.ada
c74306a.ada	c83030c.ada	c85007e.ada	c87b14b.ada
c74307a.ada	c83031a.ada	c85009a.ada	c87b14c.ada
c74401d.ada	c83031c.ada	c85011a.ada	c87b14d.ada
c74401e.ada	c83031e.ada	c85013a.ada	c87b15a.ada
c74401k.ada	c83032a.ada	c85014a.ada	c87b16a.ada
c74401q.ada	c83033a.ada	c85014b.ada	c87b17a.ada
c74402a.ada	c83051a.ada	c85014c.ada	c87b18a.ada
c74402b.ada	c831001.a	c85017a.ada	c87b18b.ada
c74406a.ada	c83b02a.ada	c85018a.ada	c87b19a.ada
c74407b.ada	c83b02b.ada	c85018b.ada	c87b23a.ada
c74409b.ada	c83e02a.ada	c85019a.ada	c87b24a.ada
c760001.a	c83e02b.ada	c851001.a	c87b24b.ada
c760002.a	c83e03a.ada	c851002.a	c87b26b.ada

c87b27a.ada	c93004c.ada	c94008b.ada	c95085g.ada
c87b28a.ada	c93004d.ada	c94008c.ada	c95085h.ada
c87b29a.ada	c93004f.ada	c94008d.ada	c95085i.ada
c87b30a.ada	c93005a.ada	c94010a.ada	c95085j.ada
c87b31a.ada	c93005b.ada	c94011a.ada	c95085k.ada
c87b32a.ada	c93005c.ada	c94020a.ada	c95085l.ada
c87b33a.ada	c93005d.ada	c940a03.a	c95085m.ada
c87b34a.ada	c93005e.ada	c950001.a	c95085n.ada
c87b34b.ada	c93005f.ada	c95008a.ada	c95085o.ada
c87b34c.ada	c93005g.ada	c95009a.ada	c95086a.ada
c87b35c.ada	c93005h.ada	c95010a.ada	c95086b.ada
c87b38a.ada	c93006a.ada	c95011a.ada	c95086c.ada
c87b39a.ada	c93007a.ada	c95012a.ada	c95086d.ada
c87b40a.ada	c93008a.ada	c95021a.ada	c95086e.ada
c87b41a.ada	c93008b.ada	c95022a.ada	c95086f.ada
c87b42a.ada	c940001.a	c95022b.ada	c95087a.ada
c87b43a.ada	c940002.a	c95033a.ada	c95087b.ada
c87b44a.ada	c940004.a	c95033b.ada	c95087c.ada
c87b45a.ada	c940005.a	c95034a.ada	c95087d.ada
c87b45c.ada	c940006.a	c95034b.ada	c95088a.ada
c87b47a.ada	c940007.a	c95035a.ada	c95089a.ada
c87b48a.ada	c940010.a	c95040a.ada	c95090a.ada
c87b48b.ada	c940011.a	c95040b.ada	c95092a.ada
c87b50a.ada	c940012.a	c95040c.ada	c95093a.ada
c87b54a.ada	c940013.a	c95040d.ada	c95095a.ada
c87b57a.ada	c940014.a	c95041a.ada	c95095b.ada
c87b62a.ada	c940015.a	c95065a.ada	c95095c.ada
c87b62b.ada	c940016.a	c95065b.ada	c95095d.ada
c87b62c.ada	c94001a.ada	c95065c.ada	c95095e.ada
c87b62d.tst	c94001b.ada	c95065d.ada	c951001.a
c910001.a	c94001c.ada	c95065e.ada	c951002.a
c910002.a	c94001e.ada	c95065f.ada	c953001.a
c910003.a	c94001f.ada	c95066a.ada	c953002.a
c91004b.ada	c94001g.ada	c95067a.ada	c953003.a
c91004c.ada	c94002a.ada	c95071a.ada	c954001.a
c91006a.ada	c94002b.ada	c95072a.ada	c954010.a
c91007a.ada	c94002d.ada	c95072b.ada	c954011.a
c920001.a	c94002e.ada	c95073a.ada	c954012.a
c92002a.ada	c94002f.ada	c95074c.ada	c954013.a
c92003a.ada	c94002g.ada	c95076a.ada	c954014.a
c92005a.ada	c94004a.ada	c95078a.ada	c954015.a
c92005b.ada	c94004b.ada	c95080b.ada	c954016.a
c92006a.ada	c94004c.ada	c95082g.ada	c954017.a
c930001.a	c94005a.ada	c95085a.ada	c954018.a
c93001a.ada	c94005b.ada	c95085b.ada	c954019.a
c93002a.ada	c94006a.ada	c95085c.ada	c954020.a
c93003a.ada	c94007a.ada	c95085d.ada	c954021.a
c93004a.ada	c94007b.ada	c95085e.ada	c954022.a
c93004b.ada	c94008a.ada	c95085f.ada	c954023.a

c954024.a	c97301c.ada	ca1011a1.ada	ca11011.a
c954025.a	c97301d.ada	ca1011a2.ada	ca11012.a
c954026.a	c97301e.ada	ca1011a3.ada	ca11013.a
c954027.a	c97302a.ada	ca1011a4.ada	ca11014.a
c954a01.a	c97303a.ada	ca1011a5.ada	ca11015.a
c954a02.a	c97303b.ada	ca1011a6.ada	ca11016.a
c954a03.a	c97303c.ada	ca1012a0.ada	ca11017.a
c960001.a	c97304a.ada	ca1012a1.ada	ca11018.a
c960002.a	c97304b.ada	ca1012a2.ada	ca11019.a
c960004.a	c97305a.ada	ca1012a3.ada	ca11020.a
c96001a.ada	c97305b.ada	ca1012a4.ada	ca11021.a
c96004a.ada	c97305c.ada	ca1012b0.ada	ca11022.a
c96005a.ada	c97305d.ada	ca1012b2.ada	ca110230.a
c96005b.tst	c97307a.ada	ca1012b4.ada	ca110231.a
c96005d.ada	c974001.a	ca1013a0.ada	ca110232.am
c96005f.ada	c974002.a	ca1013a1.ada	ca1102a0.ada
c96006a.ada	c974003.a	ca1013a2.ada	ca1102a1.ada
c96007a.ada	c974004.a	ca1013a3.ada	ca1102a2.ada
c96008a.ada	c974005.a	ca1013a4.ada	ca1106a.ada
c96008b.ada	c974006.a	ca1013a5.ada	ca1108a.ada
c97112a.ada	c974007.a	ca1013a6.ada	ca1108b.ada
c97113a.ada	c974008.a	ca1014a0.ada	ca11a01.a
c97114a.ada	c974009.a	ca1014a1.ada	ca11a02.a
c97115a.ada	c974010.a	ca1014a2.ada	ca11b01.a
c97116a.ada	c974011.a	ca1014a3.ada	ca11b02.a
c97117a.ada	c974012.a	ca1020e0.ada	ca11c01.a
c97117b.ada	c974013.a	ca1020e1.ada	ca11c02.a
c97117c.ada	c974014.a	ca1020e2.ada	ca11c03.a
c97118a.ada	c980001.a	ca1020e3.ada	ca11d010.a
c97120a.ada	c980002.a	ca1022a0.ada	ca11d011.a
c97120b.ada	c980003.a	ca1022a1.ada	ca11d012.a
c97201a.ada	c990001.a	ca1022a2.ada	ca11d013.am
c97201b.ada	c99005a.ada	ca1022a3.ada	ca11d02.a
c97201c.ada	c9a003a.ada	ca1022a4.ada	ca11d03.a
c97201d.ada	c9a004a.ada	ca1022a5.ada	ca120010.a
c97201e.ada	c9a007a.ada	ca1022a6.ada	ca120011.a
c97201g.ada	c9a009a.ada	ca11001.a	ca120012.am
c97201h.ada	c9a009c.ada	ca11002.a	ca12002.a
c97201x.ada	c9a009f.ada	ca11003.a	ca13001.a
c97202a.ada	c9a009g.ada	ca110040.a	ca13002.a
c97203a.ada	c9a009h.ada	ca110041.a	ca13003.a
c97203b.ada	c9a010a.ada	ca110042.am	ca13a01.a
c97203c.ada	c9a011a.ada	ca110050.a	ca13a02.a
c97204a.ada	c9a011b.ada	ca110051.am	ca140230.a
c97204b.ada	ca1003a.ada	ca11006.a	ca140231.a
c97205a.ada	ca1004a.ada	ca11007.a	ca140232.am
c97205b.ada	ca1005a.ada	ca11008.a	ca140233.a
c97301a.ada	ca1006a.ada	ca11009.a	ca140280.a
c97301b.ada	ca1011a0.ada	ca11010.a	ca140281.a

ca140282.a	ca5003a5.ada	cb40a021.am	cc3007b.ada
ca140283.am	ca5003a6.ada	cb40a030.a	cc3011a.ada
ca15003.a	ca5003b0.ada	cb40a031.am	cc3011d.ada
ca200020.a	ca5003b1.ada	cb40a04.a	cc3012a.ada
ca200021.a	ca5003b2.ada	cb41001.a	cc3015a.ada
ca200022.am	ca5003b3.ada	cb41002.a	cc3016b.ada
ca200030.a	ca5003b4.ada	cb41003.a	cc3016c.ada
ca200031.am	ca5003b5.ada	cb41004.a	cc3016f.ada
ca2001h0.ada	ca5004a.ada	cb5001a.ada	cc3016i.ada
ca2001h1.ada	ca5004b0.ada	cb5001b.ada	cc3017b.ada
ca2001h2.ada	ca5004b1.ada	cb5002a.ada	cc3019a.ada
ca2001h3.ada	ca5004b2.ada	cc1004a.ada	cc3019b0.ada
ca2002a0.ada	ca5006a.ada	cc1005b.ada	cc3019b1.ada
ca2002a1.ada	cb10002.a	cc1010a.ada	cc3019b2.ada
ca2002a2.ada	cb1001a.ada	cc1010b.ada	cc3019c0.ada
ca2003a0.ada	cb1004a.ada	cc1018a.ada	cc3019c1.ada
ca2003a1.ada	cb1005a.ada	cc1104c.ada	cc3019c2.ada
ca2004a0.ada	cb1010a.ada	cc1107b.ada	cc3106b.ada
ca2004a1.ada	cb1010c.ada	cc1111a.ada	cc3120a.ada
ca2004a2.ada	cb1010d.ada	cc1204a.ada	cc3120b.ada
ca2004a3.ada	cb20001.a	cc1207b.ada	cc3121a.ada
ca2004a4.ada	cb20003.a	cc1220a.ada	cc3123a.ada
ca2007a0.ada	cb20004.a	cc1221a.ada	cc3125a.ada
ca2007a1.ada	cb20005.a	cc1221b.ada	cc3125b.ada
ca2007a2.ada	cb20006.a	cc1221c.ada	cc3125c.ada
ca2007a3.ada	cb20007.a	cc1221d.ada	cc3125d.ada
ca2008a0.ada	cb2004a.ada	cc1222a.ada	cc3126a.ada
ca2008a1.ada	cb2005a.ada	cc1223a.ada	cc3127a.ada
ca2008a2.ada	cb2006a.ada	cc1224a.ada	cc3128a.ada
ca2009a.ada	cb2007a.ada	cc1225a.tst	cc3203a.ada
ca2009c0.ada	cb20a02.a	cc1226b.ada	cc3207b.ada
ca2009c1.ada	cb30001.a	cc1227a.ada	cc3220a.ada
ca2009d.ada	cb30002.a	cc1301a.ada	cc3221a.ada
ca2009f0.ada	cb3003a.ada	cc1302a.ada	cc3222a.ada
ca2009f1.ada	cb3003b.ada	cc1304a.ada	cc3223a.ada
ca2009f2.ada	cb3004a.ada	cc1304b.ada	cc3224a.ada
ca2011b.ada	cb40005.a	cc1307a.ada	cc3225a.ada
ca21001.a	cb4001a.ada	cc1307b.ada	cc3230a.ada
ca21002.a	cb4002a.ada	cc1308a.ada	cc3231a.ada
ca3011a0.ada	cb4003a.ada	cc1310a.ada	cc3232a.ada
ca3011a1.ada	cb4004a.ada	cc1311a.ada	cc3233a.ada
ca3011a2.ada	cb4005a.ada	cc1311b.ada	cc3234a.ada
ca3011a3.ada	cb4006a.ada	cc2002a.ada	cc3235a.ada
ca3011a4.ada	cb4007a.ada	cc30001.a	cc3236a.ada
ca5003a0.ada	cb4008a.ada	cc30002.a	cc3240a.ada
ca5003a1.ada	cb4009a.ada	cc30003.a	cc3305a.ada
ca5003a2.ada	cb4013a.ada	cc30004.a	cc3305b.ada
ca5003a3.ada	cb40a01.a	cc3004a.ada	cc3305c.ada
ca5003a4.ada	cb40a020.a	cc3007a.ada	cc3305d.ada

cc3601a.ada	cd1009g.ada	cd2a32a.ada	cd4051d.ada
cc3601c.ada	cd1009h.ada	cd2a32c.ada	cd5003a.ada
cc3602a.ada	cd1009i.ada	cd2a32e.ada	cd5003b.ada
cc3603a.ada	cd1009j.ada	cd2a32g.ada	cd5003c.ada
cc3605a.ada	cd1009k.tst	cd2a32i.ada	cd5003d.ada
cc3606a.ada	cd1009l.ada	cd2a32j.ada	cd5003e.ada
cc3606b.ada	cd1009m.ada	cd2a51a.ada	cd5003f.ada
cc3607b.ada	cd1009n.ada	cd2a53a.ada	cd5003g.ada
cc40001.a	cd1009o.ada	cd2a53e.ada	cd5003h.ada
cc50001.a	cd1009p.ada	cd2a83c.tst	cd5003i.ada
cc50a01.a	cd1009q.ada	cd2a91c.tst	cd5011a.ada
cc50a02.a	cd1009r.ada	cd2b11b.ada	cd5011c.ada
cc51001.a	cd1009s.ada	cd2b11d.ada	cd5011e.ada
cc51002.a	cd1009t.tst	cd2b11e.ada	cd5011g.ada
cc51003.a	cd1009u.tst	cd2b11f.ada	cd5011i.ada
cc51004.a	cd1009v.ada	cd2b16a.ada	cd5011k.ada
cc51006.a	cd1009w.ada	cd2c11a.tst	cd5011m.ada
cc51007.a	cd1009x.ada	cd2c11d.tst	cd5011q.ada
cc51008.a	cd1009y.ada	cd2d11a.ada	cd5011s.ada
cc51009.a	cd1009z.ada	cd2d13a.ada	cd5012a.ada
cc510100.a	cd1c03a.ada	cd30001.a	cd5012b.ada
cc510101.a	cd1c03b.ada	cd30002.a	cd5012e.ada
cc510102.a	cd1c03c.ada	cd30003.a	cd5012f.ada
cc510103.am	cd1c03e.tst	cd30004.a	cd5012i.ada
cc51a01.a	cd1c03f.ada	cd300050.am	cd5012m.ada
cc51b03.a	cd1c03g.ada	cd300051.c	cd5013a.ada
cc51d01.a	cd1c03h.ada	cd30006.a	cd5013c.ada
cc51d02.a	cd1c03i.ada	cd30007.a	cd5013e.ada
cc54001.a	cd1c04a.ada	cd30008.a	cd5013g.ada
cc54002.a	cd1c04d.ada	cd30009.a	cd5013i.ada
cc54003.a	cd1c04e.ada	cd3014a.ada	cd5013k.ada
cc54004.a	cd1c06a.tst	cd3014c.ada	cd5013m.ada
cc60001.a	cd20001.a	cd3014d.ada	cd5013o.ada
cc70001.a	cd2a21a.ada	cd3014f.ada	cd5014a.ada
cc70002.a	cd2a21c.ada	cd3015a.ada	cd5014c.ada
cc70003.a	cd2a21e.ada	cd3015c.ada	cd5014e.ada
cc70a01.a	cd2a22a.ada	cd3015e.ada	cd5014g.ada
cc70a02.a	cd2a22e.ada	cd3015f.ada	cd5014i.ada
cc70b01.a	cd2a22i.ada	cd3015g.ada	cd5014k.ada
cc70b02.a	cd2a22j.ada	cd3015h.ada	cd5014m.ada
cc70c01.a	cd2a23a.ada	cd3015i.ada	cd5014o.ada
cc70c02.a	cd2a23e.ada	cd3015k.ada	cd5014t.ada
cd10001.a	cd2a24a.ada	cd3021a.ada	cd5014v.ada
cd10002.a	cd2a24e.ada	cd40001.a	cd5014x.ada
cd1009a.ada	cd2a24i.ada	cd4031a.ada	cd5014y.ada
cd1009b.ada	cd2a24j.ada	cd4041a.tst	cd5014z.ada
cd1009d.ada	cd2a31a.ada	cd4051a.ada	cd70001.a
cd1009e.ada	cd2a31c.ada	cd4051b.ada	cd7002a.ada
cd1009f.ada	cd2a31e.ada	cd4051c.ada	cd7007b.ada

cd7101d.ada	ce2102s.ada	ce2204a.ada	ce3104c.ada
cd7101e.dep	ce2102t.ada	ce2204b.ada	ce3106a.ada
cd7101f.dep	ce2102u.ada	ce2204c.ada	ce3106b.ada
cd7101g.tst	ce2102v.ada	ce2204d.ada	ce3107a.tst
cd7103d.ada	ce2102w.ada	ce2205a.ada	ce3107b.ada
cd7202a.ada	ce2102x.ada	ce2206a.ada	ce3108a.ada
cd7204b.ada	ce2102y.ada	ce2208b.ada	ce3108b.ada
cd7204c.ada	ce2103a.tst	ce2401a.ada	ce3110a.ada
cd72a01.a	ce2103b.tst	ce2401b.ada	ce3112c.ada
cd72a02.a	ce2103c.ada	ce2401c.ada	ce3112d.ada
cd7305a.ada	ce2103d.ada	ce2401e.ada	ce3114a.ada
cd90001.a	ce2104a.ada	ce2401f.ada	ce3115a.ada
cd92001.a	ce2104b.ada	ce2401h.ada	ce3201a.ada
cda201a.ada	ce2104c.ada	ce2401i.ada	ce3202a.ada
cda201b.ada	ce2104d.ada	ce2401j.ada	ce3206a.ada
cda201c.ada	ce2106a.ada	ce2401k.ada	ce3207a.ada
cda201e.ada	ce2106b.ada	ce2401l.ada	ce3301a.ada
cdb0001.a	ce2108e.ada	ce2402a.ada	ce3302a.ada
cdb0002.a	ce2108f.ada	ce2403a.tst	ce3303a.ada
cdb0a01.a	ce2108g.ada	ce2404a.ada	ce3304a.tst
cdb0a02.a	ce2108h.ada	ce2404b.ada	ce3305a.ada
cdb3a01.a	ce2109a.ada	ce2405b.ada	ce3306a.ada
cdb4001.a	ce2109b.ada	ce2406a.ada	ce3401a.ada
cdd1001.a	ce2109c.ada	ce2407a.ada	ce3402a.ada
cdd2001.a	ce2110a.ada	ce2407b.ada	ce3402c.ada
cdd2a01.a	ce2110c.ada	ce2408a.ada	ce3402d.ada
cdd2a02.a	ce2111a.ada	ce2408b.ada	ce3402e.ada
cdd2a03.a	ce2111b.ada	ce2409a.ada	ce3403a.ada
cde0001.a	ce2111c.ada	ce2409b.ada	ce3403b.ada
cde0002.a	ce2111e.ada	ce2410a.ada	ce3403c.ada
cde0003.a	ce2111f.ada	ce2410b.ada	ce3403d.ada
ce2102a.ada	ce2111g.ada	ce2411a.ada	ce3403e.ada
ce2102b.ada	ce2111i.ada	ce3002b.tst	ce3403f.ada
ce2102c.tst	ce2201a.ada	ce3002c.tst	ce3404a.ada
ce2102d.ada	ce2201b.ada	ce3002d.ada	ce3404b.ada
ce2102e.ada	ce2201c.ada	ce3002f.ada	ce3404c.ada
ce2102f.ada	ce2201d.dep	ce3102a.ada	ce3404d.ada
ce2102g.ada	ce2201e.dep	ce3102b.tst	ce3405a.ada
ce2102h.tst	ce2201f.ada	ce3102d.ada	ce3405c.ada
ce2102i.ada	ce2201g.ada	ce3102e.ada	ce3405d.ada
ce2102j.ada	ce2201h.ada	ce3102f.ada	ce3406a.ada
ce2102k.ada	ce2201i.ada	ce3102g.ada	ce3406b.ada
ce2102l.ada	ce2201j.ada	ce3102h.ada	ce3406c.ada
ce2102m.ada	ce2201k.ada	ce3102i.ada	ce3406d.ada
ce2102n.ada	ce2201l.ada	ce3102j.ada	ce3407a.ada
ce2102o.ada	ce2201m.ada	ce3102k.ada	ce3407b.ada
ce2102p.ada	ce2201n.ada	ce3103a.ada	ce3407c.ada
ce2102q.ada	ce2202a.ada	ce3104a.ada	ce3408a.ada
ce2102r.ada	ce2203a.tst	ce3104b.ada	ce3408b.ada

ce3408c.ada	ce3706d.ada	ce3907a.ada	cxa5013.a
ce3409a.ada	ce3706f.ada	ce3908a.ada	cxa5015.a
ce3409b.ada	ce3706g.ada	cxa3001.a	cxa5016.a
ce3409c.ada	ce3707a.ada	cxa3002.a	cxa5a01.a
ce3409d.ada	ce3708a.ada	cxa3003.a	cxa5a02.a
ce3409e.ada	ce3801a.ada	cxa3004.a	cxa5a03.a
ce3410a.ada	ce3801b.ada	cxa3005.a	cxa5a04.a
ce3410b.ada	ce3804a.ada	cxa3006.a	cxa5a05.a
ce3410c.ada	ce3804b.ada	cxa3007.a	cxa5a06.a
ce3410d.ada	ce3804c.ada	cxa3008.a	cxa5a07.a
ce3410e.ada	ce3804d.ada	cxa4001.a	cxa5a08.a
ce3411a.ada	ce3804e.ada	cxa4002.a	cxa5a09.a
ce3411c.ada	ce3804f.ada	cxa4003.a	cxa5a10.a
ce3412a.ada	ce3804g.ada	cxa4004.a	cxa8001.a
ce3413a.ada	ce3804h.ada	cxa4005.a	cxa8002.a
ce3413b.ada	ce3804i.ada	cxa4006.a	cxa8003.a
ce3413c.ada	ce3804j.ada	cxa4007.a	cxa9001.a
ce3414a.ada	ce3804m.ada	cxa4008.a	cxa9002.a
ce3601a.ada	ce3804o.ada	cxa4009.a	cxaa001.a
ce3602a.ada	ce3804p.ada	cxa4010.a	cxaa002.a
ce3602b.ada	ce3805a.ada	cxa4011.a	cxaa003.a
ce3602c.ada	ce3805b.ada	cxa4012.a	cxaa004.a
ce3602d.ada	ce3806a.ada	cxa4013.a	cxaa005.a
ce3603a.ada	ce3806b.ada	cxa4014.a	cxaa006.a
ce3604a.ada	ce3806c.ada	cxa4015.a	cxaa007.a
ce3604b.ada	ce3806d.ada	cxa4016.a	cxaa008.a
ce3605a.ada	ce3806e.ada	cxa4017.a	cxaa009.a
ce3605b.ada	ce3806f.ada	cxa4018.a	cxaa010.a
ce3605c.ada	ce3806g.ada	cxa4019.a	cxaa011.a
ce3605d.ada	ce3806h.ada	cxa4020.a	cxaa012.a
ce3605e.ada	ce3809a.ada	cxa4021.a	cxaa013.a
ce3606a.ada	ce3809b.ada	cxa4022.a	cxaa014.a
ce3606b.ada	ce3810a.ada	cxa4023.a	cxaa015.a
ce3701a.ada	ce3810b.ada	cxa4024.a	cxaa016.a
ce3704a.ada	ce3815a.ada	cxa4025.a	cxaa017.a
ce3704b.ada	ce3901a.ada	cxa4026.a	cxaa018.a
ce3704c.ada	ce3902b.ada	cxa4027.a	cxaa019.a
ce3704d.ada	ce3904a.ada	cxa4028.a	cxaa020.a
ce3704e.ada	ce3904b.ada	cxa4029.a	cxaa021.a
ce3704f.ada	ce3905a.ada	cxa4030.a	cxaa022.a
ce3704m.ada	ce3905b.ada	cxa4031.a	cxab001.a
ce3704n.ada	ce3905c.ada	cxa4032.a	cxab002.au
ce3704o.ada	ce3905l.ada	cxa4033.a	cxab003.au
ce3705a.ada	ce3906a.ada	cxa4034.a	cxab004.au
ce3705b.ada	ce3906b.ada	cxa4035.a	cxab005.au
ce3705c.ada	ce3906c.ada	cxa4036.a	cxac001.a
ce3705d.ada	ce3906d.ada	cxa4037.a	cxac002.a
ce3705e.ada	ce3906e.ada	cxa5011.a	cxac003.a
ce3706c.ada	ce3906f.ada	cxa5012.a	cxac004.a

cxac005.a	cxai035.a	cxb3022.a	la140042.a
cxac006.a	cxai036.a	cxb4001.a	la140050.a
cxac007.a	cxaia01.a	cxb4002.a	la140051.a
cxac008.a	cxaia02.a	cxb4003.a	la140052.am
cxaca01.a	cxaia03.a	cxb4004.a	la140053.a
cxaca02.a	cxaia04.a	cxb4005.a	la140060.a
cxacb01.a	cxaia05.a	cxb4006.a	la140061.a
cxacb02.a	cxaia06.a	cxb4007.a	la140062.am
cxacc01.a	cxaia07.a	cxb4008.a	la140063.a
cxaf001.a	cxaia08.a	cxb40090.cbl	la140070.a
cxag001.a	cxaia09.a	cxb40091.cbl	la140071.a
cxag002.a	cxaia10.a	cxb40092.cbl	la140072.am
cxah001.a	cxaia11.a	cxb40093.am	la140073.a
cxah002.a	cxaia12.a	cxb5001.a	la140080.a
cxah003.a	cxaia13.a	cxb5002.a	la140081.a
cxai001.a	cxaia14.a	cxb5003.a	la140082.am
cxai002.a	cxaj001.a	cxb50040.ftn	la140083.a
cxai003.a	cxb2001.a	cxb50041.ftn	la140090.a
cxai004.a	cxb2002.a	cxb50042.am	la140091.a
cxai005.a	cxb2003.a	cxb50050.ftn	la140092.am
cxai006.a	cxb3001.a	cxb50051.ftn	la140093.a
cxai007.a	cxb3002.a	cxb50052.am	la140100.a
cxai008.a	cxb3003.a	d4a002a.ada	la140101.a
cxai009.a	cxb30040.c	d4a002b.ada	la140102.am
cxai010.a	cxb30041.am	d4a004a.ada	la140103.a
cxai011.a	cxb3005.a	d4a004b.ada	la140110.a
cxai012.a	cxb30060.c	e28002b.ada	la140111.a
cxai013.a	cxb30061.am	e28005d.ada	la140112.am
cxai014.a	cxb3007.a	e52103y.ada	la140113.a
cxai015.a	cxb3008.a	eb4011a.ada	la140120.a
cxai016.a	cxb3009.a	eb4012a.ada	la140121.a
cxai017.a	cxb3010.a	eb4014a.ada	la140122.am
cxai018.a	cxb3011.a	ee3203a.ada	la140123.a
cxai019.a	cxb3012.a	ee3204a.ada	la140130.a
cxai020.a	cxb30130.c	ee3402b.ada	la140131.a
cxai021.a	cxb30131.c	ee3409f.ada	la140132.am
cxai022.a	cxb30132.am	ee3412c.ada	la140133.a
cxai023.a	cxb3014.a	la140010.a	la140140.a
cxai024.a	cxb3015.a	la140011.am	la140141.a
cxai025.a	cxb3016.a	la140012.a	la140142.am
cxai026.a	cxb30170.c	la140020.a	la140143.a
cxai027.a	cxb30171.a	la140021.am	la140150.a
cxai028.a	cxb30172.am	la140022.a	la140151.a
cxai029.a	cxb30180.c	la140030.a	la140152.am
cxai030.a	cxb30181.a	la140031.a	la140153.a
cxai031.a	cxb30182.am	la140032.am	la140160.a
cxai032.a	cxb3019.a	la140033.a	la140161.a
cxai033.a	cxb3020.a	la140040.a	la140162.am
cxai034.a	cxb3021.a	la140041.am	la140163.a

la140170.a	la140242.am	la5001a0.ada	la5008b0.ada
la140171.a	la140243.a	la5001a1.ada	la5008b1.ada
la140172.am	la140250.a	la5001a2.ada	la5008c0.ada
la140173.a	la140251.am	la5001a3.ada	la5008c1.ada
la140180.a	la140252.a	la5001a4.ada	la5008d0.ada
la140181.a	la140260.a	la5001a5.ada	la5008d1.ada
la140182.am	la140261.a	la5001a6.ada	la5008e0.ada
la140183.a	la140262.am	la5001a7.ada	la5008e1.ada
la140190.a	la140263.a	la5007a0.ada	la5008f0.ada
la140191.a	la140270.a	la5007a1.ada	la5008f1.ada
la140192.am	la140271.a	la5007b0.ada	la5008g0.ada
la140193.a	la140272.am	la5007b1.ada	la5008g1.ada
la140200.a	la140273.a	la5007c0.ada	lc300010.a
la140201.a	la200010.a	la5007c1.ada	lc300011.a
la140202.am	la200011.a	la5007d0.ada	lc300012.am
la140203.a	la200012.am	la5007d1.ada	lc300020.a
la140210.a	la200020.a	la5007e0.ada	lc300021.a
la140211.am	la200021.a	la5007e1.ada	lc300022.am
la140212.a	la200022.a	la5007f0.ada	lc300030.a
la140220.a	la200023.am	la5007f1.ada	lc300031.a
la140221.am	la200030.a	la5007g0.ada	lc300032.am
la140222.a	la200031.a	la5007g1.ada	
la140240.a	la200032.a	la5008a0.ada	
la140241.a	la200033.am	la5008a1.ada	

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.

bxc3001.a	bxe2a01.a	bxh4012.a	cxc7005.a
bxc3002.a	bxe2a02.a	bxh4013.a	cxc7006.a
bxc5001.a	bxe2a03.a	cxc3001.a	cxd1001.a
bxc6001.a	bxe2a04.a	cxc3002.a	cxd1002.a
bxc6002.a	bxe2a05.a	cxc3003.a	cxd1003.a
bxc6003.a	bxe2a06.a	cxc3004.a	cxd1004.a
bxc6a01.a	bxe4001.a	cxc3005.a	cxd1005.a
bxc6a02.a	bxf1001.a	cxc3006.a	cxd1006.a
bxc6a03.a	bxh4001.a	cxc3007.a	cxd1007.a
bxc6a04.a	bxh4002.a	cxc3008.a	cxd1008.a
bxd1001.a	bxh4003.a	cxc3009.a	cxd2001.a
bxd1002.a	bxh4004.a	cxc3010.a	cxd2002.a
bxe2007.a	bxh4005.a	cxc6001.a	cxd2003.a
bxe2008.a	bxh4006.a	cxc6002.a	cxd2004.a
bxe2009.a	bxh4007.a	cxc6003.a	cxd2006.a
bxe2010.a	bxh4008.a	cxc7001.a	cxd2007.a
bxe2011.a	bxh4009.a	cxc7002.a	cxd2008.a
bxe2012.a	bxh4010.a	cxc7003.a	cxd3001.a
bxe2013.a	bxh4011.a	cxc7004.a	cxd3002.a

cxd3003.a	cxf2005.a	cxh1001.a	lxh40041.a
cxd4001.a	cxf2a01.a	cxh3001.a	lxh40042.a
cxd4002.a	cxf2a02.a	cxh3002.a	lxh40043.am
cxd4003.a	cxf3001.a	cxh30030.a	lxh40050.a
cxd4004.a	cxf3002.a	cxh30031.am	lxh40051.a
cxd4005.a	cxf3003.a	lxd70010.a	lxh40052.a
cxd4006.a	cxf3004.a	lxd70011.a	lxh40053.am
cxd4007.a	cxf3a01.a	lxd70012.am	lxh40060.a
cxd4008.a	cxf3a02.a	lxd70030.a	lxh40061.a
cxd4009.a	cxf3a03.a	lxd70031.a	lxh40062.a
cxd4010.a	cxf3a04.a	lxd70032.am	lxh40063.am
cxd5001.a	cxf3a05.a	lxd70040.a	lxh40070.a
cxd6001.a	cxf3a06.a	lxd70041.a	lxh40071.a
cxd6002.a	cxf3a07.a	lxd70042.am	lxh40072.a
cxd6003.a	cxf3a08.a	lxd70050.a	lxh40073.am
cxd8001.a	cxg1001.a	lxd70051.a	lxh40080.a
cxd8002.a	cxg1002.a	lxd70052.am	lxh40081.a
cxd8003.a	cxg1003.a	lxd70060.a	lxh40082.a
cxd9001.a	cxg1004.a	lxd70061.a	lxh40083.a
cxda001.a	cxg1005.a	lxd70062.am	lxh40084.am
cxda002.a	cxg2001.a	lxd70070.a	lxh40090.a
cxda003.a	cxg2002.a	lxd70071.a	lxh40091.a
cxda004.a	cxg2003.a	lxd70072.am	lxh40092.a
cxdb001.a	cxg2004.a	lxd70080.a	lxh40093.am
cxdb002.a	cxg2005.a	lxd70081.a	lxh40100.a
cxdb003.a	cxg2006.a	lxd70082.am	lxh40101.a
cxdb004.a	cxg2007.a	lxd70090.a	lxh40102.a
cxe1001.a	cxg2008.a	lxd70091.a	lxh40103.am
cxe2001.a	cxg2009.a	lxd70092.am	lxh40110.a
cxe2002.a	cxg2010.a	lxe30010.am	lxh40111.a
cxe4001.a	cxg2011.a	lxe30011.am	lxh40112.am
cxe4002.a	cxg2012.a	lxe30020.am	lxh40120.a
cxe4003.a	cxg2013.a	lxe30021.am	lxh40121.a
cxe4004.a	cxg2014.a	lxh40010.a	lxh40122.a
cxe4005.a	cxg2015.a	lxh40011.a	lxh40123.am
cxe4006.a	cxg2016.a	lxh40012.am	lxh40130.a
cxe5001.a	cxg2017.a	lxh40020.a	lxh40131.a
cxe5002.a	cxg2018.a	lxh40021.a	lxh40132.a
cxe5003.a	cxg2019.a	lxh40022.am	lxh40133.am
cxf1001.a	cxg2020.a	lxh40030.a	lxh40140.a
cxf2001.a	cxg2021.a	lxh40031.a	lxh40141.a
cxf2002.a	cxg2022.a	lxh40032.a	lxh40142.am
cxf2003.a	cxg2023.a	lxh40033.am	
cxf2004.a	cxg2024.a	lxh40040.a	

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.

f340a000.a	f460a00.a	fa11b00.a	fc70b00.a
f340a001.a	f552a00.a	fa11c00.a	fc70c00.a
f341a00.a	f650a00.a	fa11d00.a	fd72a00.a
f390a00.a	f650b00.a	fa13a00.a	fdb0a00.a
f392a00.a	f730a000.a	fa13b00.a	fdb3a00.a
f392c00.a	f730a001.a	fa21a00.a	fdd2a00.a
f392d00.a	f731a00.a	fb20a00.a	fxa5a00.a
f393a00.a	f732a00.a	fb40a00.a	fxaca00.a
f393b00.a	f732b00.a	fc50a00.a	fxacb00.a
f394a00.a	f732c00.a	fc51a00.a	fxacc00.a
f3a1a00.a	f750a00.a	fc51b00.a	fxaia00.a
f3a2a00.a	f760a00.a	fc51c00.a	fxc6a00.a
f416a00.a	f940a00.a	fc51d00.a	fxe2a00.a
f431a00.a	f954a00.a	fc54a00.a	fxf2a00.a
f433a00.a	fa11a00.a	fc70a00.a	fxf3a00.a

A.4 Documentation Files

This section lists all the files containing ACATS 4.1 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.

acats-ug.pdf	ug-02.htm	ug-428.htm	ug-53.htm
cont.gif	ug-1.htm	ug-43.htm	ug-531.htm
cover03a.pdf	ug-11.htm	ug-431.htm	ug-532.htm
cover03b.pdf	ug-12.htm	ug-432.htm	ug-54.htm
cover04a.pdf	ug-2.htm	ug-433.htm	ug-541.htm
cover04b.pdf	ug-3.htm	ug-44.htm	ug-542.htm
cover05.pdf	ug-31.htm	ug-45.htm	ug-55.htm
cover06a.pdf	ug-32.htm	ug-46.htm	ug-551.htm
cover06b.pdf	ug-321.htm	ug-47.htm	ug-552.htm
cover07.pdf	ug-322.htm	ug-48.htm	ug-553.htm
cover08.pdf	ug-323.htm	ug-5.htm	ug-554.htm
cover10.pdf	ug-4.htm	ug-51.htm	ug-555.htm
cover13.pdf	ug-41.htm	ug-511.htm	ug-5551.htm
cover-b.pdf	ug-411.htm	ug-512.htm	ug-5552.htm
coverkey.pdf	ug-412.htm	ug-5121.htm	ug-5553.htm
coversum.pdf	ug-413.htm	ug-5122.htm	ug-5554.htm
coverage.txt	ug-414.htm	ug-513.htm	ug-5555.htm
dirs.png	ug-415.htm	ug-5131.htm	ug-556.htm
find.gif	ug-416.htm	ug-5132.htm	ug-56.htm
index.gif	ug-42.htm	ug-5133.htm	ug-561.htm
leg-name.png	ug-421.htm	ug-5134.htm	ug-562.htm
lib.gif	ug-422.htm	ug-52.htm	ug-563.htm
mod-name.png	ug-423.htm	ug-521.htm	ug-564.htm
next.gif	ug-424.htm	ug-522.htm	ug-565.htm
prev.gif	ug-425.htm	ug-523.htm	ug-57.htm
testobj.txt	ug-426.htm	ug-524.htm	ug-571.htm
ug-01.htm	ug-427.htm	ug-525.htm	ug-572.htm

ug-58.htm	ug-a53.htm	ug-d1.htm	ug-d23.htm
ug-6.htm	ug-a54.htm	ug-d11.htm	ug-d24.htm
ug-61.htm	ug-a55.htm	ug-d12.htm	ug-d25.htm
ug-611.htm	ug-a6.htm	ug-d13.htm	ug-d26.htm
ug-612.htm	ug-a7.htm	ug-d14.htm	ug-d27.htm
ug-613.htm	ug-a8.htm	ug-d15.htm	ug-d28.htm
ug-614.htm	ug-a9.htm	ug-d16.htm	ug-d29.htm
ug-615.htm	ug-aa.htm	ug-d17.htm	ug-d2a.htm
ug-62.htm	ug-ab.htm	ug-d18.htm	ug-d2b.htm
ug-621.htm	ug-ac.htm	ug-d19.htm	ug-d2c.htm
ug-622.htm	ug-ad.htm	ug-d1a.htm	ug-d2d.htm
ug-623.htm	ug-ae.htm	ug-d1b.htm	ug-d2e.htm
ug-63.htm	ug-af.htm	ug-d1c.htm	ug-d2f.htm
ug-631.htm	ug-b.htm	ug-d1d.htm	ug-d2g.htm
ug-632.htm	ug-b1.htm	ug-d1e.htm	ug-d2h.htm
ug-64.htm	ug-b2.htm	ug-d1f.htm	ug-d2i.htm
ug-65.htm	ug-b3.htm	ug-d1g.htm	ug-d2j.htm
ug-a.htm	ug-c.htm	ug-d1h.htm	ug-d2k.htm
ug-a1.htm	ug-c1.htm	ug-d1i.htm	ug-d2l.htm
ug-a2.htm	ug-c2.htm	ug-d1j.htm	ug-e.htm
ug-a3.htm	ug-c3.htm	ug-d1k.htm	ug-f.htm
ug-a4.htm	ug-c4.htm	ug-d1l.htm	ug-toc.htm
ug-a5.htm	ug-c41.htm	ug-d2.htm	ug-ttl.htm
ug-a51.htm	ug-c42.htm	ug-d21.htm	usage-1.png
ug-a52.htm	ug-d.htm	ug-d22.htm	usage-2.png

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 4.1 Files

acats41.lst

A.5.2 Support Units Referenced by Many Tests

checkfil.ada	impdefc.a	impdefh.a	tctouch.ada
enumchek.ada	impdefd.a	lencheck.ada	
fcndecl.ada	impdefe.a	report.a	
impdef.a	impdefg.a	spprt13s.tst	

A.5.3 Preprocessing Tools and Data

macrosub.ada	macro.dfs	tsttests.dat

A.5.4 Tests for Reporting Code

cz00004.a	cz1101a.ada	cz1102a.ada	cz1103a.ada

A.5.5 Test Grading Tools

grade.a	special.a	trace.a
grd_data.a	summary.a	tst_sum.a

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	cxc3008	cxe5002	la14006
bxc5001	cxc3010	cxe5003	la14007
bxh4001	cxd1004	cxg1002	la14008
bxh4002	cxd1005	cxg1005	la14009
bxh4003	cxd2001	cxg2002	la14010
bxh4004	cxd2002	cxg2003	la14011
bxh4005	cxd2003	cxg2004	la14012
bxh4006	cxd2004	cxg2006	la14013
bxh4007	cxd2006	cxg2007	la14014
bxh4008	cxd2007	cxg2008	la14015
bxh4009	cxd2008	cxg2009	la14016
bxh4010	cxd3001	cxg2010	la14017
bxh4011	cxd3002	cxg2011	la14018
bxh4012	cxd3003	cxg2012	la14019
bxh4013	cxd4001	cxg2013	la14020
c3a1003	cxd4003	cxg2014	la14021
c3a1004	cxd4004	cxg2015	la14022
ca11023	cxd4005	cxg2016	la14024
ca12001	cxd4006	cxg2017	la14025
cc51010	cxd4007	cxg2018	la14026
cd30005	cxd4008	cxg2019	la14027
cxb3004	cxd4009	cxg2020	la20001
cxb3006	cxd4010	cxg2021	lc30001
cxb3008	cxda003	cxg2022	lc30002
cxb3013	cxda004	cxg2023	lc30003
cxb3017	cxe1001	cxg2024	lxd7009
cxb3018	cxe2001	cxh1001	lxe3001
cxb4009	cxe2002	cxh3001	lxe3002
cxb5004	cxe4001	cxh3003	lxh4001
cxb5005	cxe4002	la14001	lxh4002
cxc3001	cxe4003	la14002	lxh4003
cxc3003	cxe4004	la14003	lxh4004
cxc3004	cxe4005	la14004	lxh4005
cxc3006	cxe4006	la14005	lxh4006

lxh4007	lxh4009	lxh4011	lxh4013
lxh4008	lxh4010	lxh4012	lxh4014

A.7 Test Files Added Since ACATS 4.0

The following test files are new in ACATS 4.1:

b3930110.a	b750a06.a	bxai001.a	c552002.a
b3930111.a	b750a08.a	bxai002.a	c640002.a
b3930112.a	b750a09.a	bxai003.a	c650a02.a
b3930113.a	b750a10.a	bxai004.a	c650b01.a
b3a1008.a	b750a11.a	bxai005.a	c650b02.a
b3a10090.a	b750a12.a	bxai006.a	c650b03.a
b3a10091.a	b750a13.a	bxai007.a	c7320010.a
b3a1010.a	b831004.a	bxai008.a	c7320011.a
b3a20170.a	b8310050.a	bxaia01.a	c7320012.am
b3a20171.a	b8310051.a	bxaia02.a	c732002.a
b3a20172.a	b8310052.a	bxaia03.a	c732a01.a
b3a20173.a	b8310053.a	bxaia04.a	c732a02.a
b3a20174.a	b8400020.a	bxb3001.a	c732b01.a
b415001.a	b8400021.a	bxb3002.a	c732b02.a
b415002.a	b8400022.a	bxb3003.a	c732c01.a
b416002.a	b8400023.a	bxb3004.a	c760014.a
b416a01.a	b8400024.a	c324003.a	c760015.a
b431005.a	b8400025.a	c324004.a	c831001.a
b431006.a	b840003.a	c324005.a	ca21002.a
b433003.a	b860001.a	c390012.a	cb30001.a
b457002.a	ba120170.a	c391003.a	cb30002.a
b457003.a	ba120171.a	c3a10030.a	cc510100.a
b457004.a	ba120172.a	c3a10031.a	cc510101.a
b457005.a	ba120173.a	c3a10032.am	cc510102.a
b457006.a	ba150030.a	c3a10040.a	cc510103.am
b457007.a	ba150031.a	c3a10041.a	cc60001.a
b480002.a	ba150032.a	c3a10042.am	cd30008.a
b480003.a	ba150033.a	c3a2004.a	cd30009.a
b551001.a	ba150034.a	c416a01.a	cdb3a01.a
b551002.a	ba150035.a	c416a02.a	cxa4037.a
b552001.a	ba150036.a	c431002.a	cxa5016.a
b552a04.a	ba150037.a	c433002.a	cxaa021.a
b552a05.a	ba150038.a	c433003.a	cxaa022.a
b640001.a	ba150039.a	c433004.a	cxab002.au
b650005.a	ba15003a.a	c433005.a	cxab003.au
b650006.a	ba15003b.am	c433006.a	cxab004.au
b730010.a	bc60001.a	c457005.a	cxab005.au
b732001.a	bc60002.a	c457006.a	cxac008.a
b732c01.a	bc60003.a	c457007.a	cxag001.a
b732c02.a	bc60004.a	c540003.a	cxag002.a
b750a04.a	bd11002.a	c550001.a	cxah001.a
b750a05.a	bdb3a01.a	c552001.a	cxah002.a

cxah003.a	cxb3021.a	f650b00.a	f732c00.a
cxb3019.a	cxb3022.a	f732a00.a	fdb3a00.a
cxb3020.a	f416a00.a	f732b00.a	

A.8 Documentation Files Added Since ACATS 4.0

The following documentation files are new in ACATS 4.1:

cover06a.pdf	ug-615.htm	ug-65.htm	ug-d1g.htm
cover06b.pdf	ug-62.htm	ug-a55.htm	ug-d1h.htm
cover-b.pdf	ug-621.htm	ug-ad.htm	ug-d1i.htm
ug-6.htm	ug-622.htm	ug-ae.htm	ug-d1j.htm
ug-61.htm	ug-623.htm	ug-af.htm	ug-d1k.htm
ug-611.htm	ug-63.htm	ug-d1c.htm	ug-d1l.htm
ug-612.htm	ug-631.htm	ug-d1d.htm	ug-d2j.htm
ug-613.htm	ug-632.htm	ug-d1e.htm	ug-d2k.htm
ug-614.htm	ug-64.htm	ug-d1f.htm	ug-d2l.htm

A.9 Support Files Added Since ACATS 4.0

The following support files are new in ACATS 4.1:

grade.a	report.a	summary.a	tst_sum.a
grd_data.a	special.a	trace.a	

A.10 Test Files Modified Since ACATS 4.0

The following test files have been modified from their ACATS 4.0 versions:

b730009.a	bdd2005.a	cxai027.a	cc30007.a
b750a01.a	c324001.a	cc30003.a	f552a00.a
b750a02.a	c324002.a	cc30004.a	
bdd2004.a	c415001.a	cd30006.a	

A.11 Support Files Modified Since ACATS 4.0

The following support file has been modified from its ACATS 4.0 version:

impdef.a

A.12 Documentation Files Modified Since ACATS 4.0

The following documentation files have been modified from their ACATS 4.0 versions:

acats-ug.pdf	cover04b.pdf	cover10.pdf	dirs.png
cover03a.pdf	cover05.pdf	cover13.pdf	mod-name.png
cover03b.pdf	cover07.pdf	coverkey.pdf	ug-01.htm
cover04a.pdf	cover08.pdf	coversum.pdf	ug-21.htm

ug-1.htm	ug-51.htm	ug-563.htm	ug-d11.htm
ug-11.htm	ug-511.htm	ug-564.htm	ug-d12.htm
ug-12.htm	ug-512.htm	ug-565.htm	ug-d13.htm
ug-2.htm	ug-5121.htm	ug-57.htm	ug-d14.htm
ug-3.htm	ug-5122.htm	ug-571.htm	ug-d15.htm
ug-31.htm	ug-513.htm	ug-572.htm	ug-d16.htm
ug-32.htm	ug-5131.htm	ug-58.htm	ug-d17.htm
ug-321.htm	ug-5132.htm	ug-a.htm	ug-d18.htm
ug-322.htm	ug-5133.htm	ug-a1.htm	ug-d19.htm
ug-323.htm	ug-5134.htm	ug-a2.htm	ug-d1a.htm
ug-4.htm	ug-52.htm	ug-a3.htm	ug-d1b.htm
ug-41.htm	ug-521.htm	ug-a4.htm	ug-d2.htm
ug-411.htm	ug-522.htm	ug-a5.htm	ug-d21.htm
ug-412.htm	ug-523.htm	ug-a51.htm	ug-d22.htm
ug-413.htm	ug-524.htm	ug-a52.htm	ug-d23.htm
ug-414.htm	ug-525.htm	ug-a53.htm	ug-d24.htm
ug-415.htm	ug-53.htm	ug-a54.htm	ug-d25.htm
ug-416.htm	ug-531.htm	ug-a6.htm	ug-d26.htm
ug-42.htm	ug-532.htm	ug-a7.htm	ug-d27.htm
ug-421.htm	ug-54.htm	ug-a8.htm	ug-d28.htm
ug-422.htm	ug-541.htm	ug-a9.htm	ug-d29.htm
ug-423.htm	ug-542.htm	ug-aa.htm	ug-d2a.htm
ug-424.htm	ug-55.htm	ug-ab.htm	ug-d2b.htm
ug-425.htm	ug-551.htm	ug-ac.htm	ug-d2c.htm
ug-426.htm	ug-552.htm	ug-b.htm	ug-d2d.htm
ug-427.htm	ug-553.htm	ug-b1.htm	ug-d2e.htm
ug-428.htm	ug-554.htm	ug-b2.htm	ug-d2f.htm
ug-43.htm	ug-555.htm	ug-b3.htm	ug-d2g.htm
ug-431.htm	ug-5551.htm	ug-c.htm	ug-d2h.htm
ug-432.htm	ug-5552.htm	ug-c1.htm	ug-d2i.htm
ug-433.htm	ug-5553.htm	ug-c2.htm	ug-e.htm
ug-44.htm	ug-5554.htm	ug-c3.htm	ug-f.htm
ug-45.htm	ug-5555.htm	ug-c4.htm	ug-toc.htm
ug-46.htm	ug-556.htm	ug-c41.htm	ug-ttl.htm
ug-47.htm	ug-56.htm	ug-c42.htm	
ug-48.htm	ug-561.htm	ug-d.htm	
ug-5.htm	ug-562.htm	ug-d1.htm	

A.13 Test Files Deleted Since ACATS 4.0

The following test files were present in ACATS 4.0 but do not appear in ACATS 4.1:

b43102a.ada	cd33001.a	cd33002.a

A.14 Documentation Files Deleted Since ACATS 4.0

The following documentation files were present in ACATS 4.0 but do not appear in ACATS 4.1:

cover06.pdf

A.15 Support Files Deleted Since ACATS 4.0

The following support files were present in ACATS 4.0 but do not appear in ACATS 4.1:

repbody.ada	repspec.ada

Annex B: Parameterization Files

In ACATS 4.1, 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 hierearchy 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 4.1 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.
-- $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 4.1 "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	CC1225A.TST	CE2203A.TST
B22001G.TST	BD8004A.TST	CD1009K.TST	CE2403A.TST
B22001I.TST	BD8004B.TST	CD1009T.TST	CE3002B.TST
B22001J.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 4.1 16-05-13 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 4.1.
   * 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:
                xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                xxxxxxxxxxxxxxxxxxxxxxxx.
   - 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.
   - CZ00004 <><><><><> 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 ****************************.

,.,. PASS_TEST ACATS 4.1 16-05-13 09:57:42
---- PASS_TEST CHECKING 'TEST' AND 'RESULT' FOR 'PASSED'.
   - PASS_TEST THIS LINE IS EXACTLY 'MAX_LEN' LONG. ...5...60....5...70.
   - PASS_TEST THIS COMMENT HAS A WORD THAT SPANS THE FOLD POINT. THIS
               COMMENT FITS EXACTLY ON TWO LINES. ..5...60....5...70.
   - 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 4.1 16-05-13 09:57:42
---- FAIL_TEST CHECKING 'FAILED' AND 'RESULT' FOR 'FAILED'.
   * FAIL_TEST 'RESULT' SHOULD NOW BE 'FAILED'.
**** FAIL_TEST FAILED ****************************.

,.,. NA_TEST ACATS 4.1 16-05-13 09:57:42
---- NA_TEST CHECKING 'NOT-APPLICABLE'.
   + NA_TEST 'RESULT' SHOULD NOW BE 'NOT-APPLICABLE'.
++++ NA_TEST NOT-APPLICABLE ++++++++++++++++++++.

,.,. FAIL_NA_TEST ACATS 4.1 16-05-13 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 4.1 16-05-13 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_TEST ACATS 4.1 16-05-13 09:57:42
---- SPEC_FAIL_TEST CHECKING 'SPEC_ACT', 'FAILED', 'SPEC_ACT'.
   ! SPEC_FAIL_TEST 'RESULT' BECOMES 'TENTATIVELY PASSED'.
   * SPEC_FAIL_TEST 'RESULT' BECOMES 'FAILED'.
   ! SPEC_FAIL_TEST CALLING 'SPECIAL_ACTION' DOESN'T CHANGE 'RESULT'.
**** SPEC_FAIL_TEST FAILED ****************************.

,.,. CZ1101A ACATS 4.1 16-05-13 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 4.1 16-05-13 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 4.1 16-05-13 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 END OF FILE ERROR.
   * CZ1103A FROM CHECK_FILE: END_OF_FILE 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 4.1 16-05-13 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 4.1 16-05-13 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 4.1 16-05-13 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 conformity assessment, 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.2.15 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	C45411B.DEP	C45611B.DEP	C55B07B.DEP
B52004E.DEP	C45502B.DEP	C45613B.DEP	CD7101E.DEP
B55B09D.DEP	C45503B.DEP	C45614B.DEP	
C45231B.DEP	C45504B.DEP	C45631B.DEP	
C45304B.DEP	C45504E.DEP	C45632B.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	C45411C.DEP	C45504F.DEP	C45631C.DEP
B55B09C.DEP	C45502C.DEP	C45611C.DEP	C45632C.DEP
C45231C.DEP	C45503C.DEP	C45613C.DEP	C55B07A.DEP
C45304C.DEP	C45504C.DEP	C45614C.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:

```
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 tests contained in the following files are not applicable:

CE2201D.DEP	CE2201E.DEP

D.1.11 Package Ada.Directories.Hierarchical_File_Names

If package Ada.Directories.Hierarchical_File_Names is not supported, then the test contained in the following file is not applicable:

CXAG002.A

D.1.12 Convention C

If convention C is not supported, then the tests contained in the following files are not applicable:

BXB3001.A	CXB30041.AM	CXB30182.AM	CXB3022.A
BXB3002.A	CXB30061.AM	CXB3019.A	
BXB3003.A	CXB30132.AM	CXB3020.A	
BXB3004.A	CXB30172.AM	CXB3021.A	

D.1.13 Convention COBOL

If convention COBOL is not supported, then the test contained in the following file is not applicable:

CXB40093.AM

D.1.14 Convention Fortran

If convention Fortran is not supported, then the tests contained in the following files are not applicable:

C552002.A	CXB50042.AM	CXB50052.AM

D.1.15 Package Interfaces.C

If package Interfaces.C is not supported, then the tests contained in the following files are not applicable:

BXB3001.A	BXB3004.A	CXB3003.A	CXB30061.AM
BXB3002.A	CXB3001.A	CXB30041.AM	CXB3007.A
BXB3003.A	CXB3002.A	CXB3005.A	CXB3008.A

CXB3009.A	CXB30132.AM	CXB30172.AM	CXB3021.A
CXB3010.A	CXB3014.A	CXB30182.AM	CXB3022.A
CXB3011.A	CXB3015.A	CXB3019.A	
CXB3012.A	CXB3016.A	CXB3020.A	

See also 5.5.5.2, “Foreign Language Interface Tests” for more information on processing these tests.

D.1.16 Package Interfaces.C.Strings

If package Interfaces.C.Strings is not supported, then the tests contained in the following files are not applicable:

CXB3002.A	CXB3010.A	CXB30132.AM	CXB30182.AM
CXB3008.A	CXB3011.A	CXB3014.A	
CXB3009.A	CXB3012.A	CXB3016.A	

D.1.17 Package Interfaces.C.Pointers

If package Interfaces.C.Pointers is not supported, then the tests contained in the following files are not applicable:

CXB3003.A	CXB3014.A	CXB3015.A	CXB3016.A

D.1.18 Package Interfaces.COBOL

If package Interfaces.COBOL is not supported, then the tests contained in the following files are not applicable:

CXB4001.A	CXB4004.A	CXB4007.A
CXB4002.A	CXB4005.A	CXB4008.A
CXB4003.A	CXB4006.A	CXB40093.AM

See also 5.5.5.2, “Foreign Language Interface Tests” for more information on processing these tests.

D.1.19 Package Interfaces.Fortran

If package Interfaces.Fortran is not supported, then the tests contained in the following files are not applicable:

CXB5001.A	CXB5003.A	CXB50052.AM
CXB5002.A	CXB50042.AM	

See also 5.5.5.2, “Foreign Language Interface Tests” for more information on processing these tests.

D.1.20 Unchecked Unions

If unchecked unions are not supported, then the tests contained in the following files are not applicable:

BXB3001.A	BXB3003.A	CXB3021.A
BXB3002.A	BXB3004.A	CXB3022.A

D.1.21 Special Handling Tests

Tests 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

```
type F is digits System.Max_Digits;
```

If `F'Machine_Overflows = False` and `2.0*F'Machine_Radix**F'Machine_EMax <= F'Base'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	CE3405C.ADA	CE3704A.ADA	CE3905A.ADA
CE3102A.ADA	CE3405D.ADA	CE3704B.ADA	CE3905B.ADA
CE3102B.TST	CE3406A.ADA	CE3704C.ADA	CE3905C.ADA
CE3102F.ADA	CE3406B.ADA	CE3704D.ADA	CE3905L.ADA
CE3102G.ADA	CE3406C.ADA	CE3704E.ADA	CE3906A.ADA
CE3102H.ADA	CE3406D.ADA	CE3704F.ADA	CE3906B.ADA
CE3102J.ADA	CE3407A.ADA	CE3704M.ADA	CE3906C.ADA
CE3102K.ADA	CE3407B.ADA	CE3704N.ADA	CE3906E.ADA
CE3103A.ADA	CE3407C.ADA	CE3704O.ADA	CE3906F.ADA
CE3104A.ADA	CE3408A.ADA	CE3705A.ADA	CXAA001.A
CE3104B.ADA	CE3408B.ADA	CE3705B.ADA	CXAA002.A
CE3104C.ADA	CE3408C.ADA	CE3705C.ADA	CXAA003.A
CE3106A.ADA	CE3409A.ADA	CE3705D.ADA	CXAA004.A
CE3106B.ADA	CE3409C.ADA	CE3705E.ADA	CXAA005.A
CE3107A.TST	CE3409D.ADA	CE3706D.ADA	CXAA006.A
CE3107B.ADA	CE3409E.ADA	CE3706F.ADA	CXAA007.A
CE3108A.ADA	CE3410A.ADA	CE3706G.ADA	CXAA008.A
CE3108B.ADA	CE3410C.ADA	CE3804A.ADA	CXAA009.A
CE3110A.ADA	CE3410D.ADA	CE3804B.ADA	CXAA010.A
CE3112C.ADA	CE3410E.ADA	CE3804C.ADA	CXAA011.A
CE3112D.ADA	CE3411A.ADA	CE3804D.ADA	CXAA012.A
CE3114A.ADA	CE3411C.ADA	CE3804E.ADA	CXAA013.A
CE3115A.ADA	CE3412A.ADA	CE3804F.ADA	CXAA014.A
CE3207A.ADA	CE3413A.ADA	CE3804G.ADA	CXAA015.A
CE3301A.ADA	CE3413B.ADA	CE3804H.ADA	CXAA016.A
CE3302A.ADA	CE3413C.ADA	CE3804I.ADA	CXAA017.A
CE3304A.TST	CE3414A.ADA	CE3804J.ADA	CXAA018.A
CE3305A.ADA	CE3602A.ADA	CE3804M.ADA	CXAA019.A
CE3401A.ADA	CE3602B.ADA	CE3804O.ADA	CXAA020.A
CE3402A.ADA	CE3602C.ADA	CE3804P.ADA	CXAA021.A
CE3402C.ADA	CE3602D.ADA	CE3805A.ADA	CXAA022.A
CE3402D.ADA	CE3603A.ADA	CE3805B.ADA	CXAG001.A
CE3403A.ADA	CE3604A.ADA	CE3806A.ADA	CXF3A06.A
CE3403B.ADA	CE3604B.ADA	CE3806B.ADA	CXG1003.A
CE3403C.ADA	CE3605A.ADA	CE3806D.ADA	EE3203A.ADA
CE3403E.ADA	CE3605B.ADA	CE3806E.ADA	EE3204A.ADA
CE3403F.ADA	CE3605C.ADA	CE3806G.ADA	EE3402B.ADA
CE3404B.ADA	CE3605D.ADA	CE3806H.ADA	EE3409F.ADA
CE3404C.ADA	CE3605E.ADA	CE3902B.ADA	EE3412C.ADA
CE3404D.ADA	CE3606A.ADA	CE3904A.ADA	
CE3405A.ADA	CE3606B.ADA	CE3904B.ADA	

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	EE3203A.ADA
CE3406D.ADA	CE3604A.ADA	CE3804F.ADA	EE3204A.ADA
CE3407A.ADA	CE3604B.ADA	CE3804G.ADA	EE3412C.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	CE3302A.ADA	CE3408A.ADA	CE3605A.ADA
CE3102A.ADA	CE3304A.TST	CE3408B.ADA	CE3605B.ADA
CE3102B.TST	CE3305A.ADA	CE3408C.ADA	CE3605C.ADA
CE3102F.ADA	CE3401A.ADA	CE3409A.ADA	CE3605D.ADA
CE3102G.ADA	CE3402A.ADA	CE3409C.ADA	CE3605E.ADA
CE3102H.ADA	CE3402C.ADA	CE3409D.ADA	CE3606A.ADA
CE3102J.ADA	CE3402D.ADA	CE3409E.ADA	CE3606B.ADA
CE3102K.ADA	CE3403A.ADA	CE3410A.ADA	CE3704A.ADA
CE3103A.ADA	CE3403B.ADA	CE3410C.ADA	CE3704B.ADA
CE3104A.ADA	CE3403C.ADA	CE3410D.ADA	CE3704C.ADA
CE3104B.ADA	CE3403E.ADA	CE3410E.ADA	CE3704D.ADA
CE3104C.ADA	CE3403F.ADA	CE3411A.ADA	CE3704E.ADA
CE3106A.ADA	CE3404B.ADA	CE3411C.ADA	CE3704F.ADA
CE3106B.ADA	CE3404C.ADA	CE3412A.ADA	CE3704M.ADA
CE3107A.TST	CE3404D.ADA	CE3413A.ADA	CE3704N.ADA
CE3107B.ADA	CE3405A.ADA	CE3413B.ADA	CE3704O.ADA
CE3108A.ADA	CE3405C.ADA	CE3413C.ADA	CE3705A.ADA
CE3108B.ADA	CE3405D.ADA	CE3414A.ADA	CE3705B.ADA
CE3110A.ADA	CE3406A.ADA	CE3602A.ADA	CE3705C.ADA
CE3112C.ADA	CE3406B.ADA	CE3602B.ADA	CE3705D.ADA
CE3112D.ADA	CE3406C.ADA	CE3602C.ADA	CE3705E.ADA
CE3114A.ADA	CE3406D.ADA	CE3602D.ADA	CE3706D.ADA
CE3115A.ADA	CE3407A.ADA	CE3603A.ADA	CE3706F.ADA
CE3207A.ADA	CE3407B.ADA	CE3604A.ADA	CE3706G.ADA
CE3301A.ADA	CE3407C.ADA	CE3604B.ADA	CE3804A.ADA

CE3804B.ADA	CE3806A.ADA	CE3906B.ADA	CXAA018.A
CE3804C.ADA	CE3806B.ADA	CE3906C.ADA	CXAA019.A
CE3804D.ADA	CE3806D.ADA	CE3906E.ADA	CXAA020.A
CE3804E.ADA	CE3806E.ADA	CE3906F.ADA	CXAA021.A
CE3804F.ADA	CE3806G.ADA	CXAA003.A	CXAA022.A
CE3804G.ADA	CE3806H.ADA	CXAA004.A	CXAG001.A
CE3804H.ADA	CE3902B.ADA	CXAA005.A	CXF3A06.A
CE3804I.ADA	CE3904A.ADA	CXAA009.A	CXG1003.A
CE3804J.ADA	CE3904B.ADA	CXAA010.A	EE3203A.ADA
CE3804M.ADA	CE3905A.ADA	CXAA011.A	EE3204A.ADA
CE3804O.ADA	CE3905B.ADA	CXAA012.A	EE3402B.ADA
CE3804P.ADA	CE3905C.ADA	CXAA014.A	EE3409F.ADA
CE3805A.ADA	CE3905L.ADA	CXAA016.A	EE3412C.ADA
CE3805B.ADA	CE3906A.ADA	CXAA017.A	

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:

CXAA001.A	CXAA006.A	CXAA008.A	CXAA015.A
CXAA002.A	CXAA007.A	CXAA013.A	

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	CE2111I.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:

CE2102G.ADA	CE2111C.ADA	CE2201G.ADA	CE2201M.ADA
CE2102O.ADA	CE2111F.ADA	CE2201H.ADA	CE2201N.ADA
CE2104A.ADA	CE2201A.ADA	CE2201I.ADA	CE2204A.ADA
CE2104B.ADA	CE2201B.ADA	CE2201J.ADA	CE2205A.ADA
CE2108F.ADA	CE2201C.ADA	CE2201K.ADA	CE2206A.ADA
CE2111A.ADA	CE2201F.ADA	CE2201L.ADA	CE2208B.ADA

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	CE2111I.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	CE2111I.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	CE2111I.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	CE2104D.ADA	CE2404A.ADA	CE2410B.ADA
CE2102K.ADA	CE2106B.ADA	CE2404B.ADA	CE2411A.ADA
CE2102R.ADA	CE2108G.ADA	CE2405B.ADA	CXA8003.A
CE2102W.ADA	CE2108H.ADA	CE2407A.ADA	CXA9001.A
CE2102Y.ADA	CE2110C.ADA	CE2407B.ADA	CXA9002.A
CE2103B.TST	CE2111B.ADA	CE2408A.ADA	
CE2103D.ADA	CE2111E.ADA	CE2409B.ADA	
CE2104C.ADA	CE2403A.TST	CE2410A.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	CE2111B.ADA	CE2401F.ADA	CE2406A.ADA
CE2102K.ADA	CE2111E.ADA	CE2401H.ADA	CE2408B.ADA
CE2102S.ADA	CE2111G.ADA	CE2401I.ADA	CE2409A.ADA
CE2102T.ADA	CE2401A.ADA	CE2401J.ADA	CE2411A.ADA
CE2102U.ADA	CE2401B.ADA	CE2401K.ADA	
CE2102V.ADA	CE2401C.ADA	CE2401L.ADA	
CE2109B.ADA	CE2401E.ADA	CE2405B.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	CXAC004.A	CXAC008.A	CXACB02.A
CXAC001.A	CXAC005.A	CXACA01.A	CXACC01.A
CXAC002.A	CXAC006.A	CXACA02.A	
CXAC003.A	CXAC007.A	CXACB01.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	CXAB002.AU	CXAB004.AU

D.2.13 Wide Wide Text Files (Non-supported Features)

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

CXAB003.AU	CXAB005.AU

D.2.14 Directory Operations (Non-supported Features)

If `Use_Error` or `Name_Error` is raised by every attempt to read the current default directory, then the tests contained in the following files should report NOT_APPLICABLE:

CXAG001.A	CXAG002.A

If `Use_Error` or `Name_Error` is raised by every attempt to set the current default directory, then the tests contained in the following files should report NOT_APPLICABLE:

CXAG001.A

If `Use_Error` or `Name_Error` is raised by every attempt to create a directory, then the tests contained in the following files should report NOT_APPLICABLE:

CXAG001.A

D.2.15 File I/O Tests

If sequential, text, wide text, wide wide text, direct access, and stream files are not supported, along with directory operations on them, then the tests contained in the following files should eport NOT_APPLICABLE:

CE2102A.ADA	CE2108F.ADA	CE2203A.TST	CE2410A.ADA
CE2102B.ADA	CE2108G.ADA	CE2204A.ADA	CE2410B.ADA
CE2102C.TST	CE2108H.ADA	CE2204B.ADA	CE2411A.ADA
CE2102G.ADA	CE2109A.ADA	CE2204C.ADA	CE3102A.ADA
CE2102H.TST	CE2109B.ADA	CE2204D.ADA	CE3102B.TST
CE2102K.ADA	CE2109C.ADA	CE2205A.ADA	CE3102F.ADA
CE2102N.ADA	CE2110A.ADA	CE2206A.ADA	CE3102G.ADA
CE2102O.ADA	CE2110C.ADA	CE2208B.ADA	CE3102H.ADA
CE2102P.ADA	CE2111A.ADA	CE2401A.ADA	CE3102J.ADA
CE2102Q.ADA	CE2111B.ADA	CE2401B.ADA	CE3102K.ADA
CE2102R.ADA	CE2111C.ADA	CE2401C.ADA	CE3103A.ADA
CE2102S.ADA	CE2111E.ADA	CE2401E.ADA	CE3104A.ADA
CE2102T.ADA	CE2111F.ADA	CE2401F.ADA	CE3104B.ADA
CE2102U.ADA	CE2111G.ADA	CE2401H.ADA	CE3104C.ADA
CE2102V.ADA	CE2111I.ADA	CE2401I.ADA	CE3106A.ADA
CE2102W.ADA	CE2201A.ADA	CE2401J.ADA	CE3106B.ADA
CE2102X.ADA	CE2201B.ADA	CE2401K.ADA	CE3107A.TST
CE2102Y.ADA	CE2201C.ADA	CE2401L.ADA	CE3107B.ADA
CE2103A.TST	CE2201D.DEP	CE2403A.TST	CE3108A.ADA
CE2103B.TST	CE2201E.DEP	CE2404A.ADA	CE3108B.ADA
CE2103C.ADA	CE2201F.ADA	CE2404B.ADA	CE3110A.ADA
CE2103D.ADA	CE2201G.ADA	CE2405B.ADA	CE3112C.ADA
CE2104A.ADA	CE2201H.ADA	CE2406A.ADA	CE3112D.ADA
CE2104B.ADA	CE2201I.ADA	CE2407A.ADA	CE3114A.ADA
CE2104C.ADA	CE2201J.ADA	CE2407B.ADA	CE3115A.ADA
CE2104D.ADA	CE2201K.ADA	CE2408A.ADA	CE3207A.ADA
CE2106A.ADA	CE2201L.ADA	CE2408B.ADA	CE3301A.ADA
CE2106B.ADA	CE2201M.ADA	CE2409A.ADA	CE3302A.ADA
CE2108E.ADA	CE2201N.ADA	CE2409B.ADA	CE3304A.TST

CE3305A.ADA	CE3414A.ADA	CE3804I.ADA	CXAA011.A
CE3401A.ADA	CE3602A.ADA	CE3804J.ADA	CXAA012.A
CE3402A.ADA	CE3602B.ADA	CE3804M.ADA	CXAA013.A
CE3402C.ADA	CE3602C.ADA	CE3804O.ADA	CXAA014.A
CE3402D.ADA	CE3602D.ADA	CE3804P.ADA	CXAA015.A
CE3403A.ADA	CE3603A.ADA	CE3805A.ADA	CXAA016.A
CE3403B.ADA	CE3604A.ADA	CE3805B.ADA	CXAA017.A
CE3403C.ADA	CE3604B.ADA	CE3806A.ADA	CXAA018.A
CE3403E.ADA	CE3605A.ADA	CE3806B.ADA	CXAA019.A
CE3403F.ADA	CE3605B.ADA	CE3806D.ADA	CXAA020.A
CE3404B.ADA	CE3605C.ADA	CE3806E.ADA	CXAA021.A
CE3404C.ADA	CE3605D.ADA	CE3806G.ADA	CXAA022.A
CE3404D.ADA	CE3605E.ADA	CE3806H.ADA	CXAB001.A
CE3405A.ADA	CE3606A.ADA	CE3902B.ADA	CXAB002.AU
CE3405C.ADA	CE3606B.ADA	CE3904A.ADA	CXAB003.AU
CE3405D.ADA	CE3704A.ADA	CE3904B.ADA	CXAB004.AU
CE3406A.ADA	CE3704B.ADA	CE3905A.ADA	CXAB005.AU
CE3406B.ADA	CE3704C.ADA	CE3905B.ADA	CXAC001.A
CE3406C.ADA	CE3704D.ADA	CE3905C.ADA	CXAC002.A
CE3406D.ADA	CE3704E.ADA	CE3905L.ADA	CXAC003.A
CE3407A.ADA	CE3704F.ADA	CE3906A.ADA	CXAC004.A
CE3407B.ADA	CE3704M.ADA	CE3906B.ADA	CXAC005.A
CE3407C.ADA	CE3704N.ADA	CE3906C.ADA	CXAC006.A
CE3408A.ADA	CE3704O.ADA	CE3906E.ADA	CXAC007.A
CE3408B.ADA	CE3705A.ADA	CE3906F.ADA	CXAC008.A
CE3408C.ADA	CE3705B.ADA	CXA8001.A	CXACA01.A
CE3409A.ADA	CE3705C.ADA	CXA8002.A	CXACA02.A
CE3409C.ADA	CE3705D.ADA	CXA8003.A	CXACB01.A
CE3409D.ADA	CE3705E.ADA	CXA9001.A	CXACB02.A
CE3409E.ADA	CE3706D.ADA	CXA9002.A	CXACC01.A
CE3410A.ADA	CE3706F.ADA	CXAA001.A	CXAG001.A
CE3410C.ADA	CE3706G.ADA	CXAA002.A	CXAG002.A
CE3410D.ADA	CE3804A.ADA	CXAA003.A	CXF3A06.A
CE3410E.ADA	CE3804B.ADA	CXAA004.A	CXG1003.A
CE3411A.ADA	CE3804C.ADA	CXAA005.A	EE3203A.ADA
CE3411C.ADA	CE3804D.ADA	CXAA006.A	EE3204A.ADA
CE3412A.ADA	CE3804E.ADA	CXAA007.A	EE3402B.ADA
CE3413A.ADA	CE3804F.ADA	CXAA008.A	EE3409F.ADA
CE3413B.ADA	CE3804G.ADA	CXAA009.A	EE3412C.ADA
CE3413C.ADA	CE3804H.ADA	CXAA010.A	

D.2.16 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.2.17 Environment Variables

If the target execution environment does not support reading environment variables, then the tests contained in the following files should report NOT_APPLICABLE:

CXAH001.A	CXAH002.A	CXAH003.A

If the target execution environment does not support creating, writing, and deleting environment variables, then the tests contained in the following files should report NOT_APPLICABLE:

CXAH002.A	CXAH003.A

D.2.18 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.2.19 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.2.20 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:

CXD2001.A	CXD2004.A	CXD6001.A
CXD2002.A	CXD2007.A	CXD6002.A
CXD2003.A	CXD2008.A	CXD6003.A

D.2.21 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 definencies, 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 Resource Association. (Abbreviated **ARA**) The trade association that sponsors the Ada conformity assessment system.

Ada Standard documents. The document(s) that define the Ada programming language, currently the Ada Standard [Ada2012] along with its Technical Corrigendum [TC1-2012] 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. Clauses 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.

Event Trace. A list of interesting events that occurs during the compilation, binding/linking, and execution of one or more ACATS tests. An abstraction of the implementation-specific details used by the grading tool.

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.

Grading Tool. A tool to automate grading of ACATS tests. Uses an event trace and test summary to determine the pass or fail results of ACATS tests.

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.

Range indicator. A range indicator provides detailed information about the expected location of an error; tests using range indicators can be graded more accurately by the grading tool.

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.

Test Summary. A list of information about one or more ACATS tests, describing the test requirements to the grading tool.

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

[Ada2012]
ANSI/ISO/IEC 8652:2012, Information technology — Programming languages — Ada, December 2012 *(The Ada 2012 Reference Manual, essentially the same document, can be found online at http://www.adaic.org/ada-resources/standards/ada12/.)*

[Ada83] ANSI/MIL-STD-1815A-1983, ISO 8652:1987, FIPS 119 Reference Manual for the Ada Programming Language — superseded by ISO-8652:95) *(This can be found online at http://archive.adaic.com/standards/ada83.html.)*

[Ada95] ANSI/ISO/IEC 8652:1995, Information technology — Programming languages — Ada, January 1995 *(An unofficial version can be found online at http://archive.adaic.com/standards/ada95.html.)*

[Amend1] ISO/IEC 8652:1995/AMD 1:2007, Information technology — Programming languages — Ada — Amendment 1, March 2007. *(An unofficial version can be found at http://www.ada-auth.org/amendment.html. An unofficial document ("Ada 2005 Reference Manual") that merges [Amend1] with [Ada95] and [TC1] can be found at http://www.adaic.org/ada-resources/standards/ada05/.)*

[ISO99] ISO/IEC 18009:1999, Information technology — Programming languages — Ada: Conformity Assessment of a Language Processor, December 1999

[Pro31] Ada Resource Association: Operating Procedures for Ada Conformity Assessments Version 3.1, December 2013 *(This can be found online in various formats at http://www.ada-auth.org/info.html.)*

[TC1] ISO/IEC 8652:1995/COR 1:2001, Information technology — Programming languages — Ada — Technical Corrigendum 1, June 2001. *(An unofficial version can be found at http://www.ada-auth.org/grab-bag.html.)*

[TC1-2012]
ISO/IEC 8652:20012/COR 1:2016, Information technology — Programming languages — Ada — Technical Corrigendum 1, February 2016. *(An unofficial version can be found at http://www.ada-auth.org/corrigendum1-12.html.)*

Index