THE ADA CONFORMITY ASSESSMENT TEST SUITE (ACATS) VERSION 2.4 USER'S GUIDE

March 21, 2001

Prepared by:

Ada Conformity Assessment Authority
Randall L. Brukardt, Technical Agent
2318 Winnebago Street
Madison, WI 53704

Contents

1. Introduction

The Ada Conformity Assessment Test Suite (ACATS) is the official test method used to check conformity of an Ada implementation with the Ada programming language standard (ANSI/ISO/IEC 8652:1995). 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 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 to include maintenance changes and a few new tests.

This version of the ACATS is version 2.4. As with ACATS 2.3, this version was completely developed under the auspices of the ACAA. As with it predecessors, ACATS 2.4 contains test programs to check for conformity to new language features defined in [Ada95], as well as test programs to check for conformity to language features shared between Ada83 and Ada95. Subsequent maintenance or enhancement versions of the suite, if they are required, will be numbered 2.5, 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 [Pro01], 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 2.4 summarizes the changes between ACATS 2.3 and ACATS 2.4. Section 3 describes the configuration of the ACATS, including a description of the ACATS software and delivery files. Section 4 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 2.4 release.

Refer to Sections 1.1 and 4.7 for the definition of an acceptable result and the rules for grading ACATS 2.4 test program results. Section 4.8.2 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 Ada Information Clearinghouse (sponsored by the ARA). See *http://www.adaic.org*.

1.1 Definition of Terms

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

ACATS Modification List (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.

ACVC Implementer’s Guide (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 Conformity Assessment Authority (ACAA) : The part of the certification body that provides technical guidance for operations of the Ada certification system

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

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 (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 reference [Ada95].

Ada Resource Association (ARA) : The trade association that sponsors the Ada conformity assessment system.

Ada Validation Facility (AVF) : Former designation of an Ada Conformity Assessment Laboratory (which see).

Ada Validation Organization (AVO) : Organization that formerly performed the functions of the Ada Conformity Assessment Authority (which see).

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 (CPL) : A published list identifying all certified Ada implementations. The CPL is available on the Ada Information Clearinghouse Internet site (*www.adaic.org).*

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

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

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

Coverage matrix : A document containing an analysis of every paragraph of [Ada95]. Each paragraph has an indication of whether is contains a testable Ada95 requirement, whether it is upwardly compatible from Ada83, or whether is testable in the ACATS suite (e.g. it contains an example). Paragraphs that contain testable requirements also indicate what ACATS test(s) specifically examine features described in the paragraph.

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

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

Legacy Tests : Tests that were included in ACVC 1.12 that have been incorporated into later ACVC and ACATS versions. The vast majority of these tests check for language features that are upwardly compatible from Ada83 to Ada95. Some of these tests have been modified from the ACVC 1.12 versions to ensure that Ada95 rules are properly implemented in cases where there were extensions or incompatibilities from Ada83 to Ada95.

Specialized Needs Annex : 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 Validated Compilers List.

Test Objectives Document (TOD) : A document containing the test objectives used for new ACATS tests that focus on Ada95-specific features.

Validated Compilers List (VCL) : Former designation of the Certified Processors List (which see).

Validated Implementation : Informally used to mean Conforming Implementation (see).

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.

1.2 References

[Ada83] ANSI/MIL-STD-1815A-1983, ISO 8652:1987, FIPS 119 Reference Manual for the Ada Programming Language--superseded by ISO-8652:95)

[Ada95] ANSI/ISO/IEC 8652:1995, FIPS 119-1 The Reference Manual for the Ada Programming Language, February 1995

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

[Pro01] Ada Resource Association: Operating Procedures for Ada Conformity Assessments Version 3.0, April 2001

1.3 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 [Ada95]. 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 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.

2. Changes for ACATS 2.4

Version 2.4 of the ACATS primarily is a maintenance version. It contains a few new tests to check areas that have been reported to the ACAA as problems with existing implementations, and to check conformity with the forthcoming corrections to [Ada95].

In addition, some tests known to have problems have been modified. See Appendix A for lists of added, deleted and modified tests.

3. Configuration Information

3.1 Introduction

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 2.4 is a revision of ACATS 2.3, and has the essentially the same delivery structure. The support tools are essentially unchanged, except for updating header comments and version identification.

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

3.2 Structure

The ACATS 2.4 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. Table 1 summarizes the number of tests and files in the ACATS suite.

	Total	**Core Tests**	**SNA Tests**	**Foundations**	**Docs**	**Other**
Number of Files	4285	3959	247	44	12	23
Number of Tests	3677	3488	189	0	0	0

Table 1.

The delivery structure of the test suite is described in Section 3.8.

3.2.1 Physical Organization

Table 1 summarizes the number of files that compose ACATS 2.4. In addition to files containing test code proper, the ACATS 2.4 test suite includes various support files:

Others consists of

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

Note that the number of files containing test code is larger than the number of tests in the ACATS suite because several 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 Section3.4. The files are organized into distinct directories and subdirectories based on their function in the test suite. The directory organization is explained in Section 3.8.

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 4.2.2 provides a discussion of techniques to convert these files to a usable format.

3.2.2 Logical Organization

Table 1 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 [Ada95].

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 4.5.2. No annex test depends on the implementation of any other annex, except possibly in cases where one annex specifically depends on another in Ada95 (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 4.6.5.2.] 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 3.2.4.

Tests are named using conventions that provide (limited) information about the test. The test naming conventions are explained in Section 3.4. 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 3.3.

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 4.3.

3.2.3 Legacy Tests

Many tests check only language features that are common to Ada83 and Ada95. 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 Ada95 rules in cases where language rules changed from Ada83.

3.2.4 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 3.3). 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 3.4.

3.2.5 Special Core Tests

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

Annex C Requirements

Section 13 of [Ada95] includes implementation advice paragraphs. The ACATS does not require implementations to conform to those paragraphs unless they claim support for Annex C, Systems Programming (cf. C.2(2): “The implementation shall support at least the functionality defined by the recommended levels of support in Section 13.”)

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

CD10001	CD30005	CD40001
CD20001	CD33001	CD72A01
CD30001	CD33002	CD72A02
CD30002	CD30004	CD90001
CD30003		

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

Implementations that do not claim support for the appropriate Annexes 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.

3.2.6 Foreign Language Code

Several tests for Annex B features (and one Section 13 test) include files containing non-Ada code (Fortran, C, Cobol). These tests must be compiled, bound, and run by implementations that support foreign language interfaces to the respective non-Ada language. The foreign language code uses only the most basic language semantics and should be compilable by all Fortran, C, and Cobol compilers, respectively. In cases where a foreign language does not accept the code as provided, modifications are allowable. See Section 4.3.6.

Files that contain foreign code are identified by a special file extension. See Section 3.4.2.

The tests that include Fortran code are: CXB5004 and CXB5005

The tests that include C code are: CXB3004, CXB3006, CXB3013 and CD30005

The test that includes Cobol code is: CXB4009

3.3 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.

3.3.1 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.

3.3.2 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. Lines so marked are often, but not always, constructs that were errors in Ada83 but are correct in Ada95.

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 4.3.6 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.

3.3.3 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 4.4.2.

3.3.4 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.

3.3.5 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.

3.3.6 Class L

Class L tests check that all library unit dependencies 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).

3.3.7 Foundation Code

Files containing foundation code are named using the regular test name conventions (see Section 3.4). 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.

3.3.8 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.

3.4 Naming Convention

This section describes the naming conventions used in ACATS 2.4, 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 2.4. Legacy tests use the naming conventions of early ACVC versions. Tests new since ACVC 1.12 use the new convention. The conventions are consistently distinguishable at the 7th character of the name: legacy names have a letter in the 7th position, whereas newer names have a digit.

3.4.1 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		
1	Letter	Test class (cf. Section 3.3)
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 in Figure 1.

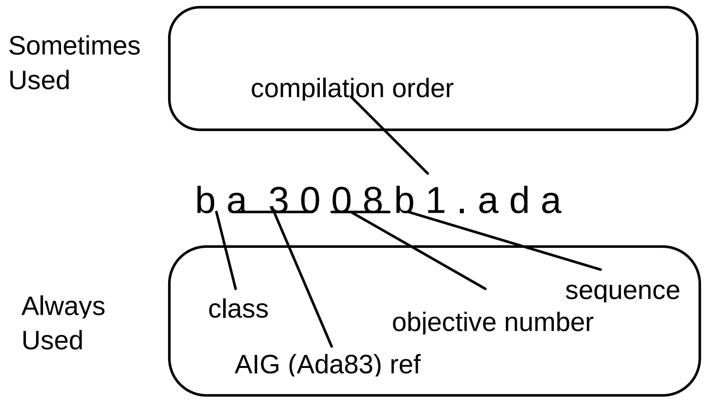

Figure 1. Legacy File Name Convention

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 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	B63009C3	B83004C2	B83E01F0
AD7001D0	B73004B0	B83004D0	B86001A1
B38103C3	B83003B0	B83024F0	B95020B2
B38103E0	B83004B0	B83E01E0	BA1001A0

BA1010A0	BA1101B0	C83F01D0	CA5003A6
BA1010B0	BA1101C2	C83F03C2	CA5003B5
BA1010C0	BA1109A2	C83F03D0	CA5004B1
BA1010D0	BA1110A1	C86004B2	CC3019B2
BA1010E0	BA2001F0	C86004C2	CC3019C2
BA1010F0	BA2003B0	CA1011A6	LA5001A7
BA1010G0	BA2011A1	CA1012A4	LA5007A1
BA1010H0	BA3001A0	CA1012B4	LA5007B1
BA1010I0	BA3001B0	CA1013A6	LA5007C1
BA1010J0	BA3001C0	CA1014A0	LA5007D1
BA1010K0	BA3001E0	CA1020E3	LA5007E1
BA1010L0	BA3001F0	CA1022A6	LA5007F1
BA1010M0	BA3006A6	CA1102A2	LA5007G1
BA1010N0	BA3006B4	CA2001H3	LA5008A1
BA1010P0	C38108C1	CA2002A0	LA5008B1
BA1010Q0	C38108D0	CA2003A0	LA5008C1
BA1011B0	C39006C0	CA2004A0	LA5008D1
BA1011C0	C39006F3	CA2007A0	LA5008E1
BA1020A0	C64005D0	CA2008A0	LA5008F1
BA1020B6	C83022G0	CA2009C0	LA5008G1
BA1020C0	C83024E1	CA2009F0	
BA1020F2	C83F01C2	CA3011A4	

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 4.3.2). 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.

Tests developed since ACVC 1.12 use different file name extensions.

Note that legacy tests have **not** been renamed for ACATS 2.4. Since [Ada95] includes some organizational differences from [Ada83], this means that the name of a legacy test sometimes will not correspond to the clause of [Ada95] in which the tested feature is described.

3.4.2 ACATS 2.4 Naming

The name of an Ada95 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		
1	Letter	Test class; foundations are marked 'F'
2	Alphanumeric	If other than an 'x', the section of [Ada95] describing the feature under test. An 'x' indicates that the test includes one or more features from an annex of [Ada95]
3	Alpha-numeric	Core clause or annex letter identifier (either core or Specialized Needs Annex)
4	Hexadecimal	Sub-clause (if a core test), or clause (if an annex test)
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.

This convention is illustrated in Figure 2.

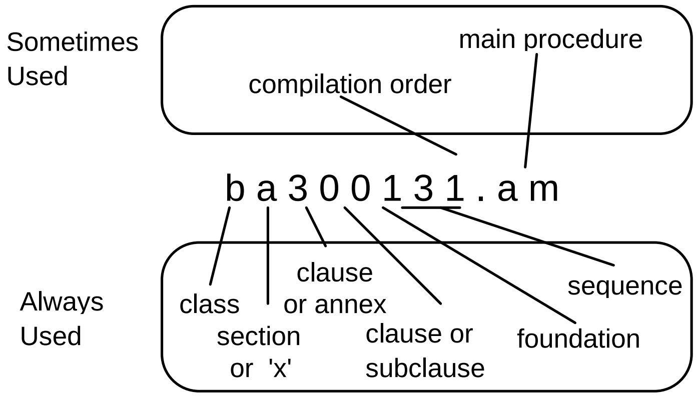

Figure 2. Naming convention in ACATS 2.4

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.
`.aw`	A file that has "code" that is not quite Ada; it contains one or more designated strings that must be replaced by a character from the upper half of ISO8850-1 (Latin-1) (see Section 4.3.3). The resulting test must be compiled and run as all other class C tests.
`.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.

3.4.3 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 program is not indicated (see the table in section 3.4.1 for files containing main subprograms). For newer tests, the extension ".am" indicates the file with the main program.

All tests apply the convention of naming the main subprogram the same as the file (excluding the file extension) plus the letter 'm' (for legacy tests only). For example, the legacy test, C39006F, is contained in four files, named `c39006f0.ada`, `c39006f1.ada`, `c39006f2.ada`, and `c39006f3.ada`. The main sub-program 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".

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.

3.5 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 all 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.

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

All tests have the line immediately after the disclaimer marked "--*". The newer 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 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. The file containing the main program has the complete prologue; the other, related files have those sections that apply to files (TEST FILES, CHANGE HISTORY) and refer to the main file for the other sections.

3.6 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 sub-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). Tests new to ACATS 2.x 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. Newer 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 the newer 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.

The newer 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".

3.7 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. 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 in 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:". 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.

3.8 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. See Fig. 3.

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.

Figure 3 sketches this scheme, but does not show complete detail. A list of all subdirectories is included in Section 4.2.2.

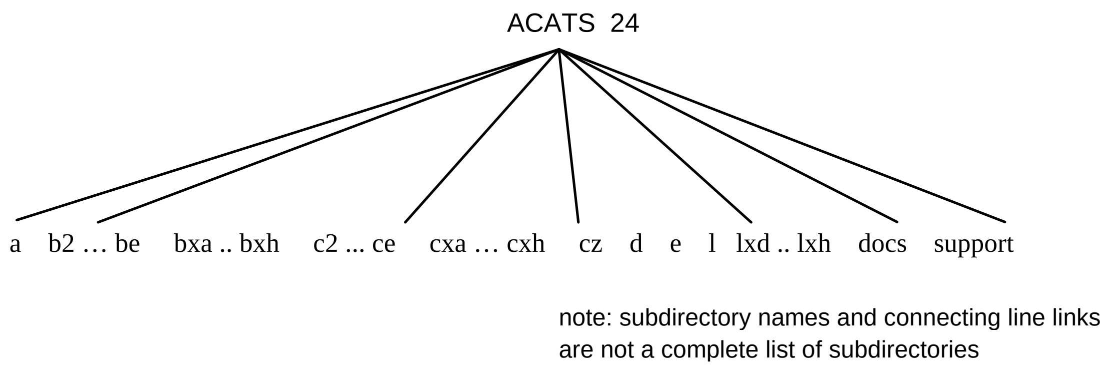
ACATS 24
a
b2 … be
bxa .. bxh
c2 ... ce
cxa … cxh
cz
d
e
l
lxd .. lxh
docs
support
note: subdirectory names and connecting line links are not a complete list of subdirectories

Figure 3. Delivery Directory Structure

3.9 File Format

To conserve space, all files in the delivered ACATS 2.4 (including test files, foundation files, and support files) have been compressed. Decompressed files (see Section 4.2.2) use only ASCII characters. Other than the documentation files, no formatting control characters, rulers or other information intended for word processors or editors is included in the files. (The documentation files are all provided as ASCII text files, but a version formatted for Microsoft Word 97 is also provided for greater readability).

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.

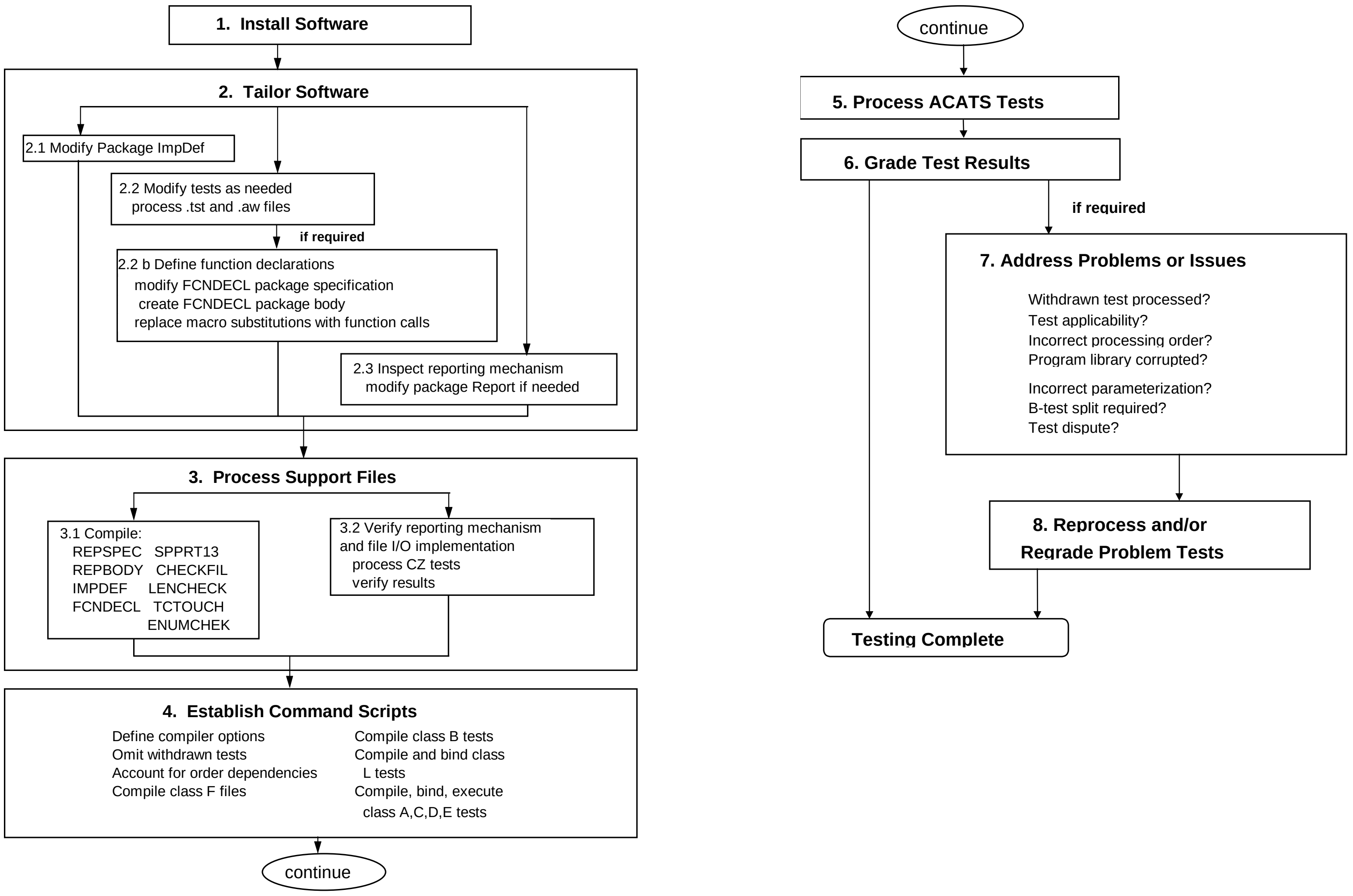
1. Install Software
2. Tailor Software
2.1 Modify Package ImpDef
2.2 Modify tests as needed
process .tst and .aw files
if required
2.2 b Define function declarations
modify FCNDECL package specification
create FCNDECL package body
replace macro substitutions with function calls
2.3 Inspect reporting 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
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

Figure 4 (Cont.) Using the ACATS

4. Using The ACATS

4.1 Introduction

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 figure 4.

4.2 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.

4.2.1 Contents of the ACATS Delivery

The delivery consists of 8 archives (sets of compressed files) or 8 compressed tar files. Each 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. Each archive contains a readme*x*.txt file (where '*x*' is a digit representing the number of the archive), which contains decompression suggestions and an overview of the contents of the archive or tar file. These files are not considered part of the ACATS; they exist so that someone finding one of the archive files can identify what it is. The remainder of 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 [Pro01] 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 are documented in the ACATS Modification List (AML). This list includes a list of all new tests, all modified tested, 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. An archive and tar file containing the new and/or modified tests is available. The files are named MOD_2_4x, 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

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

4.2.2 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 8 zip archives (files) with the file extension .zip. Eight Unix compressed files, with extension .Z, contain Unix tar files. 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:

```
./acats2_4/a          ./acats2_4/c4         ./acats2_4/bxd
./acats2_4/b2         ./acats2_4/c5         ./acats2_4/bxe
./acats2_4/b3         ./acats2_4/c6         ./acats2_4/bxf
./acats2_4/b4         ./acats2_4/c7         ./acats2_4/bxg
./acats2_4/b5         ./acats2_4/c8         ./acats2_4/bxh
./acats2_4/b6         ./acats2_4/c9         ./acats2_4/cxc
./acats2_4/b7         ./acats2_4/ca         ./acats2_4/cxd
./acats2_4/b8         ./acats2_4/cb         ./acats2_4/cxe
./acats2_4/b9         ./acats2_4/cc         ./acats2_4/cxf
./acats2_4/ba         ./acats2_4/cd         ./acats2_4/cxg
./acats2_4/bb         ./acats2_4/ce         ./acats2_4/cxh
./acats2_4/bc         ./acats2_4/cz         ./acats2_4/lxd
./acats2_4/bd         ./acats2_4/d          ./acats2_4/lxe
./acats2_4/be         ./acats2_4/e          ./acats2_4/lxh
./acats2_4/bxa        ./acats2_4/l          ./acats2_4/docs
./acats2_4/bxb        ./acats2_4/cxa        ./acats2_4/support
./acats2_4/c2         ./acats2_4/cxb
./acats2_4/c3         ./acats2_4/bxc
```

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

4.2.2.1 Decompressing Zipped Files

All ACATS files have been compressed (zipped) into compressed archives (zip-files) that have the MS-DOS file extension "`.zip`". A DOS 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 2.4 files may be decompressed using the following steps. Approximately 25 MB of free space on a DOS 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 "acats2_4", but any name can be used. Copy each 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, insure that "Use Directory Names" is checked. Also, insure that the files are decompressed into the proper directory. For command line decompressors, this means insuring that the current subdirectory is acats2_4. For "winzip", this simply means selecting acats2_4 as the extract path.

For example, using unzip, and assuming that the archive name is `ACATS2.zip`, type

```
cd acats2_4
```

to set the proper directory, and

```
unzip ACATS2
```

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.

4.2.2.2 Decompressing Unix Compress Files

All ACATS files have been included in 8 Unix tar format files and then compressed using the Unix compress utility. To create a set of ACATS files, first copy the compressed files acats23?.tar.Z from the distribution source to a hard drive. Uncompress the file with the Unix command

```
uncompress acats23?.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 ACATS2_2 directory to be created and then untar each of the ACATS files

```
tar -xvf <path>/acats23?.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.

4.2.3 Files With Non-Graphic Characters

Four ACATS test files contain 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. The following paragraphs describe the four tests.

4.2.3.1 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	ASCII Value	
		Decimal	Hex
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

4.2.3.2 B25002A

This test checks that 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.

4.2.3.3 B25002B

This test checks that the five 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	ASCII Value	
		Decimal	Hex
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

4.2.3.4 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.

4.3 Tailoring the ACATS Test Suite

There are some files in the delivery that require modification before ACATS 2.4 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. ImpDef is a package that is new to the 2.X suite, and all users will have to do this modification. The macros.dfs file must similarly be edited to include values suitable for testing. This file is slightly different from previous ACATS suites, so all users will have to modify it, but most changes can be retained from previous versions. 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 (repbody.ada) must be modified if necessary; previous modifications can generally be carried forward. The required customization is described in the following sections.

4.3.1 ImpDef Customization

All implementations *must* customize impdef.a for ACATS 2.4 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. Note that in ACATS 2.4, ImpDef uses child libraries for the Specialized Needs Annexes. The only ImpDef children that need be modified are those associated with the SNAs that the implementer intends to test during a conformity assessment.

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 one procedure and/or one function. Bodies so provided must satisfy requirements stated in ImpDef.

- It is not used for macro expansion of tests. Instead, ImpDef must be available at compile time (i.e., included in the environment) for tests that rely upon it.

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. 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.) A copy of ImpDef is included in Appendix B.

4.3.2 Macro Defs Customization

The details of macro.dfs have not changed from ACATS 2.3 to ACATS 2.4. A version of macro.dfs that was tailored for ACATS 2.3 should be valid for ACATS 2.4 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.

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, and is included in Appendix D; 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.

4.3.3 Processing for Wide_Character Tests

There are two tests in ACATS 2.4 that **require** preprocessing. They must be processed with the Wide Character tool; the macro expander tool will not work with them. Information for these tests is **not** included in macro.dfs.

There are two tests in ACATS 2.4 that check an implementation's ability to process characters drawn from the full set of graphic symbols of ISO 10646 BMP (See [Ada95] 2.1). Since such characters cannot be included in the distribution media in a way that can reliably be read by an arbitrary implementation, they contain character sequences that must be replaced by the intended character. A special tool, the WideChr program, which will automatically perform the required substitutions, has been included with this distribution.

The affected tests are contained in files with the extension .aw. Each such test contains a six or eight character sequence of the form

"[ab]"

or

"[abcd]"

Note that double quotes make up part of the special sequence (acting as part of the escape sequence). The processor will replace the string with a character that is designated by 16#abcd#, where the alphanumeric characters ‘a’, ‘b’, ‘c’, ‘d’, are hexadecimal digits. Note that the strings to be replaced do not start with ‘$’, and the replacement is synthetic, not substitution. Therefore, the macro expander tool will not work with these tests.

The WideChr tool takes the designated tests as input. The names of the required tests are included in the WideChr tool code as constants. It reads path names for the tests from ImpDef. The tool reads the tests, synthesizes the necessary replacements, and writes the resulting, compilable programs into files with the same name as the original but with the extension .a.

Substitutions using the WideChr program may be made as follows:

1. Edit the file impdef.a to indicate the path where the tests are located. This value will be concatenated with the test name to form the complete name of a file.
2. Compile and bind WideChr.
3. Run WideChr.

The program will replace all special sequences in the .aw files with synthesized characters. Test files with the original test name but the extension .a, in the same path location as the original .aw files, will contain the processable tests. The original .aw files will not be modified.

4.3.4 Package SPPRT13 and Function FcnDecl

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

The version of SPPRT13 distributed with ACATS 2.4 is slightly different from the version distributed with ACVC 1.11. A body is not required for this package (and would, therefore, be illegal in Ada95).

All implementations should verify that package SPPRT13 can be properly customized using the macro substitution technique. Note that in Ada95, a body for SPPRT13 is illegal.

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.

4.3.5 Modification of Package REPORT

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

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

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

4.3.6 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 limitations 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 a length 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 4.6.5.4) 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 4.2.1), then the required modifications must be made. The permitted modifications may be made if desired (or if necessary for the particular implementation).

4.4 Processing the Support Files

After all the files identified in Section 4.3 have been customized as needed and required, the support files can be processed and the reporting mechanism can be verified.

4.4.1 Support Files

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

`repspec.ada`	`repbody.ada`
`impdef.a`	`impdefc.a` (If testing Annex C)
`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)
`spprt13s.adt` (after macro substitution)	
`tctouch.ada`	

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

4.4.2 "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 package Report's reporting facilities, 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 Appendix C (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 Appendix C (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 Appendix C; since this output includes values from the customization impdef, non-failure lines may vary from those in the expected output. However, the number of lines and their relative positions may not change.

4.5 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.

4.5.1 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.

4.5.2 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.

4.6 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.

4.6.1 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 4.6.4) must be processed with acceptable results for conformity assessment of the core language. All legacy tests, as well as all newer 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.

4.6.2 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 which test Annex E, Distributed Systems.

4.6.3 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.

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.

4.6.4 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 Ada95 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 [Ada95] need not be processed except by implementations that wish to have Annex results documented. In that case, only the tests for the annex 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 [Ada95] 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).

4.6.5 Tests with Special Processing Requirement

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 included in Appendix A.

4.6.5.1 Foreign Language Interface Tests

Annex B, Interface to Other Languages, is part of the Ada95 core language. Any implementation that provides one or more of the packages Interfaces.C, Interfaces.COBOL, or Interfaces.Fortran **must** correctly process, and pass, the tests for interfaces to C, Cobol, and/or Fortran code respectively, with the possible exception of tests containing actual foreign code.

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

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

The tests involving interfaces to foreign code are listed in the following sections.

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.

4.6.5.1.1 C Language Interface

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

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

CD30005*	CXB3005	CXB3010	CXB3015
CXB3001	CXB3006*	CXB3011	CXB3016
CXB3002	CXB3007	CXB3012	
CXB3003	CXB3008	CXB3013*	
CXB3004*	CXB3009	CXB3014	

4.6.5.1.2 Cobol Language Interface

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

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

CXB4001	CXB4004	CXB4007
CXB4002	CXB4005	CXB4008
CXB4003	CXB4006	CXB4009*

4.6.5.1.3 Fortran Language Interface

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

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

CXB5001	CXB5003	CXB5005*
CXB5002	CXB5004*	

4.6.5.2 Tests for the Distributed Processing Annex

The ACATS tests for the Distribution 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 Real-Time Annex is also supported.

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

4.6.5.2.1 Remote_Call_Interface pragma

[Ada95] allows explicit message-based communication between active partitions as an alternative to RPC [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

4.6.5.2.2 Partition Communication System

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

CXE1001	CXE4001	CXE4003	CXE4005
CXE2001	CXE4002	CXE4004	CXE4006

4.6.5.2.3 System.RPC

Two tests provide a body for System.RPC. An implementation may include a private part that includes declarations, such as additional procedures and functions, that impose additional requirements on System.RPC. If an implementation includes additional declarations, then the same declarations (and implementations) may be added to the body of System.RPC in the tests identified below. Declarations in the private part of the implementation’s System.RPC do not affect the applicability of the tests in this group.

CXE5002	CXE5003

4.6.5.2.4 Real-Time Annex Support

Many implementations that support the Distribution 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 Distribution 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".

4.6.5.2.5 Configuring Multi-Partition Tests

Some Distribution 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.

4.6.5.2.6 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.

4.6.5.3 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	CXG2010	CXG2016	CXG2022
CXG2004	CXG2011	CXG2017	CXG2023
CXG2006	CXG2012	CXG2018	CXG2024
CXG2007	CXG2013	CXG2019	
CXG2008	CXG2014	CXG2020	
CXG2009	CXG2015	CXG2021	

4.6.5.4 Tests that use Configuration Pragmas

Several of the tests in Annex D, Real Time Processing, Annex E, Distributed Processing, and Annex H, Safety and Security, 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	

4.6.6 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 newer 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 [Ada95]. Further, note that the general style and approach of the newer 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 2.4 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.

4.7 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 above in test splitting and modification; 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 4.7.1 - 4.7.3. Tests that are non-applicable for an implementation are discussed in 4.7.4. Withdrawn tests are discussed in 4.7.5.

4.7.1 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 L of [Ada95] 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.

4.7.2 Expected Results for Class B

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. Error messages must provide some means of specifying the location of an error, but they are not required to be in direct proximity with the "-- ERROR:" marking of the errors.

Some B-tests also include the notation "-- OK" to indicate constructs that must **not** be identified as errors. **This is especially important since some constructs were errors in Ada83 that are legal in Ada95**.

Note that the error and OK markings may occur in lower or mixed case, as well as upper case.

Some B-tests exercise constructs whose correctness depends on source code that is textually separated (e.g., 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.

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, e.g.) 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.

A test is graded as "failed" if it fails to report on each error in the test or if it marks legal code as erroneous.

4.7.3 Expected Results for Class L

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 (twenty-six 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.

4.7.4 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. Appendix D of this user's guide 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:

a. whether a test completes execution and reports "NOT APPLICABLE";

b. 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.

4.7.5 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.

4.8 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).

4.8.1 Typical Issues

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

Processing a test that is withdrawn;

Processing 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 4.7.4;

Processing files (or tests, see Section 4.5.2) 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;

Incorrect values in macro.dfs, which result in incorrectly customized tests;

Incorrect substitutions in wide_character tests;

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.

4.8.2 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 [Pro01]. A petition must clearly state whether it is a claim that the test does not apply to the implementation or that the test is erroneous. 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.

4.9 Reprocessing and Regrading

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

Appendix A:
Version Description

ACATS 2.4 includes 3677 tests in 4206 files, not including foundation and other support units. From Version 2.3 to 2.4, 4 tests were added, comprising 10 files. 7 tests (7 files) were modified. Two support units (2 files) also were modified.

The following sections present a detailed description of ACATS 2.4, as follows:

A.1	Names of all files containing tests for the core language
A.2	Names of all files containing tests for the Specialized Needs Annexes
A.3	Names of all files containing foundation code
A.4	Names of all files containing ACATS 2.4 documents
A.5	Names of all files containing support units
A.6	Names of tests with special processing requirements
A.7	Test files added since ACATS 2.3
A.8	Test files modified since ACATS 2.3
A.9	Support files modified since ACATS 2.3
A.10	Files present in ACATS 2.3 but not present in ACATS 2.4

4.10 Core Test Files

The following files contain the tests for core language features of Ada95; that is, for requirements specified in Sections 2 through 13, Annex A, Annex B and Annex J.

a22006b.ada	aa2010a.ada	b24001c.ada	b2a003c.ada
a22006c.ada	aa2012a.ada	b24005a.ada	b2a003d.ada
a22006d.ada	ac1015b.ada	b24005b.ada	b2a003e.ada
a26007a.tst	ac3106a.ada	b24007a.ada	b2a003f.ada
a27003a.ada	ac3206a.ada	b24009a.ada	b2a005a.ada
a29003a.ada	ac3207a.ada	b24009b.ada	b2a005b.ada
a2a031a.ada	ad7001b.ada	b24104a.ada	b2a007a.ada
a33003a.ada	ad7001c0.ada	b24204a.ada	b2a010a.ada
a34017c.ada	ad7001c1.ada	b24204b.ada	b2a021a.ada
a35101b.ada	ad7001d0.ada	b24204c.ada	b32101a.ada
a35402a.ada	ad7001d1.ada	b24204d.ada	b32103a.ada
a35801f.ada	ad7006a.ada	b24204e.ada	b32104a.ada
a35902c.ada	ad7101a.ada	b24204f.ada	b32106a.ada
a38106d.ada	ad7101c.ada	b24205a.ada	b32201a.ada
a38106e.ada	ad7102a.ada	b24206a.ada	b32202a.ada
a49027a.ada	ad7103a.ada	b24206b.ada	b32202b.ada
a49027b.ada	ad7103c.ada	b24211b.ada	b32202c.ada
a49027c.ada	ad7104a.ada	b25002a.ada	b330001.a
a54b01a.ada	ad7201a.ada	b25002b.ada	b33001a.ada
a54b02a.ada	ad7203b.ada	b26001a.ada	b33101a.ada
a55b12a.ada	ad7205b.ada	b26002a.ada	b33102a.ada
a55b13a.ada	ad8011a.tst	b26005a.ada	b33102b.ada
a55b14a.ada	ada101a.ada	b28001a.ada	b33102c.ada
a71004a.ada	ae2113a.ada	b28001b.ada	b33102d.ada
a73001i.ada	ae2113b.ada	b28001c.ada	b33102e.ada
a73001j.ada	ae3002g.ada	b28001d.ada	b33201a.ada
a74105b.ada	ae3101a.ada	b28001e.ada	b33201b.ada
a74106a.ada	ae3702a.ada	b28001f.ada	b33201c.ada
a74106b.ada	ae3709a.ada	b28001g.ada	b33201d.ada
a74106c.ada	b22001a.tst	b28001h.ada	b33201e.ada
a74205e.ada	b22001b.tst	b28001i.ada	b33204a.ada
a74205f.ada	b22001c.tst	b28001j.ada	b33205a.ada
a83009a.ada	b22001d.tst	b28001k.ada	b33302a.ada
a83009b.ada	b22001e.tst	b28001l.ada	b34001b.ada
a83a02a.ada	b22001f.tst	b28001m.ada	b34001e.ada
a83a02b.ada	b22001g.tst	b28001n.ada	b34002b.ada
a83a06a.ada	b22001h.ada	b28001o.ada	b34003b.ada
a83a08a.ada	b22001i.tst	b28001p.ada	b34004b.ada
a83c01c.ada	b22001j.tst	b28001q.ada	b34005b.ada
a83c01h.ada	b22001k.tst	b28001r.ada	b34005e.ada
a83c01i.ada	b22001l.tst	b28001s.ada	b34005h.ada
a85007d.ada	b22001m.tst	b28001t.ada	b34005k.ada
a85013b.ada	b22001n.tst	b28001u.ada	b34005n.ada
a87b59a.ada	b23002a.ada	b28001v.ada	b34005q.ada
a95001c.ada	b23004a.ada	b28001w.ada	b34005t.ada
a95074d.ada	b23004b.ada	b29001a.ada	b34006b.ada
a97106a.ada	b24001a.ada	b2a003a.ada	b34006e.ada
a99006a.ada	b24001b.ada	b2a003b.ada	b34006h.ada

b34006k.ada
b34007b.ada
b34007e.ada
b34007h.ada
b34007k.ada
b34007n.ada
b34007q.ada
b34007t.ada
b34008b.ada
b34009b.ada
b34009e.ada
b34009h.ada
b34009k.ada
b34011a.ada
b34014b.ada
b34014d.ada
b34014f.ada
b34014i.ada
b34014m.ada
b34014o.ada
b34014q.ada
b34014s.ada
b34014v.ada
b34014z.ada
b35004a.ada
b35101a.ada
b35103a.ada
b35103b.ada
b35302a.ada
b354001.a
b35401a.ada
b35401b.ada
b35403a.ada
b35501a.ada
b35501b.ada
b35506a.ada
b35506b.ada
b35506c.ada
b35506d.ada
b35701a.ada
b35709a.ada
b35901a.ada
b35901c.ada
b35901d.ada
b35a01a.ada
b35a08a.ada
b360001.a
b36001a.ada
b36002a.ada
b36101a.ada
b36102a.ada
b36103a.ada
b36105c.dep
b36171a.ada
b36171b.ada
b36171c.ada
b36171d.ada
b36171e.ada
b36171f.ada
b36171g.ada
b36171h.ada
b36171i.ada
b36201a.ada
b36307a.ada
b370001.a
b370002.a
b37004a.ada
b37004b.ada
b37004c.ada
b37004d.ada
b37004e.ada
b37004f.ada
b37004g.ada
b37101a.ada
b37102a.ada
b37104a.ada
b37106a.ada
b37201a.ada
b37201b.ada
b37203a.ada
b37301i.ada
b37301j.ada
b37302a.ada
b37303a.ada
b37309b.ada
b37310b.ada
b37311a.ada
b37401a.ada
b37409b.ada
b380001.a
b38003a.ada
b38003b.ada
b38003c.ada
b38003d.ada
b38008a.ada
b38008b.ada
b38009a.ada
b38009d.ada
b38101a.ada
b38101b.ada
b38101c.ada
b38103a.ada
b38103b.ada
b38103c0.ada
b38103c1.ada
b38103c2.ada
b38103c3.ada
b38103d.ada
b38103e0.ada
b38103e1.ada
b38105a.ada
b38105b.ada
b38203a.ada
b390001.a
b391001.a
b391002.a
b391003.a
b391004.a
b392001.a
b392002.a
b392003.a
b392004.a
b392005.a
b392006.a
b392007.a
b392008.a
b392009.a
b392010.a
b393001.a
b393002.a
b393003.a
b393004.a
b393005.a
b393006.a
b393007.a
b3a0001.a
b3a0002.a
b3a0003.a
b3a0004.a
b3a2002.a
b3a2003.a
b3a2004.a
b3a2005.a
b3a2006.a
b3a2007.a
b3a2008.a
b3a2009.a
b3a2010.a
b3a2011.a
b3a2012.a
b3a2013.a
b3a2014.a
b3a2015.a
b3a2016.a
b41101a.ada
b41101c.ada
b41201a.ada
b41201c.ada
b41202c.ada
b41202d.ada
b41324b.ada
b41325b.ada
b41327b.ada
b420001.a
b430001.a
b43001m.ada
b43002d.ada
b43002e.ada
b43002f.ada
b43002g.ada
b43002h.ada
b43002i.ada
b43002j.ada
b43002k.ada
b43005a.ada
b43005b.ada
b43005f.ada
b43101a.ada
b43102a.ada
b43102b.ada
b43105c.ada
b43201a.ada
b43201c.ada
b43201d.ada
b43202a.ada
b43202c.ada
b43209b.ada
b43221a.ada
b43221b.ada
b43223a.ada
b44001a.ada
b44001b.ada
b44002b.ada
b44002c.ada
b44004a.ada
b44004b.ada
b44004c.ada
b44004d.ada
b44004e.ada
b45102a.ada
b45116a.ada
b45121a.ada
b45204a.ada
b45205a.ada
b45206c.ada
b45207a.ada
b45207b.ada
b45207c.ada
b45207d.ada
b45207g.ada
b45207h.ada
b45207i.ada
b45207j.ada
b45207m.ada
b45207n.ada
b45207o.ada
b45207p.ada
b45207s.ada
b45207t.ada
b45207u.ada

b45207v.ada	b48001b.ada	b54a10a.ada	b59001c.ada
b45208a.ada	b48002a.ada	b54a12a.ada	b59001d.ada
b45208b.ada	b48002b.ada	b54a20a.ada	b59001e.ada
b45208c.ada	b48002c.ada	b54a21a.ada	b59001f.ada
b45208g.ada	b48002d.ada	b54a25a.ada	b59001g.ada
b45208h.ada	b48002e.ada	b54a60a.ada	b59001h.ada
b45208i.ada	b48002g.ada	b54a60b.ada	b59001i.ada
b45208m.ada	b48003a.ada	b54b01b.tst	b610001.a
b45208n.ada	b48003b.ada	b54b01c.ada	b61001f.ada
b45208s.ada	b48003c.ada	b54b02b.ada	b61005a.ada
b45208t.ada	b48003d.ada	b54b02c.ada	b61006a.ada
b45209a.ada	b48003e.ada	b54b02d.ada	b61011a.ada
b45209b.ada	b490001.a	b54b04a.ada	b62001a.ada
b45209c.ada	b490002.a	b54b04b.ada	b62001b.ada
b45209d.ada	b49002a.ada	b54b05a.ada	b62001c.ada
b45209e.ada	b49004a.ada	b54b06a.ada	b62001d.ada
b45209f.ada	b49005a.ada	b55a01a.ada	b63001a.ada
b45209g.ada	b49007a.ada	b55a01d.ada	b63001b.ada
b45209h.ada	b49007b.ada	b55a01e.ada	b63005a.ada
b45209i.ada	b49008a.ada	b55a01j.ada	b63005b.ada
b45209j.ada	b49008c.ada	b55a01k.ada	b63006a.ada
b45209k.ada	b49009b.ada	b55a01l.ada	b63009a.ada
b45221a.ada	b49009c.ada	b55a01n.ada	b63009b.ada
b45261a.ada	b49010a.ada	b55a01o.ada	b63009c0.ada
b45261b.ada	b49011a.ada	b55a01t.ada	b63009c1.ada
b45261c.ada	b4a010c.ada	b55a01u.ada	b63009c2.ada
b45261d.ada	b4a016a.ada	b55a01v.ada	b63009c3.ada
b45301a.ada	b51001a.ada	b55b01a.ada	b63103a.ada
b45301b.ada	b51004b.ada	b55b01b.ada	b64002a.ada
b45301c.ada	b51004c.ada	b55b09b.ada	b64002c.ada
b45302a.ada	b52002a.ada	b55b09c.dep	b64003a.ada
b45341a.ada	b52002b.ada	b55b09d.dep	b64004a.ada
b455002.a	b52002c.ada	b55b12b.ada	b64004b.ada
b45501a.ada	b52002d.ada	b55b12c.ada	b64004c.ada
b45501b.ada	b52002e.ada	b55b17a.ada	b64004d.ada
b45501c.ada	b52002f.ada	b55b17b.ada	b64004e.ada
b45522a.ada	b52002g.ada	b55b17c.ada	b64004f.ada
b45537a.ada	b52004a.ada	b55b18a.ada	b641001.a
b45601a.ada	b52004b.ada	b56001a.ada	b64101a.ada
b45625a.ada	b52004c.ada	b56001d.ada	b64201a.ada
b45661a.ada	b52004d.dep	b56001e.ada	b65001a.ada
b460001.a	b52004e.dep	b56001f.ada	b65002a.ada
b460002.a	b53001a.ada	b56001g.ada	b65002b.ada
b460004.a	b53001b.ada	b56001h.ada	b660001.a
b46002a.ada	b53002a.ada	b57001a.ada	b660002.a
b46003a.ada	b53002b.ada	b57001b.ada	b66001a.ada
b46004a.ada	b53009a.ada	b57001c.ada	b66001b.ada
b46004b.ada	b53009b.ada	b57001d.ada	b66001c.ada
b46004c.ada	b53009c.ada	b58001a.ada	b66001d.ada
b46004d.ada	b54a01b.ada	b58002a.ada	b67001a.ada
b46004e.ada	b54a01f.ada	b58002b.ada	b67001b.ada
b46005a.ada	b54a01g.ada	b58002c.ada	b67001c.ada
b47001a.ada	b54a01l.ada	b58003a.ada	b67001d.ada
b480001.a	b54a05a.ada	b58003b.ada	b67001h.ada
b48001a.ada	b54a05b.ada	b59001a.ada	b67001i.ada

b67001j.ada
b67001k.ada
b67004a.ada
b71001a.ada
b71001b.ada
b71001c.ada
b71001d.ada
b71001f.ada
b71001g.ada
b71001h.ada
b71001i.ada
b71001j.ada
b71001l.ada
b71001m.ada
b71001n.ada
b71001o.ada
b71001p.ada
b71001r.ada
b71001t.ada
b71001u.ada
b71001v.ada
b7200010.a
b7200011.a
b7200012.a
b7200013.a
b7200014.a
b7200015.a
b7200016.a
b730001.a
b730002.a
b730003.a
b730004.a
b730005.a
b7300060.a
b7300061.a
b7300062.a
b7300063.am
b73001a.ada
b73001b.ada
b73001c.ada
b73001d.ada
b73001e.ada
b73001f.ada
b73001g.ada
b73001h.ada
b73004a.ada
b73004b0.ada
b73004b1.ada
b73004b2.ada
b7310010.a
b7310011.a
b7310012.a
b7310013.a
b7310014.a
b7310015.a
b7310016.am
b731a01.a
b731a02.a
b740001.a
b74001a.ada
b74001b.ada
b74101a.ada
b74101b.ada
b74103a.ada
b74103d.ada
b74103e.ada
b74103g.ada
b74103i.ada
b74104a.ada
b74105a.ada
b74105c.ada
b74201a.ada
b74202a.ada
b74202b.ada
b74202c.ada
b74202d.ada
b74203b.ada
b74203c.ada
b74203d.ada
b74203e.ada
b74205a.ada
b74207a.ada
b74304a.ada
b74304b.ada
b74304c.ada
b74404a.ada
b74404b.ada
b74409a.ada
b810001.a
b830001.a
b8300020.a
b8300021.a
b8300022.a
b8300023.a
b8300024.a
b8300025.am
b83001a.ada
b83003a.ada
b83003b0.ada
b83003b1.ada
b83003b2.ada
b83003b3.ada
b83003b4.ada
b83003c.ada
b83004a.ada
b83004b0.ada
b83004b1.ada
b83004b2.ada
b83004b3.ada
b83004c0.ada
b83004c1.ada
b83004c2.ada
b83004d0.ada
b83004d1.ada
b83004d2.ada
b83004d3.ada
b83006a.ada
b83006b.ada
b83008a.ada
b83008b.ada
b83011a.ada
b83023b.ada
b83024b.ada
b83024f0.ada
b83024f1.ada
b83024f2.ada
b83024f3.ada
b83026b.ada
b83027b.ada
b83027d.ada
b83028b.ada
b83029b.ada
b83030b.ada
b83030d.ada
b83031b.ada
b83031d.ada
b83031f.ada
b83032b.ada
b83033b.ada
b83041e.ada
b83a01a.ada
b83a01b.ada
b83a01c.ada
b83a05a.ada
b83a06b.ada
b83a06h.ada
b83a07a.ada
b83a07b.ada
b83a07c.ada
b83a08b.ada
b83a09a.ada
b83b01a.ada
b83b02c.ada
b83e01a.ada
b83e01b.ada
b83e01c.ada
b83e01d.ada
b83e01e0.ada
b83e01e1.ada
b83e01e2.ada
b83e01e3.ada
b83e01f0.ada
b83e01f1.ada
b83e01f2.ada
b83e01f3.ada
b83e01f4.ada
b83e01f5.ada
b83e01f6.ada
b83e11a.ada
b83f02a.ada
b83f02b.ada
b83f02c.ada
b840001.a
b84001a.ada
b84002b.ada
b84004a.ada
b84005b.ada
b84006a.ada
b84007a.ada
b84008b.ada
b85001a.ada
b85001b.ada
b85001c.ada
b85001d.ada
b85001e.ada
b85001f.ada
b85001g.ada
b85001h.ada
b85001i.ada
b85001j.ada
b85001k.ada
b85001l.ada
b85002a.ada
b85003a.ada
b85003b.ada
b85004a.ada
b85008f.ada
b85008g.ada
b85008h.ada
b85010a.ada
b85010b.ada
b85012a.ada
b85013c.ada
b85013d.ada
b85015a.ada
b86001a0.ada
b86001a1.ada
b87b23b.ada
b87b26a.ada
b87b48c.ada
b91001b.ada
b91001c.ada
b91001d.ada
b91001e.ada
b91001f.ada
b91001g.ada
b91002a.ada
b91002b.ada
b91002c.ada
b91002d.ada

b91002e.ada	b95070a.ada	b99002a.ada	ba1010h2.ada
b91002f.ada	b95080a.ada	b99002b.ada	ba1010i0.ada
b91002g.ada	b95080c.ada	b99002c.ada	ba1010i1.ada
b91002h.ada	b95081a.ada	b99003a.ada	ba1010i3.ada
b91002i.ada	b95082a.ada	b9a001a.ada	ba1010i4.ada
b91002j.ada	b95082b.ada	b9a001b.ada	ba1010j0.ada
b91002k.ada	b95082c.ada	ba1001a0.ada	ba1010j1.ada
b91002l.ada	b95082d.ada	ba1001a1.ada	ba1010j2.ada
b91003a.ada	b95082e.ada	ba1001a4.ada	ba1010j4.ada
b91003b.ada	b95082f.ada	ba1001ac.ada	ba1010j5.ada
b91003c.ada	b95083a.ada	ba1001d.ada	ba1010j6.ada
b91003d.ada	b95094a.ada	ba1010a0.ada	ba1010j7.ada
b91003e.ada	b95094b.ada	ba1010a1.ada	ba1010j8.ada
b91004a.ada	b95094c.ada	ba1010a2.ada	ba1010k0.ada
b91005a.ada	b951001.a	ba1010a3.ada	ba1010k1.ada
b92001a.ada	b952001.a	ba1010b0.ada	ba1010k2.ada
b92001b.ada	b952002.a	ba1010b1.ada	ba1010k3.ada
b940001.a	b952003.a	ba1010b2.ada	ba1010k4.ada
b940002.a	b952004.a	ba1010b4.ada	ba1010k5.ada
b940003.a	b954001.a	ba1010b5.ada	ba1010k6.ada
b940004.a	b954003.a	ba1010b6.ada	ba1010l0.ada
b940005.a	b954004.a	ba1010b7.ada	ba1010l1.ada
b940006.a	b960001.a	ba1010b8.ada	ba1010l2.ada
b940007.a	b96002a.ada	ba1010c0.ada	ba1010l3.ada
b95001a.ada	b97102b.ada	ba1010c1.ada	ba1010l4.ada
b95001b.ada	b97102c.ada	ba1010c2.ada	ba1010l5.ada
b95001d.ada	b97102d.ada	ba1010c3.ada	ba1010l6.ada
b95002a.ada	b97102f.ada	ba1010c4.ada	ba1010m0.ada
b95003a.ada	b97102g.ada	ba1010c5.ada	ba1010m1.ada
b95004a.ada	b97102h.ada	ba1010c6.ada	ba1010m3.ada
b95004b.ada	b97102i.ada	ba1010d0.ada	ba1010m4.ada
b95006a.ada	b97103a.ada	ba1010d1.ada	ba1010m5.ada
b95006b.ada	b97103b.ada	ba1010d2.ada	ba1010m6.ada
b95006c.ada	b97103d.ada	ba1010d3.ada	ba1010m7.ada
b95006d.ada	b97103e.ada	ba1010e0.ada	ba1010m8.ada
b95007a.ada	b97103f.ada	ba1010e1.ada	ba1010n0.ada
b95007b.ada	b97103g.ada	ba1010e2.ada	ba1010n2.ada
b95020a.ada	b97104a.ada	ba1010e3.ada	ba1010n3.ada
b95020b0.ada	b97104b.ada	ba1010e4.ada	ba1010n4.ada
b95020b1.ada	b97104c.ada	ba1010e5.ada	ba1010n5.ada
b95020b2.ada	b97104d.ada	ba1010e6.ada	ba1010p0.ada
b95030a.ada	b97104e.ada	ba1010f0.ada	ba1010p2.ada
b95031a.ada	b97104f.ada	ba1010f1.ada	ba1010q0.ada
b95032a.ada	b97104g.ada	ba1010f3.ada	ba1010q1.ada
b95061a.ada	b97107a.ada	ba1010f4.ada	ba1010q3.ada
b95061b.ada	b97108a.ada	ba1010f5.ada	ba1010q4.ada
b95061c.ada	b97108b.ada	ba1010f6.ada	ba1011b0.ada
b95061d.ada	b97109a.ada	ba1010f7.ada	ba1011b1.ada
b95061e.ada	b97110a.ada	ba1010f8.ada	ba1011b2.ada
b95061f.ada	b97110b.ada	ba1010g0.ada	ba1011b3.ada
b95061g.ada	b97111a.ada	ba1010g2.ada	ba1011b4.ada
b95062a.ada	b97206a.ada	ba1010g3.ada	ba1011b5.ada
b95063a.ada	b97306a.ada	ba1010g4.ada	ba1011b6.ada
b95064a.ada	b99001a.ada	ba1010g5.ada	ba1011b7.ada
b95068a.ada	b99001b.ada	ba1010h0.ada	ba1011b8.ada

ba1011c0.ada	ba1101c4.ada	ba2013a.ada	bc1001a.ada
ba1011c1.ada	ba1101c5.ada	ba2013b.ada	bc1002a.ada
ba1011c2.ada	ba1101c6.ada	ba21001.a	bc1005a.ada
ba1011c3.ada	ba1101e0.ada	ba21002.a	bc1008a.ada
ba1011c4.ada	ba1101e1.ada	ba210030.a	bc1008b.ada
ba1011c5.ada	ba1101f.ada	ba210031.a	bc1008c.ada
ba1011c6.ada	ba1101g.ada	ba210032.a	bc1009a.ada
ba1011c7.ada	ba1109a0.ada	ba210033.a	bc1011a.ada
ba1011c8.ada	ba1109a1.ada	ba210034.a	bc1011b.ada
ba1020a0.ada	ba1109a2.ada	ba210035.a	bc1011c.ada
ba1020a1.ada	ba1110a0.ada	ba210040.a	bc1012a.ada
ba1020a2.ada	ba1110a1.ada	ba210041.a	bc1013a.ada
ba1020a3.ada	ba1110a2.ada	ba210042.a	bc1014a.ada
ba1020a4.ada	ba1110a3.ada	ba210043.a	bc1014b.ada
ba1020a5.ada	ba1110a4.ada	ba210044.a	bc1016a.ada
ba1020a6.ada	ba1110a5.ada	ba210045.am	bc1016b.ada
ba1020a7.ada	ba12001.a	ba21a01.a	bc1101a.ada
ba1020a8.ada	ba12002.a	ba21a02.a	bc1102a.ada
ba1020b0.ada	ba12003.a	ba3001a0.ada	bc1103a.ada
ba1020b1.ada	ba12004.a	ba3001a1.ada	bc1106a.ada
ba1020b2.ada	ba12005.a	ba3001a2.ada	bc1107a.ada
ba1020b3.ada	ba12007.a	ba3001a3.ada	bc1109a.ada
ba1020b4.ada	ba12008.a	ba3001b0.ada	bc1109b.ada
ba1020b5.ada	ba13b01.a	ba3001b1.ada	bc1109c.ada
ba1020b6.ada	ba13b02.a	ba3001c0.ada	bc1109d.ada
ba1020c0.ada	ba15001.a	ba3001c1.ada	bc1110a.ada
ba1020c1.ada	ba150020.a	ba3001e0.ada	bc1201a.ada
ba1020c2.ada	ba150021.a	ba3001e1.ada	bc1201b.ada
ba1020c3.ada	ba150022.a	ba3001e2.ada	bc1201c.ada
ba1020c4.ada	ba150023.a	ba3001e3.ada	bc1201d.ada
ba1020c5.ada	ba150024.a	ba3001f0.ada	bc1201e.ada
ba1020f0.ada	ba150025.a	ba3001f1.ada	bc1201f.ada
ba1020f1.ada	ba150026.a	ba3001f2.ada	bc1201g.ada
ba1020f2.ada	ba150027.a	ba3001f3.ada	bc1201h.ada
ba11001.a	ba150028.a	ba3006a0.ada	bc1201i.ada
ba11002.a	ba150029.am	ba3006a1.ada	bc1201j.ada
ba11003.a	ba2001a.ada	ba3006a2.ada	bc1201k.ada
ba11004.a	ba2001b.ada	ba3006a3.ada	bc1201l.ada
ba11005.a	ba2001c.ada	ba3006a4.ada	bc1202a.ada
ba11007.a	ba2001d.ada	ba3006a5.ada	bc1202c.ada
ba11008.a	ba2001f0.ada	ba3006a6.ada	bc1202e.ada
ba11009.a	ba2001f1.ada	ba3006b0.ada	bc1202f.ada
ba11010.a	ba2001f2.ada	ba3006b1.ada	bc1202g.ada
ba11011.a	ba2003b0.ada	ba3006b2.ada	bc1203a.ada
ba11012.a	ba2003b1.ada	ba3006b3.ada	bc1205a.ada
ba1101a.ada	ba2011a0.ada	ba3006b4.ada	bc1206a.ada
ba1101b0.ada	ba2011a1.ada	bb10001.a	bc1207a.ada
ba1101b1.ada	ba2011a2.ada	bb20001.a	bc1208a.ada
ba1101b2.ada	ba2011a3.ada	bb2001a.ada	bc1226a.ada
ba1101b3.ada	ba2011a4.ada	bb2002a.ada	bc1230a.ada
ba1101b4.ada	ba2011a5.ada	bb2003a.ada	bc1303a.ada
ba1101c0.ada	ba2011a6.ada	bb2003b.ada	bc1303b.ada
ba1101c1.ada	ba2011a7.ada	bb2003c.ada	bc1303c.ada
ba1101c2.ada	ba2011a8.ada	bb3001a.ada	bc1303d.ada
ba1101c3.ada	ba2011a9.ada	bb3002a.ada	bc1303e.ada

bc1303f.ada
bc1303g.ada
bc1306a.ada
bc2001b.ada
bc2001c.ada
bc2001d.ada
bc2001e.ada
bc2004a.ada
bc2004b.ada
bc30001.a
bc3001a.ada
bc3002a.ada
bc3002b.ada
bc3002c.ada
bc3002d.ada
bc3002e.ada
bc3005a.ada
bc3005b.ada
bc3005c.ada
bc3006a.ada
bc3009c.ada
bc3011b.ada
bc3013a.ada
bc3016g.ada
bc3018a.ada
bc3101a.ada
bc3101b.ada
bc3102a.ada
bc3102b.ada
bc3103b.ada
bc3123c.ada
bc3201a.ada
bc3201b.ada
bc3201c.ada
bc3202a.ada
bc3202b.ada
bc3202c.ada
bc3202d.ada
bc3205c.ada
bc3301a.ada
bc3301b.ada
bc3302a.ada
bc3302b.ada
bc3303a.ada
bc3304a.ada
bc3401a.ada
bc3401b.ada
bc3402a.ada
bc3402b.ada
bc3403a.ada
bc3403b.ada
bc3403c.ada
bc3404a.ada
bc3404b.ada
bc3404c.ada
bc3404d.ada
bc3404e.ada
bc3404f.ada
bc3405a.ada
bc3405b.ada
bc3405d.ada
bc3405e.ada
bc3405f.ada
bc3501a.ada
bc3501b.ada
bc3501c.ada
bc3501d.ada
bc3501e.ada
bc3501f.ada
bc3501g.ada
bc3501h.ada
bc3501i.ada
bc3501j.ada
bc3501k.ada
bc3502a.ada
bc3502b.ada
bc3502c.ada
bc3502d.ada
bc3502e.ada
bc3502f.ada
bc3502g.ada
bc3502h.ada
bc3502i.ada
bc3502j.ada
bc3502k.ada
bc3502l.ada
bc3502m.ada
bc3502n.ada
bc3502o.ada
bc3503a.ada
bc3503c.ada
bc3503d.ada
bc3503e.ada
bc3503f.ada
bc3604a.ada
bc3604b.ada
bc3607a.ada
bc40001.a
bc40002.a
bc50001.a
bc50002.a
bc50003.a
bc50004.a
bc51002.a
bc51003.a
bc51004.a
bc51005.a
bc51006.a
bc51007.a
bc51011.a
bc51012.a
bc51013.a
bc51015.a
bc51016.a
bc51017.a
bc51018.a
bc51019.a
bc51020.a
bc51b01.a
bc51b02.a
bc51c01.a
bc51c02.a
bc53001.a
bc53002.a
bc54001.a
bc54002.a
bc54003.a
bc54a01.a
bc54a02.a
bc54a03.a
bc54a04.a
bc54a05.a
bc54a06.a
bc70001.a
bc70002.a
bc70003.a
bc70004.a
bc70005.a
bc70006.a
bc70007.a
bc70008.a
bc70009.a
bc70010.a
bd1b01a.ada
bd1b02b.ada
bd1b03c.ada
bd1b05e.ada
bd1b06j.ada
bd2001b.ada
bd2a01h.ada
bd2a02a.tst
bd2a03a.ada
bd2a03b.ada
bd2a06a.ada
bd2a25a.ada
bd2a35a.ada
bd2a45a.ada
bd2a55a.ada
bd2a55b.ada
bd2a67a.ada
bd2a77a.ada
bd2a85a.ada
bd2a85b.ada
bd2b01c.ada
bd2b02a.ada
bd2b03a.ada
bd2b03b.ada
bd2b03c.ada
bd2c01d.tst
bd2c02a.tst
bd2c03a.tst
bd2d01c.ada
bd2d01d.ada
bd2d02a.ada
bd2d03a.ada
bd2d03b.ada
bd3001a.ada
bd3001b.ada
bd3001c.ada
bd3002a.ada
bd3003a.ada
bd3003b.ada
bd3012a.ada
bd3013a.ada
bd4001a.ada
bd4002a.ada
bd4003a.ada
bd4003b.ada
bd4003c.ada
bd4006a.tst
bd4007a.ada
bd4009a.ada
bd4011a.ada
bd5001a.ada
bd5005a.ada
bd5005d.ada
bd5102a.ada
bd5102b.ada
bd5103a.ada
bd5104a.ada
bd7001a.ada
bd7101h.ada
bd7201c.ada
bd7203a.ada
bd7204a.ada
bd7205a.ada
bd7301a.ada
bd7302a.ada
bd8001a.tst
bd8002a.tst
bd8003a.tst
bd8004a.tst
bd8004b.tst
bd8004c.tst
bdb0a01.a
bdd2001.a
bde0001.a
bde0002.a
bde0003.a
bde0004.a

bde0005.a
bde0006.a
bde0007.a
bde0008.a
be2101e.ada
be2101j.ada
be2114a.ada
be2116a.ada
be2208a.ada
be3002a.ada
be3002e.ada
be3205a.ada
be3301c.ada
be3606c.ada
be3703a.ada
be3802a.ada
be3803a.ada
be3902a.ada
be3903a.ada
bxa8001.a
bxac001.a
bxac002.a
bxac003.a
bxac004.a
bxac005.a
c23001a.ada
c23003a.tst
c23003b.tst
c23003g.tst
c23003i.tst
c23006a.ada
c23006b.ada
c23006c.ada
c23006d.ada
c23006e.ada
c23006f.ada
c23006g.ada
c24002d.ada
c24003a.ada
c24003b.ada
c24003c.ada
c24106a.ada
c24202d.ada
c24203a.ada
c24203b.ada
c24207a.ada
c24211a.ada
c250001.aw
c250002.aw
c25001a.ada
c25001b.ada
c26006a.ada
c26008a.ada
c2a001a.ada
c2a001b.ada
c2a001c.ada
c2a002a.ada
c2a008a.ada
c2a021b.ada
c32001a.ada
c32001b.ada
c32001c.ada
c32001d.ada
c32001e.ada
c32107a.ada
c32107c.ada
c32108a.ada
c32108b.ada
c32111a.ada
c32111b.ada
c32112b.ada
c32113a.ada
c32115a.ada
c32115b.ada
c330001.a
c330002.a
c332001.a
c340001.a
c34001a.ada
c34001c.ada
c34001d.ada
c34001f.ada
c34002a.ada
c34002c.ada
c34003a.ada
c34003c.ada
c34004a.ada
c34004c.ada
c34005a.ada
c34005c.ada
c34005d.ada
c34005f.ada
c34005g.ada
c34005i.ada
c34005j.ada
c34005l.ada
c34005m.ada
c34005o.ada
c34005p.ada
c34005r.ada
c34005s.ada
c34005u.ada
c34005v.ada
c34006a.ada
c34006d.ada
c34006f.ada
c34006g.ada
c34006j.ada
c34006l.ada
c34007a.ada
c34007d.ada
c34007f.ada
c34007g.ada
c34007i.ada
c34007j.ada
c34007m.ada
c34007p.ada
c34007r.ada
c34007s.ada
c34007u.ada
c34007v.ada
c34008a.ada
c34009a.ada
c34009d.ada
c34009f.ada
c34009g.ada
c34009j.ada
c34009l.ada
c34011b.ada
c34012a.ada
c34014a.ada
c34014c.ada
c34014e.ada
c34014g.ada
c34014h.ada
c34014n.ada
c34014p.ada
c34014r.ada
c34014t.ada
c34014u.ada
c34018a.ada
c340a01.a
c340a02.a
c341a01.a
c341a02.a
c341a03.a
c341a04.a
c35003a.ada
c35003b.ada
c35003d.ada
c35102a.ada
c352001.a
c354002.a
c354003.a
c35502a.ada
c35502b.ada
c35502c.ada
c35502d.tst
c35502e.ada
c35502f.tst
c35502g.ada
c35502h.ada
c35502i.ada
c35502j.ada
c35502k.ada
c35502l.ada
c35502m.ada
c35502n.ada
c35502o.ada
c35502p.ada
c35503a.ada
c35503b.ada
c35503c.ada
c35503d.tst
c35503e.ada
c35503f.tst
c35503g.ada
c35503h.ada
c35503k.ada
c35503l.ada
c35503o.ada
c35503p.ada
c35504a.ada
c35504b.ada
c35505c.ada
c35505e.ada
c35505f.ada
c35507a.ada
c35507b.ada
c35507c.ada
c35507e.ada
c35507g.ada
c35507h.ada
c35507i.ada
c35507j.ada
c35507k.ada
c35507l.ada
c35507m.ada
c35507n.ada
c35507o.ada
c35507p.ada
c35508a.ada
c35508b.ada
c35508c.ada
c35508e.ada
c35508g.ada
c35508h.ada
c35508k.ada
c35508l.ada
c35508o.ada
c35508p.ada
c35703a.ada
c35704a.ada
c35704b.ada
c35704c.ada
c35704d.ada
c35801d.ada
c35902d.ada
c35904a.ada
c35904b.ada

c35a02a.ada	c37107a.ada	c38108b.ada	c392a01.a
c35a05a.ada	c37108b.ada	c38108c0.ada	c392c05.a
c35a05d.ada	c37206a.ada	c38108c1.ada	c392c07.a
c35a05n.ada	c37207a.ada	c38108c2.ada	c392d01.a
c35a05q.ada	c37208a.ada	c38108d0.ada	c392d02.a
c35a07a.ada	c37208b.ada	c38108d1.ada	c392d03.a
c35a07d.ada	c37209a.ada	c38202a.ada	c393001.a
c35a08b.ada	c37209b.ada	c3900010.a	c393007.a
c360002.a	c37210a.ada	c3900011.am	c393008.a
c36104a.ada	c37211a.ada	c390002.a	c393009.a
c36104b.ada	c37211b.ada	c390003.a	c393010.a
c36172a.ada	c37211c.ada	c390004.a	c393011.a
c36172b.ada	c37211d.ada	c3900050.a	c393012.a
c36172c.ada	c37211e.ada	c3900051.a	c393a02.a
c36174a.ada	c37213b.ada	c3900052.a	c393a03.a
c36180a.ada	c37213d.ada	c3900053.am	c393a05.a
c36202c.ada	c37213f.ada	c3900060.a	c393a06.a
c36203a.ada	c37213h.ada	c3900061.a	c393b12.a
c36204a.ada	c37213j.ada	c3900062.a	c393b13.a
c36204b.ada	c37213k.ada	c3900063.am	c393b14.a
c36204c.ada	c37213l.ada	c390007.a	c3a0001.a
c36204d.ada	c37215b.ada	c390010.a	c3a0002.a
c36205a.ada	c37215d.ada	c390011.a	c3a0003.a
c36205b.ada	c37215f.ada	c39006a.ada	c3a0004.a
c36205c.ada	c37215h.ada	c39006b.ada	c3a0005.a
c36205d.ada	c37217a.ada	c39006c0.ada	c3a0006.a
c36205e.ada	c37217b.ada	c39006c1.ada	c3a0007.a
c36205f.ada	c37217c.ada	c39006d.ada	c3a0008.a
c36205g.ada	c37304a.ada	c39006e.ada	c3a0009.a
c36205h.ada	c37305a.ada	c39006f0.ada	c3a0010.a
c36205i.ada	c37306a.ada	c39006f1.ada	c3a0011.a
c36205j.ada	c37309a.ada	c39006f2.ada	c3a00120.a
c36205k.ada	c37310a.ada	c39006f3.ada	c3a00121.a
c36205l.ada	c37312a.ada	c39006g.ada	c3a00122.am
c36301a.ada	c37402a.ada	c39007a.ada	c3a0013.a
c36301b.ada	c37403a.ada	c39007b.ada	c3a0014.a
c36302a.ada	c37404a.ada	c39008a.ada	c3a1001.a
c36304a.ada	c37404b.ada	c39008b.ada	c3a1002.a
c36305a.ada	c37405a.ada	c39008c.ada	c3a2001.a
c37002a.ada	c37411a.ada	c390a010.a	c3a2002.a
c37003a.ada	c38002a.ada	c390a011.am	c3a2003.a
c37003b.ada	c38002b.ada	c390a020.a	c3a2a01.a
c37005a.ada	c38005a.ada	c390a021.a	c3a2a02.a
c37006a.ada	c38005b.ada	c390a022.am	c410001.a
c37008a.ada	c38005c.ada	c390a030.a	c41101d.ada
c37008b.ada	c38006a.ada	c390a031.am	c41103a.ada
c37009a.ada	c38102a.ada	c391001.a	c41103b.ada
c37010a.ada	c38102b.ada	c391002.a	c41104a.ada
c37010b.ada	c38102c.ada	c392002.a	c41105a.ada
c371001.a	c38102d.ada	c392003.a	c41107a.ada
c371002.a	c38102e.ada	c392004.a	c41201d.ada
c371003.a	c38104a.ada	c392005.a	c41203a.ada
c37102b.ada	c38107a.ada	c392008.a	c41203b.ada
c37103a.ada	c38107b.ada	c392010.a	c41204a.ada
c37105a.ada	c38108a.ada	c392011.a	c41205a.ada

c41206a.ada	c432001.a	c45211a.ada	c45431a.ada
c41207a.ada	c432002.a	c45220a.ada	c455001.a
c41301a.ada	c432003.a	c45220b.ada	c45502b.dep
c41303a.ada	c432004.a	c45220c.ada	c45502c.dep
c41303b.ada	c43204a.ada	c45220d.ada	c45503a.ada
c41303c.ada	c43204c.ada	c45220e.ada	c45503b.dep
c41303e.ada	c43204e.ada	c45220f.ada	c45503c.dep
c41303f.ada	c43204f.ada	c45231a.ada	c45504a.ada
c41303g.ada	c43204g.ada	c45231b.dep	c45504b.dep
c41303i.ada	c43204h.ada	c45231c.dep	c45504c.dep
c41303j.ada	c43204i.ada	c45231d.tst	c45504d.ada
c41303k.ada	c43205a.ada	c45232b.ada	c45504e.dep
c41303m.ada	c43205b.ada	c45242b.ada	c45504f.dep
c41303n.ada	c43205c.ada	c45251a.ada	c45505a.ada
c41303o.ada	c43205d.ada	c45252a.ada	c45523a.ada
c41303q.ada	c43205e.ada	c45252b.ada	c45531a.ada
c41303r.ada	c43205g.ada	c45253a.ada	c45531b.ada
c41303s.ada	c43205h.ada	c45262a.ada	c45531c.ada
c41303u.ada	c43205i.ada	c45262b.ada	c45531d.ada
c41303v.ada	c43205j.ada	c45262c.ada	c45531e.ada
c41303w.ada	c43205k.ada	c45262d.ada	c45531f.ada
c41304a.ada	c43206a.ada	c45264a.ada	c45531g.ada
c41304b.ada	c43207b.ada	c45264b.ada	c45531h.ada
c41306a.ada	c43207d.ada	c45264c.ada	c45531i.ada
c41306b.ada	c43208a.ada	c45265a.ada	c45531j.ada
c41306c.ada	c43208b.ada	c45271a.ada	c45531k.ada
c41307d.ada	c43209a.ada	c45272a.ada	c45531l.ada
c41309a.ada	c43210a.ada	c45273a.ada	c45531m.dep
c41320a.ada	c43211a.ada	c45274a.ada	c45531n.dep
c41321a.ada	c43212a.ada	c45274b.ada	c45531o.dep
c41322a.ada	c43212c.ada	c45274c.ada	c45531p.dep
c41323a.ada	c43214a.ada	c45281a.ada	c45532a.ada
c41324a.ada	c43214b.ada	c45282a.ada	c45532b.ada
c41325a.ada	c43214c.ada	c45282b.ada	c45532c.ada
c41326a.ada	c43214d.ada	c45291a.ada	c45532d.ada
c41327a.ada	c43214e.ada	c45303a.ada	c45532e.ada
c41328a.ada	c43214f.ada	c45304a.ada	c45532f.ada
c41401a.ada	c43215a.ada	c45304b.dep	c45532g.ada
c41402a.ada	c43215b.ada	c45304c.dep	c45532h.ada
c41404a.ada	c43222a.ada	c45322a.ada	c45532i.ada
c420001.a	c43224a.ada	c45323a.ada	c45532j.ada
c42006a.ada	c433001.a	c45331a.ada	c45532k.ada
c42007e.ada	c44003d.ada	c45342a.ada	c45532l.ada
c43003a.ada	c44003f.ada	c45343a.ada	c45532m.dep
c43004a.ada	c44003g.ada	c45344a.ada	c45532n.dep
c43004c.ada	c450001.a	c45345b.ada	c45532o.dep
c431001.a	c45112a.ada	c45347a.ada	c45532p.dep
c43103a.ada	c45112b.ada	c45347b.ada	c45534b.ada
c43103b.ada	c45113a.ada	c45347c.ada	c45536a.dep
c43104a.ada	c45114b.ada	c45347d.ada	c45611a.ada
c43105a.ada	c452001.a	c45411a.ada	c45611b.dep
c43105b.ada	c45201a.ada	c45411b.dep	c45611c.dep
c43106a.ada	c45201b.ada	c45411c.dep	c45613a.ada
c43107a.ada	c45202b.ada	c45411d.ada	c45613b.dep
c43108a.ada	c45210a.ada	c45413a.ada	c45613c.dep

c45614a.ada	c47008a.ada	c52005c.ada	c54a24b.ada
c45614b.dep	c47009a.ada	c52005d.ada	c54a42a.ada
c45614c.dep	c47009b.ada	c52005e.ada	c54a42b.ada
c45622a.ada	c48004a.ada	c52005f.ada	c54a42c.ada
c45624a.ada	c48004b.ada	c52008a.ada	c54a42d.ada
c45624b.ada	c48004c.ada	c52008b.ada	c54a42e.ada
c45631a.ada	c48004d.ada	c52009a.ada	c54a42f.ada
c45631b.dep	c48004e.ada	c52009b.ada	c54a42g.ada
c45631c.dep	c48004f.ada	c52010a.ada	c55b03a.ada
c45632a.ada	c48005a.ada	c52011a.ada	c55b04a.ada
c45632b.dep	c48005b.ada	c52011b.ada	c55b05a.ada
c45632c.dep	c48006a.ada	c52101a.ada	c55b06a.ada
c45651a.ada	c48006b.ada	c52102a.ada	c55b06b.ada
c45662a.ada	c48007a.ada	c52102b.ada	c55b07a.dep
c45662b.ada	c48007b.ada	c52102c.ada	c55b07b.dep
c45672a.ada	c48007c.ada	c52102d.ada	c55b10a.ada
c460001.a	c48008a.ada	c52103a.ada	c55b11a.ada
c460002.a	c48008c.ada	c52103b.ada	c55b11b.ada
c460004.a	c48009a.ada	c52103c.ada	c55b15a.ada
c460005.a	c48009b.ada	c52103f.ada	c55b16a.ada
c460006.a	c48009c.ada	c52103g.ada	c55c02a.ada
c460007.a	c48009d.ada	c52103h.ada	c55c02b.ada
c460008.a	c48009e.ada	c52103k.ada	c56002a.ada
c460009.a	c48009f.ada	c52103l.ada	c57003a.ada
c460010.a	c48009g.ada	c52103m.ada	c57004a.ada
c460011.a	c48009h.ada	c52103p.ada	c57004b.ada
c46011a.ada	c48009i.ada	c52103q.ada	c58004c.ada
c46013a.ada	c48009j.ada	c52103r.ada	c58004d.ada
c46014a.ada	c48010a.ada	c52103x.ada	c58004g.ada
c46021a.ada	c48011a.ada	c52104a.ada	c58005a.ada
c46024a.ada	c48012a.ada	c52104b.ada	c58005b.ada
c46031a.ada	c490001.a	c52104c.ada	c58005h.ada
c46032a.ada	c490002.a	c52104f.ada	c58006a.ada
c46033a.ada	c490003.a	c52104g.ada	c58006b.ada
c46041a.ada	c49020a.ada	c52104h.ada	c59002a.ada
c46042a.ada	c49021a.ada	c52104k.ada	c59002b.ada
c46043b.ada	c49022a.ada	c52104l.ada	c59002c.ada
c46044b.ada	c49022b.ada	c52104m.ada	c61008a.ada
c46051a.ada	c49022c.ada	c52104p.ada	c61009a.ada
c46051b.ada	c49023a.ada	c52104q.ada	c61010a.ada
c46051c.ada	c49024a.ada	c52104r.ada	c62002a.ada
c46052a.ada	c49025a.ada	c52104x.ada	c62003a.ada
c46053a.ada	c49026a.ada	c52104y.ada	c62003b.ada
c46054a.ada	c4a005b.ada	c53007a.ada	c62004a.ada
c460a01.a	c4a006a.ada	c540001.a	c62006a.ada
c460a02.a	c4a007a.tst	c54a03a.ada	c631001.a
c47002a.ada	c4a010a.ada	c54a04a.ada	c640001.a
c47002b.ada	c4a010b.ada	c54a07a.ada	c64002b.ada
c47002c.ada	c4a011a.ada	c54a13a.ada	c64004g.ada
c47002d.ada	c4a012b.ada	c54a13b.ada	c64005a.ada
c47003a.ada	c4a013a.ada	c54a13c.ada	c64005b.ada
c47004a.ada	c4a014a.ada	c54a13d.ada	c64005c.ada
c47005a.ada	c51004a.ada	c54a22a.ada	c64005d0.ada
c47006a.ada	c52005a.ada	c54a23a.ada	c64005da.ada
c47007a.ada	c52005b.ada	c54a24a.ada	c64005db.ada

c64005dc.ada	c66002g.ada	c761003.a	c85005b.ada
c641001.a	c67002a.ada	c761004.a	c85005c.ada
c64103b.ada	c67002b.ada	c761005.a	c85005d.ada
c64103c.ada	c67002c.ada	c761006.a	c85005e.ada
c64103d.ada	c67002d.ada	c761007.a	c85005f.ada
c64103e.ada	c67002e.ada	c761010.a	c85005g.ada
c64103f.ada	c67003f.ada	c83007a.ada	c85006a.ada
c64104a.ada	c67005a.ada	c83012d.ada	c85006b.ada
c64104b.ada	c67005b.ada	c83022a.ada	c85006c.ada
c64104c.ada	c67005c.ada	c83022g0.ada	c85006d.ada
c64104d.ada	c67005d.ada	c83022g1.ada	c85006e.ada
c64104e.ada	c72001b.ada	c83023a.ada	c85006f.ada
c64104f.ada	c72002a.ada	c83024a.ada	c85006g.ada
c64104g.ada	c730001.a	c83024e0.ada	c85007a.ada
c64104h.ada	c730002.a	c83024e1.ada	c85007e.ada
c64104i.ada	c730003.a	c83025a.ada	c85009a.ada
c64104j.ada	c730004.a	c83025c.ada	c85011a.ada
c64104k.ada	c73002a.ada	c83027a.ada	c85013a.ada
c64104l.ada	c730a01.a	c83027c.ada	c85014a.ada
c64104m.ada	c730a02.a	c83028a.ada	c85014b.ada
c64104n.ada	c731001.a	c83029a.ada	c85014c.ada
c64104o.ada	c74004a.ada	c83030a.ada	c85017a.ada
c64105a.ada	c74203a.ada	c83030c.ada	c85018a.ada
c64105b.ada	c74206a.ada	c83031a.ada	c85018b.ada
c64105c.ada	c74207b.ada	c83031c.ada	c85019a.ada
c64105d.ada	c74208a.ada	c83031e.ada	c854001.a
c64106a.ada	c74208b.ada	c83032a.ada	c854002.a
c64106b.ada	c74209a.ada	c83033a.ada	c86003a.ada
c64106c.ada	c74210a.ada	c83051a.ada	c86004a.ada
c64106d.ada	c74211a.ada	c83b02a.ada	c86004b0.ada
c64107a.ada	c74211b.ada	c83b02b.ada	c86004b1.ada
c64108a.ada	c74302a.ada	c83e02a.ada	c86004b2.ada
c64109a.ada	c74302b.ada	c83e02b.ada	c86004c0.ada
c64109b.ada	c74305a.ada	c83e03a.ada	c86004c1.ada
c64109c.ada	c74305b.ada	c83f01a.ada	c86004c2.ada
c64109d.ada	c74306a.ada	c83f01b.ada	c86006i.ada
c64109e.ada	c74307a.ada	c83f01c0.ada	c86007a.ada
c64109f.ada	c74401d.ada	c83f01c1.ada	c87a05a.ada
c64109g.ada	c74401e.ada	c83f01c2.ada	c87a05b.ada
c64109h.ada	c74401k.ada	c83f01d0.ada	c87b02a.ada
c64109i.ada	c74401q.ada	c83f01d1.ada	c87b02b.ada
c64109j.ada	c74402a.ada	c83f03a.ada	c87b03a.ada
c64109k.ada	c74402b.ada	c83f03b.ada	c87b04a.ada
c64109l.ada	c74406a.ada	c83f03c0.ada	c87b04b.ada
c64201b.ada	c74407b.ada	c83f03c1.ada	c87b04c.ada
c64201c.ada	c74409b.ada	c83f03c2.ada	c87b05a.ada
c64202a.ada	c760001.a	c83f03d0.ada	c87b06a.ada
c650001.a	c760002.a	c83f03d1.ada	c87b07a.ada
c65003a.ada	c760007.a	c840001.a	c87b07b.ada
c65003b.ada	c760009.a	c84002a.ada	c87b07c.ada
c66002a.ada	c760010.a	c84005a.ada	c87b07d.ada
c66002c.ada	c760011.a	c84008a.ada	c87b07e.ada
c66002d.ada	c760012.a	c84009a.ada	c87b08a.ada
c66002e.ada	c761001.a	c85004b.ada	c87b09a.ada
c66002f.ada	c761002.a	c85005a.ada	c87b09c.ada

c87b10a.ada	c92002a.ada	c94005b.ada	c95085i.ada
c87b11a.ada	c92003a.ada	c94006a.ada	c95085j.ada
c87b11b.ada	c92005a.ada	c94007a.ada	c95085k.ada
c87b13a.ada	c92005b.ada	c94007b.ada	c95085l.ada
c87b14a.ada	c92006a.ada	c94008a.ada	c95085m.ada
c87b14b.ada	c930001.a	c94008b.ada	c95085n.ada
c87b14c.ada	c93001a.ada	c94008c.ada	c95085o.ada
c87b14d.ada	c93002a.ada	c94008d.ada	c95086a.ada
c87b15a.ada	c93003a.ada	c94010a.ada	c95086b.ada
c87b16a.ada	c93004a.ada	c94011a.ada	c95086c.ada
c87b17a.ada	c93004b.ada	c94020a.ada	c95086d.ada
c87b18a.ada	c93004c.ada	c940a03.a	c95086e.ada
c87b18b.ada	c93004d.ada	c95008a.ada	c95086f.ada
c87b19a.ada	c93004f.ada	c95009a.ada	c95087a.ada
c87b23a.ada	c93005a.ada	c95010a.ada	c95087b.ada
c87b24a.ada	c93005b.ada	c95011a.ada	c95087c.ada
c87b24b.ada	c93005c.ada	c95012a.ada	c95087d.ada
c87b26b.ada	c93005d.ada	c95021a.ada	c95088a.ada
c87b27a.ada	c93005e.ada	c95022a.ada	c95089a.ada
c87b28a.ada	c93005f.ada	c95022b.ada	c95090a.ada
c87b29a.ada	c93005g.ada	c95033a.ada	c95092a.ada
c87b30a.ada	c93005h.ada	c95033b.ada	c95093a.ada
c87b31a.ada	c93006a.ada	c95034a.ada	c95095a.ada
c87b32a.ada	c93007a.ada	c95034b.ada	c95095b.ada
c87b33a.ada	c93008a.ada	c95035a.ada	c95095c.ada
c87b34a.ada	c93008b.ada	c95040a.ada	c95095d.ada
c87b34b.ada	c940001.a	c95040b.ada	c95095e.ada
c87b34c.ada	c940002.a	c95040c.ada	c951001.a
c87b35c.ada	c940004.a	c95040d.ada	c951002.a
c87b38a.ada	c940005.a	c95041a.ada	c953001.a
c87b39a.ada	c940006.a	c95065a.ada	c953002.a
c87b40a.ada	c940007.a	c95065b.ada	c953003.a
c87b41a.ada	c940010.a	c95065c.ada	c954001.a
c87b42a.ada	c940011.a	c95065d.ada	c954010.a
c87b43a.ada	c940012.a	c95065e.ada	c954011.a
c87b44a.ada	c940013.a	c95065f.ada	c954012.a
c87b45a.ada	c940014.a	c95066a.ada	c954013.a
c87b45c.ada	c940015.a	c95067a.ada	c954014.a
c87b47a.ada	c940016.a	c95071a.ada	c954015.a
c87b48a.ada	c94001a.ada	c95072a.ada	c954016.a
c87b48b.ada	c94001b.ada	c95072b.ada	c954017.a
c87b50a.ada	c94001c.ada	c95073a.ada	c954018.a
c87b54a.ada	c94001e.ada	c95074c.ada	c954019.a
c87b57a.ada	c94001f.ada	c95076a.ada	c954020.a
c87b62a.ada	c94001g.ada	c95078a.ada	c954021.a
c87b62b.ada	c94002a.ada	c95080b.ada	c954022.a
c87b62c.ada	c94002b.ada	c95082g.ada	c954023.a
c87b62d.tst	c94002d.ada	c95085a.ada	c954024.a
c910001.a	c94002e.ada	c95085b.ada	c954025.a
c910002.a	c94002f.ada	c95085c.ada	c954026.a
c910003.a	c94002g.ada	c95085d.ada	c954a01.a
c91004b.ada	c94004a.ada	c95085e.ada	c954a02.a
c91004c.ada	c94004b.ada	c95085f.ada	c954a03.a
c91006a.ada	c94004c.ada	c95085g.ada	c960001.a
c91007a.ada	c94005a.ada	c95085h.ada	c960002.a

c960004.a
c96001a.ada
c96004a.ada
c96005a.ada
c96005b.tst
c96005d.ada
c96005f.ada
c96006a.ada
c96007a.ada
c96008a.ada
c96008b.ada
c97112a.ada
c97113a.ada
c97114a.ada
c97115a.ada
c97116a.ada
c97117a.ada
c97117b.ada
c97117c.ada
c97118a.ada
c97120a.ada
c97120b.ada
c97201a.ada
c97201b.ada
c97201c.ada
c97201d.ada
c97201e.ada
c97201g.ada
c97201h.ada
c97201x.ada
c97202a.ada
c97203a.ada
c97203b.ada
c97203c.ada
c97204a.ada
c97204b.ada
c97205a.ada
c97205b.ada
c97301a.ada
c97301b.ada
c97301c.ada
c97301d.ada
c97301e.ada
c97302a.ada
c97303a.ada
c97303b.ada
c97303c.ada
c97304a.ada
c97304b.ada
c97305a.ada
c97305b.ada
c97305c.ada
c97305d.ada
c97307a.ada
c974001.a
c974002.a
c974003.a
c974004.a
c974005.a
c974006.a
c974007.a
c974008.a
c974009.a
c974010.a
c974011.a
c974012.a
c974013.a
c974014.a
c980001.a
c980002.a
c980003.a
c99004a.ada
c99005a.ada
c9a003a.ada
c9a004a.ada
c9a007a.ada
c9a009a.ada
c9a009c.ada
c9a009f.ada
c9a009g.ada
c9a009h.ada
c9a010a.ada
c9a011a.ada
c9a011b.ada
ca1003a.ada
ca1004a.ada
ca1005a.ada
ca1006a.ada
ca1011a0.ada
ca1011a1.ada
ca1011a2.ada
ca1011a3.ada
ca1011a4.ada
ca1011a5.ada
ca1011a6.ada
ca1012a0.ada
ca1012a1.ada
ca1012a2.ada
ca1012a3.ada
ca1012a4.ada
ca1012b0.ada
ca1012b2.ada
ca1012b4.ada
ca1013a0.ada
ca1013a1.ada
ca1013a2.ada
ca1013a3.ada
ca1013a4.ada
ca1013a5.ada
ca1013a6.ada
ca1014a0.ada
ca1014a1.ada
ca1014a2.ada
ca1014a3.ada
ca1020e0.ada
ca1020e1.ada
ca1020e2.ada
ca1020e3.ada
ca1022a0.ada
ca1022a1.ada
ca1022a2.ada
ca1022a3.ada
ca1022a4.ada
ca1022a5.ada
ca1022a6.ada
ca11001.a
ca11002.a
ca11003.a
ca110040.a
ca110041.a
ca110042.am
ca110050.a
ca110051.am
ca11006.a
ca11007.a
ca11008.a
ca11009.a
ca11010.a
ca11011.a
ca11012.a
ca11013.a
ca11014.a
ca11015.a
ca11016.a
ca11017.a
ca11018.a
ca11019.a
ca11020.a
ca11021.a
ca11022.a
ca1102a0.ada
ca1102a1.ada
ca1102a2.ada
ca1106a.ada
ca1108a.ada
ca1108b.ada
ca11a01.a
ca11a02.a
ca11b01.a
ca11b02.a
ca11c01.a
ca11c02.a
ca11c03.a
ca11d010.a
ca11d011.a
ca11d012.a
ca11d013.am
ca11d02.a
ca11d03.a
ca13001.a
ca13002.a
ca13003.a
ca13a01.a
ca13a02.a
ca140230.a
ca140231.a
ca140232.am
ca140233.a
ca140280.a
ca140281.a
ca140282.a
ca140283.am
ca15003.a
ca200020.a
ca200021.a
ca200022.am
ca2001h0.ada
ca2001h1.ada
ca2001h2.ada
ca2001h3.ada
ca2002a0.ada
ca2002a1.ada
ca2002a2.ada
ca2003a0.ada
ca2003a1.ada
ca2004a0.ada
ca2004a1.ada
ca2004a2.ada
ca2004a3.ada
ca2004a4.ada
ca2007a0.ada
ca2007a1.ada
ca2007a2.ada
ca2007a3.ada
ca2008a0.ada
ca2008a1.ada
ca2008a2.ada
ca2009a.ada
ca2009c0.ada
ca2009c1.ada
ca2009d.ada
ca2009f0.ada
ca2009f1.ada
ca2009f2.ada
ca2011b.ada
ca21001.a
ca3011a0.ada
ca3011a1.ada
ca3011a2.ada
ca3011a3.ada

ca3011a4.ada	cb40a031.am	cc3017c.ada	cc51004.a
ca5003a0.ada	cb40a04.a	cc3019a.ada	cc51006.a
ca5003a1.ada	cb41001.a	cc3019b0.ada	cc51007.a
ca5003a2.ada	cb41002.a	cc3019b1.ada	cc51a01.a
ca5003a3.ada	cb41003.a	cc3019b2.ada	cc51b03.a
ca5003a4.ada	cb41004.a	cc3019c0.ada	cc51d01.a
ca5003a5.ada	cb5001a.ada	cc3019c1.ada	cc51d02.a
ca5003a6.ada	cb5001b.ada	cc3019c2.ada	cc54001.a
ca5003b0.ada	cb5002a.ada	cc3106b.ada	cc54002.a
ca5003b1.ada	cc1004a.ada	cc3120a.ada	cc54003.a
ca5003b2.ada	cc1005b.ada	cc3120b.ada	cc54004.a
ca5003b3.ada	cc1010a.ada	cc3121a.ada	cc70001.a
ca5003b4.ada	cc1010b.ada	cc3123a.ada	cc70002.a
ca5003b5.ada	cc1018a.ada	cc3125a.ada	cc70003.a
ca5004a.ada	cc1104c.ada	cc3125b.ada	cc70a01.a
ca5004b0.ada	cc1107b.ada	cc3125c.ada	cc70a02.a
ca5004b1.ada	cc1111a.ada	cc3125d.ada	cc70b01.a
ca5004b2.ada	cc1204a.ada	cc3126a.ada	cc70b02.a
ca5006a.ada	cc1207b.ada	cc3127a.ada	cc70c01.a
cb10002.a	cc1220a.ada	cc3128a.ada	cc70c02.a
cb1001a.ada	cc1221a.ada	cc3203a.ada	cd10001.a
cb1004a.ada	cc1221b.ada	cc3207b.ada	cd1009a.ada
cb1005a.ada	cc1221c.ada	cc3220a.ada	cd1009b.ada
cb1010a.ada	cc1221d.ada	cc3221a.ada	cd1009d.ada
cb1010c.ada	cc1222a.ada	cc3222a.ada	cd1009e.ada
cb1010d.ada	cc1223a.ada	cc3223a.ada	cd1009f.ada
cb20001.a	cc1224a.ada	cc3224a.ada	cd1009g.ada
cb20003.a	cc1225a.tst	cc3225a.ada	cd1009h.ada
cb20004.a	cc1226b.ada	cc3230a.ada	cd1009i.ada
cb20005.a	cc1227a.ada	cc3231a.ada	cd1009j.ada
cb20006.a	cc1301a.ada	cc3232a.ada	cd1009k.tst
cb20007.a	cc1302a.ada	cc3233a.ada	cd1009l.ada
cb2004a.ada	cc1304a.ada	cc3234a.ada	cd1009m.ada
cb2005a.ada	cc1304b.ada	cc3235a.ada	cd1009n.ada
cb2006a.ada	cc1307a.ada	cc3236a.ada	cd1009o.ada
cb2007a.ada	cc1307b.ada	cc3240a.ada	cd1009p.ada
cb20a02.a	cc1308a.ada	cc3305a.ada	cd1009q.ada
cb3003a.ada	cc1310a.ada	cc3305b.ada	cd1009r.ada
cb3003b.ada	cc1311a.ada	cc3305c.ada	cd1009s.ada
cb3004a.ada	cc1311b.ada	cc3305d.ada	cd1009t.tst
cb40005.a	cc2002a.ada	cc3601a.ada	cd1009u.tst
cb4001a.ada	cc30001.a	cc3601c.ada	cd1009v.ada
cb4002a.ada	cc30002.a	cc3602a.ada	cd1009w.ada
cb4003a.ada	cc3004a.ada	cc3603a.ada	cd1009x.ada
cb4004a.ada	cc3007a.ada	cc3605a.ada	cd1009y.ada
cb4005a.ada	cc3007b.ada	cc3606a.ada	cd1009z.ada
cb4006a.ada	cc3011a.ada	cc3606b.ada	cd1c03a.ada
cb4007a.ada	cc3011d.ada	cc3607b.ada	cd1c03b.ada
cb4008a.ada	cc3012a.ada	cc40001.a	cd1c03c.ada
cb4009a.ada	cc3015a.ada	cc50001.a	cd1c03e.tst
cb4013a.ada	cc3016b.ada	cc50a01.a	cd1c03f.ada
cb40a01.a	cc3016c.ada	cc50a02.a	cd1c03g.ada
cb40a020.a	cc3016f.ada	cc51001.a	cd1c03h.ada
cb40a021.am	cc3016i.ada	cc51002.a	cd1c03i.ada
cb40a030.a	cc3017b.ada	cc51003.a	cd1c04a.ada

cd1c04d.ada	cd3015f.ada	cd5014t.ada	ce2103c.ada
cd1c04e.ada	cd3015g.ada	cd5014v.ada	ce2103d.ada
cd1c06a.tst	cd3015h.ada	cd5014x.ada	ce2104a.ada
cd20001.a	cd3015i.ada	cd5014y.ada	ce2104b.ada
cd2a21a.ada	cd3015k.ada	cd5014z.ada	ce2104c.ada
cd2a21c.ada	cd3021a.ada	cd70001.a	ce2104d.ada
cd2a21e.ada	cd33001.a	cd7002a.ada	ce2106a.ada
cd2a22a.ada	cd33002.a	cd7007b.ada	ce2106b.ada
cd2a22e.ada	cd40001.a	cd7101d.ada	ce2108e.ada
cd2a22i.ada	cd4031a.ada	cd7101e.dep	ce2108f.ada
cd2a22j.ada	cd4041a.tst	cd7101f.dep	ce2108g.ada
cd2a23a.ada	cd4051a.ada	cd7101g.tst	ce2108h.ada
cd2a23e.ada	cd4051b.ada	cd7103d.ada	ce2109a.ada
cd2a24a.ada	cd4051c.ada	cd7202a.ada	ce2109b.ada
cd2a24e.ada	cd4051d.ada	cd7204b.ada	ce2109c.ada
cd2a24i.ada	cd5003a.ada	cd7204c.ada	ce2110a.ada
cd2a24j.ada	cd5003b.ada	cd72a01.a	ce2110c.ada
cd2a31a.ada	cd5003c.ada	cd72a02.a	ce2111a.ada
cd2a31c.ada	cd5003d.ada	cd7305a.ada	ce2111b.ada
cd2a31e.ada	cd5003e.ada	cd90001.a	ce2111c.ada
cd2a32a.ada	cd5003f.ada	cd92001.a	ce2111e.ada
cd2a32c.ada	cd5003g.ada	cda201a.ada	ce2111f.ada
cd2a32e.ada	cd5003h.ada	cda201b.ada	ce2111g.ada
cd2a32g.ada	cd5003i.ada	cda201c.ada	ce2111i.ada
cd2a32i.ada	cd5011a.ada	cda201e.ada	ce2120a.tst
cd2a32j.ada	cd5011c.ada	cdb0a01.a	ce2120b.tst
cd2a51a.ada	cd5011e.ada	cdb0a02.a	ce2201a.ada
cd2a53a.ada	cd5011g.ada	cde0001.a	ce2201b.ada
cd2a53e.ada	cd5011i.ada	ce2102a.ada	ce2201c.ada
cd2a83c.tst	cd5011k.ada	ce2102b.ada	ce2201d.dep
cd2a91c.tst	cd5011m.ada	ce2102c.tst	ce2201e.dep
cd2b11a.ada	cd5011q.ada	ce2102d.ada	ce2201f.ada
cd2b11b.ada	cd5011s.ada	ce2102e.ada	ce2201g.ada
cd2b11d.ada	cd5012a.ada	ce2102f.ada	ce2201h.ada
cd2b11e.ada	cd5012b.ada	ce2102g.ada	ce2201i.ada
cd2b11f.ada	cd5012e.ada	ce2102h.tst	ce2201j.ada
cd2b15c.ada	cd5012f.ada	ce2102i.ada	ce2201k.ada
cd2b16a.ada	cd5012i.ada	ce2102j.ada	ce2201l.ada
cd2c11a.tst	cd5012m.ada	ce2102k.ada	ce2201m.ada
cd2c11d.tst	cd5013a.ada	ce2102l.ada	ce2201n.ada
cd2d11a.ada	cd5013c.ada	ce2102m.ada	ce2202a.ada
cd2d13a.ada	cd5013e.ada	ce2102n.ada	ce2203a.tst
cd30001.a	cd5013g.ada	ce2102o.ada	ce2204a.ada
cd30002.a	cd5013i.ada	ce2102p.ada	ce2204b.ada
cd30003.a	cd5013k.ada	ce2102q.ada	ce2204c.ada
cd30004.a	cd5013m.ada	ce2102r.ada	ce2204d.ada
cd300050.am	cd5013o.ada	ce2102s.ada	ce2205a.ada
cd300051.c	cd5014a.ada	ce2102t.ada	ce2206a.ada
cd3014a.ada	cd5014c.ada	ce2102u.ada	ce2208b.ada
cd3014c.ada	cd5014e.ada	ce2102v.ada	ce2401a.ada
cd3014d.ada	cd5014g.ada	ce2102w.ada	ce2401b.ada
cd3014f.ada	cd5014i.ada	ce2102x.ada	ce2401c.ada
cd3015a.ada	cd5014k.ada	ce2102y.ada	ce2401e.ada
cd3015c.ada	cd5014m.ada	ce2103a.tst	ce2401f.ada
cd3015e.ada	cd5014o.ada	ce2103b.tst	ce2401h.ada

ce2401i.ada
ce2401j.ada
ce2401k.ada
ce2401l.ada
ce2402a.ada
ce2403a.tst
ce2404a.ada
ce2404b.ada
ce2405b.ada
ce2406a.ada
ce2407a.ada
ce2407b.ada
ce2408a.ada
ce2408b.ada
ce2409a.ada
ce2409b.ada
ce2410a.ada
ce2410b.ada
ce2411a.ada
ce3002b.tst
ce3002c.tst
ce3002d.ada
ce3002f.ada
ce3102a.ada
ce3102b.tst
ce3102d.ada
ce3102e.ada
ce3102f.ada
ce3102g.ada
ce3102h.ada
ce3102i.ada
ce3102j.ada
ce3102k.ada
ce3103a.ada
ce3104a.ada
ce3104b.ada
ce3104c.ada
ce3106a.ada
ce3106b.ada
ce3107a.tst
ce3107b.ada
ce3108a.ada
ce3108b.ada
ce3110a.ada
ce3112c.ada
ce3112d.ada
ce3114a.ada
ce3115a.ada
ce3119a.tst
ce3201a.ada
ce3202a.ada
ce3206a.ada
ce3207a.ada
ce3301a.ada
ce3302a.ada
ce3303a.ada
ce3304a.tst
ce3305a.ada
ce3306a.ada
ce3401a.ada
ce3402a.ada
ce3402c.ada
ce3402d.ada
ce3402e.ada
ce3403a.ada
ce3403b.ada
ce3403c.ada
ce3403d.ada
ce3403e.ada
ce3403f.ada
ce3404a.ada
ce3404b.ada
ce3404c.ada
ce3404d.ada
ce3405a.ada
ce3405c.ada
ce3405d.ada
ce3406a.ada
ce3406b.ada
ce3406c.ada
ce3406d.ada
ce3407a.ada
ce3407b.ada
ce3407c.ada
ce3408a.ada
ce3408b.ada
ce3408c.ada
ce3409a.ada
ce3409b.ada
ce3409c.ada
ce3409d.ada
ce3409e.ada
ce3410a.ada
ce3410b.ada
ce3410c.ada
ce3410d.ada
ce3410e.ada
ce3411a.ada
ce3411c.ada
ce3412a.ada
ce3413a.ada
ce3413b.ada
ce3413c.ada
ce3414a.ada
ce3601a.ada
ce3602a.ada
ce3602b.ada
ce3602c.ada
ce3602d.ada
ce3603a.ada
ce3604a.ada
ce3604b.ada
ce3605a.ada
ce3605b.ada
ce3605c.ada
ce3605d.ada
ce3605e.ada
ce3606a.ada
ce3606b.ada
ce3701a.ada
ce3704a.ada
ce3704b.ada
ce3704c.ada
ce3704d.ada
ce3704e.ada
ce3704f.ada
ce3704m.ada
ce3704n.ada
ce3704o.ada
ce3705a.ada
ce3705b.ada
ce3705c.ada
ce3705d.ada
ce3705e.ada
ce3706c.ada
ce3706d.ada
ce3706f.ada
ce3706g.ada
ce3707a.ada
ce3708a.ada
ce3801a.ada
ce3801b.ada
ce3804a.ada
ce3804b.ada
ce3804c.ada
ce3804d.ada
ce3804e.ada
ce3804f.ada
ce3804g.ada
ce3804h.ada
ce3804i.ada
ce3804j.ada
ce3804m.ada
ce3804o.ada
ce3804p.ada
ce3805a.ada
ce3805b.ada
ce3806a.ada
ce3806b.ada
ce3806c.ada
ce3806d.ada
ce3806e.ada
ce3806f.ada
ce3806g.ada
ce3806h.ada
ce3809a.ada
ce3809b.ada
ce3810a.ada
ce3810b.ada
ce3815a.ada
ce3901a.ada
ce3902b.ada
ce3904a.ada
ce3904b.ada
ce3905a.ada
ce3905b.ada
ce3905c.ada
ce3905l.ada
ce3906a.ada
ce3906b.ada
ce3906c.ada
ce3906d.ada
ce3906e.ada
ce3906f.ada
ce3907a.ada
ce3908a.ada
cxa3001.a
cxa3002.a
cxa3003.a
cxa3004.a
cxa4001.a
cxa4002.a
cxa4003.a
cxa4004.a
cxa4005.a
cxa4006.a
cxa4007.a
cxa4008.a
cxa4009.a
cxa4010.a
cxa4011.a
cxa4012.a
cxa4013.a
cxa4014.a
cxa4015.a
cxa4016.a
cxa4017.a
cxa4018.a
cxa4019.a
cxa4020.a
cxa4021.a
cxa4022.a
cxa4023.a
cxa4024.a
cxa4025.a
cxa4026.a
cxa4027.a
cxa4028.a
cxa4029.a
cxa4030.a

cxa4031.a	cxb3002.a	la140010.a	la140152.am
cxa4032.a	cxb3003.a	la140011.am	la140153.a
cxa4033.a	cxb30040.c	la140012.a	la140160.a
cxa5011.a	cxb30041.am	la140020.a	la140161.a
cxa5012.a	cxb3005.a	la140021.am	la140162.am
cxa5013.a	cxb30060.c	la140022.a	la140163.a
cxa5015.a	cxb30061.am	la140030.a	la140170.a
cxa5a01.a	cxb3007.a	la140031.a	la140171.a
cxa5a02.a	cxb3008.a	la140032.am	la140172.am
cxa5a03.a	cxb3009.a	la140033.a	la140173.a
cxa5a04.a	cxb3010.a	la140040.a	la140180.a
cxa5a05.a	cxb3011.a	la140041.am	la140181.a
cxa5a06.a	cxb3012.a	la140042.a	la140182.am
cxa5a07.a	cxb30130.c	la140050.a	la140183.a
cxa5a08.a	cxb30131.c	la140051.a	la140190.a
cxa5a09.a	cxb30132.am	la140052.am	la140191.a
cxa5a10.a	cxb3014.a	la140053.a	la140192.am
cxa8001.a	cxb3015.a	la140060.a	la140193.a
cxa8002.a	cxb3016.a	la140061.a	la140200.a
cxa8003.a	cxb4001.a	la140062.am	la140201.a
cxa9001.a	cxb4002.a	la140063.a	la140202.am
cxa9002.a	cxb4003.a	la140070.a	la140203.a
cxaa001.a	cxb4004.a	la140071.a	la140210.a
cxaa002.a	cxb4005.a	la140072.am	la140211.am
cxaa003.a	cxb4006.a	la140073.a	la140212.a
cxaa004.a	cxb4007.a	la140080.a	la140220.a
cxaa005.a	cxb4008.a	la140081.a	la140221.am
cxaa006.a	cxb40090.cbl	la140082.am	la140222.a
cxaa007.a	cxb40091.cbl	la140083.a	la140240.a
cxaa008.a	cxb40092.cbl	la140090.a	la140241.a
cxaa009.a	cxb40093.am	la140091.a	la140242.am
cxaa010.a	cxb5001.a	la140092.am	la140243.a
cxaa011.a	cxb5002.a	la140093.a	la140250.a
cxaa012.a	cxb5003.a	la140100.a	la140251.am
cxaa013.a	cxb50040.ftn	la140101.a	la140252.a
cxaa014.a	cxb50041.ftn	la140102.am	la140260.a
cxaa015.a	cxb50042.am	la140103.a	la140261.a
cxaa016.a	cxb50050.ftn	la140110.a	la140262.am
cxaa017.a	cxb50051.ftn	la140111.a	la140263.a
cxaa018.a	cxb50052.am	la140112.am	la140270.a
cxab001.a	d4a002a.ada	la140113.a	la140271.a
cxac001.a	d4a002b.ada	la140120.a	la140272.am
cxac002.a	d4a004a.ada	la140121.a	la140273.a
cxac003.a	d4a004b.ada	la140122.am	la200010.a
cxac004.a	e28002b.ada	la140123.a	la200011.a
cxaca01.a	e28005d.ada	la140130.a	la200012.am
cxaca02.a	e52103y.ada	la140131.a	la5001a0.ada
cxacb01.a	eb4011a.ada	la140132.am	la5001a1.ada
cxacb02.a	eb4012a.ada	la140133.a	la5001a2.ada
cxacc01.a	eb4014a.ada	la140140.a	la5001a3.ada
cxaf001.a	ee3203a.ada	la140141.a	la5001a4.ada
cxb2001.a	ee3204a.ada	la140142.am	la5001a5.ada
cxb2002.a	ee3402b.ada	la140143.a	la5001a6.ada
cxb2003.a	ee3409f.ada	la140150.a	la5001a7.ada
cxb3001.a	ee3412c.ada	la140151.a	la5007a0.ada

la5007a1.ada	la5007e0.ada	la5008a1.ada	la5008e0.ada
la5007b0.ada	la5007e1.ada	la5008b0.ada	la5008e1.ada
la5007b1.ada	la5007f0.ada	la5008b1.ada	la5008f0.ada
la5007c0.ada	la5007f1.ada	la5008c0.ada	la5008f1.ada
la5007c1.ada	la5007g0.ada	la5008c1.ada	la5008g0.ada
la5007d0.ada	la5007g1.ada	la5008d0.ada	la5008g1.ada
la5007d1.ada	la5008a0.ada	la5008d1.ada	

4.11 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	cxc3008.a	cxda003.a	cxg2005.a
bxc3002.a	cxc3009.a	cxda004.a	cxg2006.a
bxc5001.a	cxc6001.a	cxdb001.a	cxg2007.a
bxc6001.a	cxc6002.a	cxdb002.a	cxg2008.a
bxc6002.a	cxc6003.a	cxdb003.a	cxg2009.a
bxc6003.a	cxc7001.a	cxdb004.a	cxg2010.a
bxc6a01.a	cxc7002.a	cxe1001.a	cxg2011.a
bxc6a02.a	cxc7003.a	cxe2001.a	cxg2012.a
bxc6a03.a	cxc7004.a	cxe2002.a	cxg2013.a
bxc6a04.a	cxd1001.a	cxe4001.a	cxg2014.a
bxd1001.a	cxd1002.a	cxe4002.a	cxg2015.a
bxd1002.a	cxd1003.a	cxe4003.a	cxg2016.a
bxe2007.a	cxd1004.a	cxe4004.a	cxg2017.a
bxe2008.a	cxd1005.a	cxe4005.a	cxg2018.a
bxe2009.a	cxd1006.a	cxe4006.a	cxg2019.a
bxe2010.a	cxd1007.a	cxe5001.a	cxg2020.a
bxe2011.a	cxd1008.a	cxe5002.a	cxg2021.a
bxe2012.a	cxd2001.a	cxe5003.a	cxg2022.a
bxe2013.a	cxd2002.a	cxf1001.a	cxg2023.a
bxe2a01.a	cxd2003.a	cxf2001.a	cxg2024.a
bxe2a02.a	cxd2004.a	cxf2002.a	cxh1001.a
bxe2a03.a	cxd2006.a	cxf2003.a	cxh3001.a
bxe2a04.a	cxd2007.a	cxf2004.a	cxh3002.a
bxe2a05.a	cxd2008.a	cxf2005.a	cxh30030.a
bxe2a06.a	cxd3001.a	cxf2a01.a	cxh30031.am
bxe4001.a	cxd3002.a	cxf2a02.a	lxd70010.a
bxf1001.a	cxd3003.a	cxf3001.a	lxd70011.a
bxh4001.a	cxd4001.a	cxf3002.a	lxd70012.am
bxh4002.a	cxd4002.a	cxf3003.a	lxd70030.a
bxh4003.a	cxd4003.a	cxf3004.a	lxd70031.a
bxh4004.a	cxd4004.a	cxf3a01.a	lxd70032.am
bxh4005.a	cxd4005.a	cxf3a02.a	lxd70040.a
bxh4006.a	cxd4006.a	cxf3a03.a	lxd70041.a
bxh4007.a	cxd4007.a	cxf3a04.a	lxd70042.am
bxh4008.a	cxd4008.a	cxf3a05.a	lxd70050.a
bxh4009.a	cxd4009.a	cxf3a06.a	lxd70051.a
bxh4010.a	cxd4010.a	cxf3a07.a	lxd70052.am
bxh4011.a	cxd5001.a	cxf3a08.a	lxd70060.a
bxh4012.a	cxd6001.a	cxg1001.a	lxd70061.a
bxh4013.a	cxd6002.a	cxg1002.a	lxd70062.am
cxc3001.a	cxd6003.a	cxg1003.a	lxd70070.a
cxc3002.a	cxd8001.a	cxg1004.a	lxd70071.a
cxc3003.a	cxd8002.a	cxg1005.a	lxd70072.am
cxc3004.a	cxd8003.a	cxg2001.a	lxd70080.a
cxc3005.a	cxd9001.a	cxg2002.a	lxd70081.a
cxc3006.a	cxda001.a	cxg2003.a	lxd70082.am
cxc3007.a	cxda002.a	cxg2004.a	lxd70090.a

lxd70091.a	lxh40033.am	lxh40072.a	lxh40110.a
lxd70092.am	lxh40040.a	lxh40073.am	lxh40111.a
lxe30010.am	lxh40041.a	lxh40080.a	lxh40112.am
lxe30011.am	lxh40042.a	lxh40081.a	lxh40120.a
lxe30020.am	lxh40043.am	lxh40082.a	lxh40121.a
lxe30021.am	lxh40050.a	lxh40083.a	lxh40122.a
lxh40010.a	lxh40051.a	lxh40084.am	lxh40123.am
lxh40011.a	lxh40052.a	lxh40090.a	lxh40130.a
lxh40012.am	lxh40053.am	lxh40091.a	lxh40131.a
lxh40020.a	lxh40060.a	lxh40092.a	lxh40132.a
lxh40021.a	lxh40061.a	lxh40093.am	lxh40133.am
lxh40022.am	lxh40062.a	lxh40100.a	lxh40140.a
lxh40030.a	lxh40063.am	lxh40101.a	lxh40141.a
lxh40031.a	lxh40070.a	lxh40102.a	lxh40142.am
lxh40032.a	lxh40071.a	lxh40103.am	

4.12 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
f340a001.a
f341a00.a
f390a00.a
f392a00.a
f392c00.a
f392d00.a
f393a00.a
f393b00.a
f3a2a00.a
f460a00.a
f730a000.a
f730a001.a
f731a00.a
f940a00.a
f954a00.a
fa11a00.a
fa11b00.a
fa11c00.a
fa11d00.a
fa13a00.a
fa13b00.a
fa21a00.a
fb20a00.a
fb40a00.a
fc50a00.a
fc51a00.a
fc51b00.a
fc51c00.a
fc51d00.a
fc54a00.a
fc70a00.a
fc70b00.a
fc70c00.a
fd72a00.a
fdb0a00.a
fxa5a00.a
fxaca00.a
fxacb00.a
fxacc00.a
fxc6a00.a
fxe2a00.a
fxf2a00.a
fxf3a00.a

4.13 Documentation Files

This section lists all the files containing ACATS 2.4 documentation. Files with the ".DOC" extension are in Microsoft Word 97 format. All other files are in ASCII text format.

coverage.txt
testobj.txt
ug-apxa.txt
ug-apxa.doc
ug-apxb.txt
ug-apxb.doc
ug-apxc.txt
ug-apxc.doc
ug-apxd.txt
ug-apxd.doc
ug-body.txt
ug-body.doc

4.14 Other Files

This section lists the files falling into the "others" category in Table 1 of the User's Guide. They are listed in the categories described in Section 3.2.1.

4.14.1 List of ACATS 2.4 Files

acats24.lst

4.14.2 Support Units Referenced by Many Tests

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

4.14.3 Preprocessing Tools and Data

macrosub.ada
widechr.a
macro.dfs
tsttests.dat

4.14.4 Tests for Reporting Code

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

4.15 Tests With Special Requirements

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

ba15001	cxd2003	cxg2004	la14012
bxc5001	cxd2004	cxg2006	la14013
bxh4001	cxd2006	cxg2007	la14014
bxh4002	cxd2007	cxg2008	la14015
bxh4003	cxd2008	cxg2009	la14016
bxh4004	cxd3001	cxg2010	la14017
bxh4005	cxd3002	cxg2011	la14018
bxh4006	cxd3003	cxg2012	la14019
bxh4007	cxd4001	cxg2013	la14020
bxh4008	cxd4003	cxg2014	la14021
bxh4009	cxd4004	cxg2015	la14022
bxh4010	cxd4005	cxg2016	la14024
bxh4011	cxd4006	cxg2017	la14025
bxh4012	cxd4007	cxg2018	la14026
bxh4013	cxd4008	cxg2019	la14027
c250001	cxd4009	cxg2020	lxd7009
c250002	cxd4010	cxg2021	lxe3001
cd30005	cxda003	cxg2022	lxe3002
cxb3004	cxda004	cxg2023	lxh4001
cxb3006	cxe1001	cxg2024	lxh4002
cxb3008	cxe2001	cxh1001	lxh4003
cxb3013	cxe2002	cxh3001	lxh4004
cxb4009	cxe4001	cxh3003	lxh4005
cxb5004	cxe4002	la14001	lxh4006
cxb5005	cxe4003	la14002	lxh4007
cxc3001	cxe4004	la14003	lxh4008
cxc3003	cxe4005	la14004	lxh4009
cxc3004	cxe4006	la14005	lxh4010
cxc3006	cxe5002	la14006	lxh4011
cxc3008	cxe5003	la14007	lxh4012
cxd1004	cxg1002	la14008	lxh4013
cxd1005	cxg1005	la14009	lxh4014
cxd2001	cxg2002	la14010	
cxd2002	cxg2003	la14011	

4.16 Test Files Added In ACATS 2.4

The following test files are new in ACATS 2.4:

b393007.a	b7310011.a	b7310014.a	c433001.a
b3a2016.a	b7310012.a	b7310015.a	
b7310010.a	b7310013.a	b7310016.am	

4.17 Test Files Modified For ACATS 2.4

The following test files have been modified from their ACATS 2.3 versions:

c392010.a	cb41002.a	cxb3015.a	lxh40112.am
c761010.a	cb41004.a	lxd70011.a	

4.18 Support Files Modified For ACATS 2.4

The following support files have been modified from their ACATS 2.3 versions:

repbody.ada
tctouch.ada

4.19 Test Files Deleted Since ACATS 2.3

The following test files were present in ACATS 2.3 but do not appear in ACATS 2.4:

None.

Appendix B: Parameterization Files

In ACATS 2.4, 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 tests written since the introduction of Ada 95, 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.

4.20 Macro Substitution File

The support file "MACRO.DFS" provides substitutions for special symbols that appear in certain ACATS 2.4 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
```

4.21 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 2.4 "support" directory as "TSTTESTS.DAT".

A26007A.TST	BD2C01D.TST	C35503F.TST	CE2102C.TST
AD8011A.TST	BD2C02A.TST	C45231D.TST	CE2102H.TST
B22001A.TST	BD2C03A.TST	C4A007A.TST	CE2103A.TST
B22001B.TST	BD4006A.TST	C87B62D.TST	CE2103B.TST
B22001C.TST	BD8001A.TST	C96005B.TST	CE2120A.TST
B22001D.TST	BD8002A.TST	CC1225A.TST	CE2120B.TST
B22001E.TST	BD8003A.TST	CD1009K.TST	CE2203A.TST
B22001F.TST	BD8004A.TST	CD1009T.TST	CE2403A.TST
B22001G.TST	BD8004B.TST	CD1009U.TST	CE3002B.TST
B22001I.TST	BD8004C.TST	CD1C03E.TST	CE3002C.TST
B22001J.TST	C23003A.TST	CD1C06A.TST	CE3102B.TST
B22001K.TST	C23003B.TST	CD2A83C.TST	CE3107A.TST
B22001L.TST	C23003G.TST	CD2A91C.TST	CE3119A.TST
B22001M.TST	C23003I.TST	CD2C11A.TST	CE3304A.TST
B22001N.TST	C35502D.TST	CD2C11D.TST	SPPRT13S.TST
B54B01B.TST	C35502F.TST	CD4041A.TST	
BD2A02A.TST	C35503D.TST	CD7101G.TST	

4.22 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 includes comments showing where such modifications are expected.

```
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
```

Appendix C:
Output 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.

4.23 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 2.4 01-02-26 14:46:13
---- CZ00004 Check that Impdef values have been supplied for the special
                needs annexes.  Check that the routines in TCTouch work
                correctly.
   - CZ00004 TCTouch ACATS 2.4.
   * 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 ***************************.
```

4.24 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 2.4 01-02-26 14:47:42
---- 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 2.4 01-02-26 14:47:42
---- FAIL_TEST CHECKING 'FAILED' AND 'RESULT' FOR 'FAILED'.
   * FAIL_TEST 'RESULT' SHOULD NOW BE 'FAILED'.
**** FAIL_TEST FAILED ****************************.

,.,. NA_TEST ACATS 2.4 01-02-26 14:47:42
---- NA_TEST CHECKING 'NOT-APPLICABLE'.
   + NA_TEST 'RESULT' SHOULD NOW BE 'NOT-APPLICABLE'.
++++ NA_TEST NOT-APPLICABLE ++++++++++++++++++++.

,.,. FAIL_NA_TEST ACATS 2.4 01-02-26 14:47:42
---- 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 2.4 01-02-26 14:47:42
---- 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 2.4 01-02-26 14:47:42
---- 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 2.4 01-02-26 14:47:42
---- CZ1101A CHECKING 'SPECIAL_ACTION' ALONE.
   ! CZ1101A 'RESULT' BECOMES 'TENTATIVELY PASSED'.
!!!! CZ1101A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!         SEE '!' COMMENTS FOR SPECIAL NOTES!!
```

4.25 Sample Output From CZ1102A

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

```
,.,. CZ1102A ACATS 2.4 01-02-26 14:47:57
---- CZ1102A CHECK THAT THE DYNAMIC VALUE ROUTINES OF THE REPORT PACKAGE
                WORK CORRECTLY.
==== CZ1102A PASSED ============================.
```

4.26 Sample Output From CZ1103A

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

4.26.1 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 2.4 01-02-26 14:48:10
---- 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 2.4 01-02-26 14:48:10
---- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
   ! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!         SEE '!' COMMENTS FOR SPECIAL NOTES!!
```

4.26.2 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 2.4 01-02-26 14:49:00
---- 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 2.4 01-02-26 14:49:00
---- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
   ! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!         SEE '!' COMMENTS FOR SPECIAL NOTES!!
```

Appendix D: Test Applicability Criteria

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

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

Tests which require a floating point DIGITS value that exceeds SYSTEM.MAX_DIGITS need not be submitted to the compiler. (The testing laboratory may require pre-validation evidence that the tests are properly rejected.)

If file I/O is not supported, then the tests listed in Section D.2.14 will not be part of the customized test suite for bare target validations and will not be run on-site.

4.27 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.

4.27.1 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	

4.27.2 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

4.27.3

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

4.27.4 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

4.27.5 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

4.27.6 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

4.27.7 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	

4.27.8 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

4.27.9 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

4.27.10 Instantiation of Sequential_IO with indefinite types

If Sequential_IO does not support indefinite types, then the test contained in the following files are not applicable:

CE2201D.DEP	CE2201E.DEP

4.27.11 Special Handling Tests

Test requiring special handling may also be not applicable. See section 4.6.5, "Tests with Special Processing Requirements", for details.

4.28 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 Appendix F.

4.28.1 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:

c45523a.ada	c45622a.ada

4.28.2 Value of MACHINE_OVERFLOWS is True

If MACHINE_OVERFLOWS is TRUE for floating point types, then the tests contained in the following files should report NOT_APPLICABLE:

c45624a.ada	c45624b.ada

4.28.3 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

4.28.4

Floating Point Overflow

Consider the declaration

```
TYPE F IS DIGITS SYSTEM.MAX_DIGITS;
```

If

```
F'MACHINE_OVERFLOWS is 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

4.28.5 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

4.28.6 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	ce3103a.ada	ce3108a.ada	ce3207a.ada
ce3102a.ada	ce3104a.ada	ce3108b.ada	ce3301a.ada
ce3102b.tst	ce3104b.ada	ce3110a.ada	ce3302a.ada
ce3102f.ada	ce3104c.ada	ce3112c.ada	ce3304a.tst
ce3102g.ada	ce3106a.ada	ce3112d.ada	ce3305a.ada
ce3102h.ada	ce3106b.ada	ce3114a.ada	ce3401a.ada
ce3102j.ada	CE3107A.TST	ce3115a.ada	ce3402a.ada
ce3102k.ada	ce3107b.ada	ce3119a.tst	ce3402c.ada

ce3402d.ada	ce3412a.ada	ce3705e.ada	ce3906a.ada
ce3403a.ada	ce3413a.ada	ce3706d.ada	ce3906b.ada
ce3403b.ada	ce3413b.ada	ce3706f.ada	ce3906c.ada
ce3403c.ada	ce3413c.ada	ce3706g.ada	ce3906e.ada
ce3403e.ada	ce3414a.ada	ce3804a.ada	ce3906f.ada
ce3403f.ada	ce3602a.ada	ce3804b.ada	cxaa001.a
ce3404b.ada	ce3602b.ada	ce3804c.ada	cxaa002.a
ce3404c.ada	ce3602c.ada	ce3804d.ada	cxaa003.a
ce3404d.ada	ce3602d.ada	ce3804e.ada	cxaa004.a
ce3405a.ada	ce3603a.ada	ce3804f.ada	cxaa005.a
ce3405c.ada	ce3604a.ada	ce3804g.ada	cxaa006.a
ce3405d.ada	ce3604b.ada	ce3804h.ada	cxaa007.a
ce3406a.ada	ce3605a.ada	ce3804i.ada	cxaa008.a
ce3406b.ada	ce3605b.ada	ce3804j.ada	cxaa009.a
ce3406c.ada	ce3605c.ada	ce3804m.ada	cxaa010.a
ce3406d.ada	ce3605d.ada	ce3804o.ada	cxaa011.a
ce3407a.ada	ce3605e.ada	ce3804p.ada	cxaa012.a
ce3407b.ada	ce3606a.ada	ce3805a.ada	cxaa013.a
ce3407c.ada	ce3606b.ada	ce3805b.ada	cxaa014.a
ce3408a.ada	ce3704a.ada	ce3806a.ada	cxaa015.a
ce3408b.ada	ce3704b.ada	ce3806b.ada	cxaa016.a
ce3408c.ada	ce3704c.ada	ce3806d.ada	CXAA017.A
ce3409a.ada	ce3704d.ada	ce3806e.ada	CXAA018.A
ce3409c.ada	ce3704e.ada	ce3806g.ada	CXF3A06.A
ce3409d.ada	ce3704f.ada	ce3806h.ada	cxg1003.a
ce3409e.ada	ce3704m.ada	ce3902b.ada	ee3203a.ada
ce3410a.ada	ce3704n.ada	ce3904a.ada	ee3204a.ada
ce3410c.ada	ce3704o.ada	ce3904b.ada	ee3402b.ada
ce3410d.ada	ce3705a.ada	ce3905a.ada	ee3409f.ada
ce3410e.ada	ce3705b.ada	ce3905b.ada	ee3412c.ada
ce3411a.ada	ce3705c.ada	ce3905c.ada	
ce3411c.ada	ce3705d.ada	ce3905l.ada	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a text file with mode IN_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:

ce3102f.ada	ce3302a.ada	ce3404d.ada	ce3409c.ada
ce3102h.ada	ce3402a.ada	ce3406a.ada	ce3409d.ada
ce3104b.ada	ce3403b.ada	ce3406c.ada	ce3409e.ada
ce3106a.ada	ce3403c.ada	ce3406d.ada	ce3410c.ada
ce3108a.ada	ce3403e.ada	ce3407a.ada	ce3410d.ada
ce3108b.ada	ce3403f.ada	ce3407c.ada	ce3410e.ada
ce3112d.ada	ce3404b.ada	ce3408a.ada	ce3411a.ada
ce3301a.ada	ce3404c.ada	ce3408c.ada	ce3411c.ada

ce3412a.ada	ce3704d.ada	ce3804b.ada	ce3806b.ada
ce3413a.ada	ce3704e.ada	ce3804d.ada	ce3806d.ada
ce3413b.ada	ce3704f.ada	ce3804e.ada	ce3806g.ada
ce3413c.ada	ce3704m.ada	ce3804f.ada	ce3902b.ada
ce3602a.ada	ce3704n.ada	ce3804g.ada	ce3904b.ada
ce3602b.ada	ce3704o.ada	ce3804h.ada	ce3905a.ada
ce3602d.ada	ce3705b.ada	ce3804i.ada	ce3905c.ada
ce3603a.ada	ce3705c.ada	ce3804j.ada	ce3905l.ada
ce3604a.ada	ce3705d.ada	ce3804m.ada	ce3906b.ada
ce3604b.ada	ce3705e.ada	ce3804p.ada	ce3906c.ada
ce3605c.ada	ce3706d.ada	ce3805a.ada	ee3203a.ada
ce3704a.ada	ce3706g.ada	ce3805b.ada	ee3204a.ada
ce3704c.ada	ce3804a.ada	ce3806a.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 (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	ce3305a.ada	ce3409d.ada	ce3704b.ada
ce3102a.ada	ce3401a.ada	ce3409e.ada	ce3704c.ada
ce3102b.tst	ce3402a.ada	ce3410a.ada	ce3704d.ada
ce3102f.ada	ce3402c.ada	ce3410c.ada	ce3704e.ada
ce3102g.ada	ce3402d.ada	ce3410d.ada	ce3704f.ada
ce3102h.ada	ce3403a.ada	ce3410e.ada	ce3704m.ada
ce3102j.ada	ce3403b.ada	ce3411a.ada	ce3704n.ada
ce3102k.ada	ce3403c.ada	ce3411c.ada	ce3704o.ada
ce3103a.ada	ce3403e.ada	ce3412a.ada	ce3705a.ada
ce3104a.ada	ce3403f.ada	ce3413a.ada	ce3705b.ada
ce3104b.ada	ce3404b.ada	ce3413b.ada	ce3705c.ada
ce3104c.ada	ce3404c.ada	ce3413c.ada	ce3705d.ada
ce3106a.ada	ce3404d.ada	ce3414a.ada	ce3705e.ada
ce3106b.ada	ce3405a.ada	ce3602a.ada	ce3706d.ada
CE3107A.TST	ce3405c.ada	ce3602b.ada	ce3706f.ada
ce3107b.ada	ce3405d.ada	ce3602c.ada	ce3706g.ada
ce3108a.ada	ce3406a.ada	ce3602d.ada	ce3804a.ada
ce3108b.ada	ce3406b.ada	ce3603a.ada	ce3804b.ada
ce3110a.ada	ce3406c.ada	ce3604a.ada	ce3804c.ada
ce3112c.ada	ce3406d.ada	ce3604b.ada	ce3804d.ada
ce3112d.ada	ce3407a.ada	ce3605a.ada	ce3804e.ada
ce3114a.ada	ce3407b.ada	ce3605b.ada	ce3804f.ada
ce3115a.ada	ce3407c.ada	ce3605c.ada	ce3804g.ada
ce3119a.tst	ce3408a.ada	ce3605d.ada	ce3804h.ada
ce3207a.ada	ce3408b.ada	ce3605e.ada	ce3804i.ada
ce3301a.ada	ce3408c.ada	ce3606a.ada	ce3804j.ada
ce3302a.ada	ce3409a.ada	ce3606b.ada	ce3804m.ada
ce3304a.tst	ce3409c.ada	ce3704a.ada	ce3804o.ada

ce3804p.ada	ce3904a.ada	ce3906f.ada	CXAA017.A
ce3805a.ada	ce3904b.ada	cxaa003.a	CXAA018.A
ce3805b.ada	ce3905a.ada	cxaa004.a	CXF3A06.A
ce3806a.ada	ce3905b.ada	cxaa005.a	cxg1003.a
ce3806b.ada	ce3905c.ada	cxaa009.a	ee3203a.ada
ce3806d.ada	ce3905l.ada	cxaa010.a	ee3204a.ada
ce3806e.ada	ce3906a.ada	cxaa011.a	ee3402b.ada
ce3806g.ada	ce3906b.ada	cxaa012.a	ee3409f.ada
ce3806h.ada	ce3906c.ada	cxaa014.a	ee3412c.ada
ce3902b.ada	ce3906e.ada	cxaa016.a	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a text file with mode APPEND_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:

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

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

4.28.7 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

4.28.8 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	ce2201c.ada	ce2203a.tst
ce2102c.tst	ce2108e.ada	CE2201D.DEP	ce2204a.ada
ce2102g.ada	ce2108f.ada	CE2201E.DEP	ce2204b.ada
ce2102n.ada	ce2109a.ada	ce2201f.ada	ce2204c.ada
ce2102o.ada	ce2110a.ada	ce2201g.ada	ce2204d.ada
ce2102p.ada	ce2111a.ada	ce2201h.ada	ce2205a.ada
ce2102q.ada	ce2111c.ada	ce2201i.ada	ce2206a.ada
ce2102x.ada	ce2111f.ada	ce2201j.ada	ce2208b.ada
CE2103A.TST	ce2111i.ada	ce2201k.ada	cxa8001.a
ce2103c.ada	ce2120b.tst	ce2201l.ada	cxa8002.a
ce2104a.ada	ce2201a.ada	ce2201m.ada	
ce2104b.ada	ce2201b.ada	ce2201n.ada	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a sequential file with mode IN_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:

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 (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	ce2104a.ada	ce2111i.ada	ce2201i.ada
ce2102c.tst	ce2104b.ada	ce2120b.tst	ce2201j.ada
ce2102g.ada	ce2106a.ada	ce2201a.ada	ce2201k.ada
ce2102n.ada	ce2108e.ada	ce2201b.ada	ce2201l.ada
ce2102o.ada	ce2108f.ada	ce2201c.ada	ce2201m.ada
ce2102p.ada	ce2109a.ada	CE2201D.DEP	ce2201n.ada
ce2102q.ada	ce2110a.ada	CE2201E.DEP	ce2203a.tst
ce2102x.ada	ce2111a.ada	ce2201f.ada	ce2204a.ada
CE2103A.TST	ce2111c.ada	ce2201g.ada	ce2204b.ada
ce2103c.ada	ce2111f.ada	ce2201h.ada	ce2204c.ada

ce2204d.ada	ce2206a.ada	cxa8001.a
ce2205a.ada	ce2208b.ada	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a sequential file with mode APPEND_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:

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

4.28.9 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

4.28.10 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	ce2104c.ada	ce2401c.ada	ce2407a.ada
ce2102h.tst	ce2104d.ada	ce2401e.ada	ce2407b.ada
ce2102k.ada	ce2106b.ada	ce2401f.ada	ce2408a.ada
ce2102r.ada	ce2108g.ada	ce2401h.ada	ce2408b.ada
ce2102s.ada	ce2108h.ada	ce2401i.ada	ce2409a.ada
ce2102t.ada	ce2109b.ada	ce2401j.ada	ce2409b.ada
ce2102u.ada	ce2110c.ada	ce2401k.ada	ce2410a.ada
ce2102v.ada	ce2111b.ada	ce2401l.ada	ce2410b.ada
ce2102w.ada	ce2111e.ada	ce2403a.tst	ce2411a.ada
ce2102y.ada	ce2111g.ada	ce2404a.ada	cxa8003.a
CE2103B.TST	ce2120a.tst	ce2404b.ada	cxa9001.a
ce2103d.ada	ce2401a.ada	ce2405b.ada	cxa9002.a
	ce2401b.ada	ce2406a.ada	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a direct access file with mode IN_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:

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 (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	ce2403a.tst	ce2410a.ada
ce2102k.ada	ce2106b.ada	ce2404a.ada	ce2410b.ada
ce2102r.ada	ce2108g.ada	ce2404b.ada	ce2411a.ada
ce2102w.ada	ce2108h.ada	ce2405b.ada	cxa8003.a
ce2102y.ada	ce2110c.ada	ce2407a.ada	cxa9001.a
CE2103B.TST	ce2111b.ada	ce2407b.ada	cxa9002.a
ce2103d.ada	ce2111e.ada	ce2408a.ada	
ce2104c.ada	ce2120a.tst	ce2409b.ada	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a direct access file with mode INOUT_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:

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

4.28.11 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

4.28.12 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:

cxac001.a	CXAC004.A	cxacb01.a
cxac002.a	cxaca01.a	cxacb02.a
cxac003.a	cxaca02.a	cxacc01.a

4.28.13 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 test contained in the following file should report NOT_APPLICABLE:

cxab001.a

4.28.14 File I/O Tests

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

ce2102a.ada	ce2104b.ada	ce2201a.ada	ce2401b.ada
ce2102b.ada	ce2104c.ada	ce2201b.ada	ce2401c.ada
ce2102c.tst	ce2104d.ada	ce2201c.ada	ce2401e.ada
ce2102g.ada	ce2106a.ada	CE2201D.DEP	ce2401f.ada
ce2102h.tst	ce2106b.ada	CE2201E.DEP	ce2401h.ada
ce2102k.ada	ce2108e.ada	ce2201f.ada	ce2401i.ada
ce2102n.ada	ce2108f.ada	ce2201g.ada	ce2401j.ada
ce2102o.ada	ce2108g.ada	ce2201h.ada	ce2401k.ada
ce2102p.ada	ce2108h.ada	ce2201i.ada	ce2401l.ada
ce2102q.ada	ce2109a.ada	ce2201j.ada	ce2403a.tst
ce2102r.ada	ce2109b.ada	ce2201k.ada	ce2404a.ada
ce2102s.ada	ce2109c.ada	ce2201l.ada	ce2404b.ada
ce2102t.ada	ce2110a.ada	ce2201m.ada	ce2405b.ada
ce2102u.ada	ce2110c.ada	ce2201n.ada	ce2406a.ada
ce2102v.ada	ce2111a.ada	ce2203a.tst	ce2407a.ada
ce2102w.ada	ce2111b.ada	ce2204a.ada	ce2407b.ada
ce2102x.ada	ce2111c.ada	ce2204b.ada	ce2408a.ada
ce2102y.ada	ce2111e.ada	ce2204c.ada	ce2408b.ada
CE2103A.TST	ce2111f.ada	ce2204d.ada	ce2409a.ada
CE2103B.TST	ce2111g.ada	ce2205a.ada	ce2409b.ada
ce2103c.ada	ce2111i.ada	ce2206a.ada	ce2410a.ada
ce2103d.ada	ce2120a.tst	ce2208b.ada	ce2410b.ada
ce2104a.ada	ce2120b.tst	ce2401a.ada	ce2411a.ada

ce3102a.ada	ce3406a.ada	ce3704e.ada	ce3906c.ada
ce3102b.tst	ce3406b.ada	ce3704f.ada	ce3906e.ada
ce3102f.ada	ce3406c.ada	ce3704m.ada	ce3906f.ada
ce3102g.ada	ce3406d.ada	ce3704n.ada	cxa8001.a
ce3102h.ada	ce3407a.ada	ce3704o.ada	cxa8002.a
ce3102j.ada	ce3407b.ada	ce3705a.ada	cxa8003.a
ce3102k.ada	ce3407c.ada	ce3705b.ada	cxa9001.a
ce3103a.ada	ce3408a.ada	ce3705c.ada	cxa9002.a
ce3104a.ada	ce3408b.ada	ce3705d.ada	cxaa001.a
ce3104b.ada	ce3408c.ada	ce3705e.ada	cxaa002.a
ce3104c.ada	ce3409a.ada	ce3706d.ada	cxaa003.a
ce3106a.ada	ce3409c.ada	ce3706f.ada	cxaa004.a
ce3106b.ada	ce3409d.ada	ce3706g.ada	cxaa005.a
CE3107A.TST	ce3409e.ada	ce3804a.ada	cxaa006.a
ce3107b.ada	ce3410a.ada	ce3804b.ada	cxaa007.a
ce3108a.ada	ce3410c.ada	ce3804c.ada	cxaa008.a
ce3108b.ada	ce3410d.ada	ce3804d.ada	cxaa009.a
ce3110a.ada	ce3410e.ada	ce3804e.ada	cxaa010.a
ce3112c.ada	ce3411a.ada	ce3804f.ada	cxaa011.a
ce3112d.ada	ce3411c.ada	ce3804g.ada	cxaa012.a
ce3114a.ada	ce3412a.ada	ce3804h.ada	cxaa013.a
ce3115a.ada	ce3413a.ada	ce3804i.ada	cxaa014.a
ce3119a.tst	ce3413b.ada	ce3804j.ada	cxaa015.a
ce3207a.ada	ce3413c.ada	ce3804m.ada	cxaa016.a
ce3301a.ada	ce3414a.ada	ce3804o.ada	CXAA017.A
ce3302a.ada	ce3602a.ada	ce3804p.ada	CXAA018.A
ce3304a.tst	ce3602b.ada	ce3805a.ada	cxab001.a
ce3305a.ada	ce3602c.ada	ce3805b.ada	cxac001.a
ce3401a.ada	ce3602d.ada	ce3806a.ada	cxac002.a
ce3402a.ada	ce3603a.ada	ce3806b.ada	cxac003.a
ce3402c.ada	ce3604a.ada	ce3806d.ada	CXAC004.A
ce3402d.ada	ce3604b.ada	ce3806e.ada	cxaca01.a
ce3403a.ada	ce3605a.ada	ce3806g.ada	cxaca02.a
ce3403b.ada	ce3605b.ada	ce3806h.ada	cxacb01.a
ce3403c.ada	ce3605c.ada	ce3902b.ada	cxacb02.a
ce3403e.ada	ce3605d.ada	ce3904a.ada	cxacc01.a
ce3403f.ada	ce3605e.ada	ce3904b.ada	CXF3A06.A
ce3404b.ada	ce3606a.ada	ce3905a.ada	cxg1003.a
ce3404c.ada	ce3606b.ada	ce3905b.ada	ee3203a.ada
ce3404d.ada	ce3704a.ada	ce3905c.ada	ee3204a.ada
ce3405a.ada	ce3704b.ada	ce3905l.ada	ee3402b.ada
ce3405c.ada	ce3704c.ada	ce3906a.ada	ee3409f.ada
ce3405d.ada	ce3704d.ada	ce3906b.ada	ee3412c.ada

4.28.15 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

4.28.16 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

4.28.17 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	CXC3005

4.28.18 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	CXD2004	CXD6001
CXD2002	CXD2007	CXD6002
CXD2003	CXD2008	CXD6003

4.28.19 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