APPENDIX A
ALPHA/BETA ACCEPTANCE REPORT PRO FORMA

Name of Site

Name, Version and Release of Test System Evaluated

Alpha or Beta Test (or re-test)

Brief Description of Test Environment

Product(s) Against Which Test System Evaluated (Not Mandatory if confidential)

A.1 INTRODUCTION

A.1.1 OBJECTIVE OF DOCUMENT

The objective of this Alpha/Beta acceptance report is to summarise the test system acceptance activity as a whole. In addition, it records the formal recommendation as to whether the test system should be accepted.

A.1.2 SCOPE OF DOCUMENT

This acceptance report includes both objective and subjective comments on a test system and records the formal recommendation on whether the test system should be accepted. The objective and subjective sections of this report are clearly differentiated. A strategy is provided for evaluating the product in such a way that quality criteria can be recorded to give comprehensive and comparable metrics. This report also describes the results of each product/environment combination test and provides a summary of the test coverage and any incident reports raised that are still outstanding.

A.2 MANAGEMENT SUMMARY

A.2.1 OVERALL RECOMMENDATIONS.

Record your overall recommendation as to whether the X/Open Test Development Group should accept the test product. If the recommendation is negative this must be justified within the objective comments section of this document.

WE RECOMMEND THAT X/OPEN ACCEPT THIS TEST SYSTEM Delete as appropriate YES/NO

A.3 OBJECTIVE COMMENTS

Please insert any additional or overall objective comments on the test system's suitability for acceptance in this section. Completion of the section is mandatory if the overall recommendation is rejection. Comments should be made with reference to the acceptance criteria described later in this Appendix but need not be restricted to this. Any other comments, however, must be objective in nature: that is, related to whether the test system meets its functional specification or whether the specification itself is correct.

The evaluator must gather the following information and report it here. Any other data that is pertinent to be highlighted should also be recorded in this section.

- Summary of purpose and scope of review
- Summary of overall impressions as measured by these evaluation criteria
- Summary of changes since last release if relevant
- Summary of problem reports against test suite

A.4 SUBJECTIVE COMMENTS

Please insert any subjective comments on the test systems suitability for acceptance in this section. It is permissible in this section to make reference to issues of style or design approach, although this section will not in itself be used to justify rejection of the test system. This section will be taken into consideration by X/Open for the placement of future work including the product support arrangements.

A.5 STRATEGY AND PROCESS

A.5.1 OVERVIEW

The primary objective of this Appendix is to describe the areas of consideration for evaluating a test suite software delivery in such a way as to provide information that has a direct bearing on the overall quality. In addition the method attempts to combine the evaluated pieces into a visual representation that is easily summarised into one graph. The ultimate goal would be to have a suite that achieves Six Sigma.

A suite of Six Sigma quality would have fewer than 3 defects per 1,000,000 lines of code. This goal has not yet been achieved by any test suite, with the average suite running at around 1000 per 1,000,000 lines of code. In order to represent the data it is presented with two ratings. The first is a letter grade from A to F with A being less than 10 defects, B less than 100, C less than 1000, D less than 10,000 and F more than 10,000. The second is a number rating from 1 to 10 comparing this subject suite to suites of similar types with 1 being the worst possible and a 10 the best.

These ratings are then given to each of the following areas described in this chapter. The actual graph is depicted in a later chapter.

A.5.2 TEST SUITE EVALUATION INSTRUCTIONS

Evaluation of the completed test suite is only the final stage of acquiring a test suite, which may be newly created or adapted from existing technology. Many other criteria, concerning for instance specification, design and commercial terms, are also relevant. The complete set of criteria for choosing a test suite are defined in Appendix E, and these alpha/beta test procedures will be found in the appropriate context in that list of criteria.

The following is a set of guidelines to assist a evaluator when reviewing and evaluating a test suite. To achieve a consistent and quality product, test suites must be subjected to quality controls. This list is not intended to be mandatory or exclusively complete in every case - each test suite acceptance activity must be treated on its own merits.

Open System test suites present many unusual quality problems. By placing the test suites in the Open Systems arena they are assured a wide audience. To be well received the test suite must be accurate, easily understood, run on a variety of implementations and project an image of quality. A evaluator must evaluate all aspects of a test suite. The most important aspect may be the code.

All design and coding of X/Open produced test suites shall conform to the X/Open Programmers Style Guide. The X/Open Programmers Style guide plus the X/Open Test Suite bid requirements are the basis for this evaluation procedure.

A.5.2.1 ANALYSIS OF TEST RESULTS

Be sure and keep the following in mind when analysing the results of a test suite run.

1. Environment tested

- state machine, release, date etc.

2. Test Suite Problems
   - Documentation problems
   - Installation problems
   - Configuration and build problems
   - Execution problems
3. Implementation Problems
   - Installation problems
   - Configuration and build problems
   - Execution problems

A.5.2.2 DOCUMENTATION QUALITY

The quality of the test suite documentation is very important to the usability and perceived quality of the test suite. A person gets the initial impressions about a test suite from the documentation. Things the evaluator needs to look for in the documentation are:

1. Installation instructions
   a. Readability

      EVALUATORS COMMENTS:

   b. Accuracy

      EVALUATORS COMMENTS:

2. Configuration and build instructions
   a. Readability

      EVALUATORS COMMENTS:

   b. Accuracy

      EVALUATORS COMMENTS:

3. Execution instructions
   a. Readability

      EVALUATORS COMMENTS:

b. Accuracy

EVALUATORS COMMENTS:

A.5.2.3 SOURCE CODE QUALITY

There are many factors that determine source code quality of test suites. They must conform themselves to the standard(s) that is being confirmed. They must not make assumptions about the implementation.

A test suite consists of at least three parts.

- The Test Suite Driver, usually TET
- The Assertion Tests
- Utilities

A evaluator needs to review and report on all of these parts. The following outline will help to remind one of what to check for.

A. Style

Style refers to the overall appearance of the code. Since Open System test suites are source code products the appearance of the code is a part of the quality.

1. Each module has an appropriate header / prolog with copyright notice.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Each module has an explanation of the test strategy, coding implementation and of build requirements and resources.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

3. The code is well commented and the comments are accurate and relevant.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

4. The code is as simple as possible and is maintainable.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

B. Architecture

The type of driver, or test suite programming interface, is important. As standards testing evolves the various standards groups are attempting to use the same driver to reduce the complexity of testing. This gives the users of the test suite familiarity and the developer of a test suite a common programming interface.

All tests should relate to a test assertion as described in POSIX 1003.3 or some other formal methodology for defining implementation functionality.

1. Architecture should be TET based.

   a. Does it allow test rig capability?

      YES / NO (Pick one)

      EVALUATORS COMMENTS:

   b. Does it follow POSIX 1003.3 reporting methodology?

      YES / NO (Pick one)

      EVALUATORS COMMENTS:

   c. Can the suite driver be extended in functionality?

      YES / NO (Pick one)

      EVALUATORS COMMENTS:

   d. Test programs written in shell are able to run as stand alone scripts.

      YES / NO (Pick one)

      EVALUATORS COMMENTS:

   e. Test programs which depend on optional functionality that is not supported by the system under test give a report code of UNSUPPORTED.

      YES / NO (Pick one)

      EVALUATORS COMMENTS:

2. The test coverage needs to be complete and provide thorough sampling.

   a. Number of assertions that are tested.

      EVALUATORS COMMENTS:

   b. The percentage of coverage considering the complexity of the test assertions.

      EVALUATORS COMMENTS:

c. How big is the test suite, (ie. Number of Lines Of Code)

EVALUATORS COMMENTS:

C. Portability

The goal of standards is to improve the portability of applications across systems. It follows that the suite itself should be portable as well.

1. Any reliance on non "Standard" dependencies, such as a utility that is not defined by the standard, is minimised and clearly documented.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Null pointer references are not used.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

3. Data type size assumptions should not be present; 8 bit bytes, 1 byte characters, or pointers and integers are the same size.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

4. Tests are not hardware dependent. (16, 32, 64 bit systems)

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

5. Can the test actually fail?

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

6. Tests are written in a defensive manner so that a coding error will not present itself as a PASS condition.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

7. PASS conditions are proved and not assumed.

   YES / NO (Pick one)

EVALUATORS COMMENTS:

8. Lint will run without any errors or warnings on all C language code.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

9. Test programs have a header / prolog which identifies what function is being tested, how to compile and execute, and what the expected results are.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

10. Any security paradym used is portable across different systems.

    YES / NO (Pick one)

    EVALUATORS COMMENTS:

11. Any specific hardware resources necessary to run the test system are not proprietary or otherwise constrained.

    YES / NO (Pick one)

    EVALUATORS COMMENTS:

12. The implementation of directory structures are not exploited in such a way as to hinder portability.

    YES / NO (Pick one)

    EVALUATORS COMMENTS:

D. Maintainability

Test suites are required to be maintained as the reported errors are discovered and as the standard evolves. In addition the test should allow individual tests to run for problem isolation purposes.

1. Are the test sections readable and understandable with not an over use of macro's?

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Is the line of code count reasonable for logical sections; eg are the modules or test sections reasonable, or are there too many levels of indirection and nesting of functions.

YES / NO (Pick one)

EVALUATORS COMMENTS:

E. Resilience

A test suite should be designed with resiliency.

1. Infinite loop constructs and goto statements are avoided.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

   OK

2. Do the C test programs use ANSI C prototypes?

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

3. Errors are handled correctly, avoiding panics or aborted runs.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

4. Graceful shutdown of the test system can be achieved without corruption of the underlying environment.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

5. All return codes are checked and it is not assumed that the operation happened.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

F. Test suite Aids

What is the quality of the various aids provided with the test suite for the test user to manipulate the test suite?

1. Are test suite aids well documented?

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Are test suite aids complete and functional?

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

A.5.2.4 INSTALLATION

Install the test suite following the procedures outlined in the documentation. This may be the most important part of the test suite since it provides the first impression. If the installation goes smoothly the user starts off with a feeling of quality rather than frustration.

1. Review the instructions for readability and completeness.

   EVALUATORS COMMENTS:

2. Verify the accuracy.

   EVALUATORS COMMENTS:

A.5.2.5 CONFIGURATION AND BUILD/COMPILE

Configure and build the Test suite as described in the documentation.

1. The configure and build procedures are accurate and complete.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Setup files and executables are placed in separate directories.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

3. All warnings and errors can be resolved.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

A.5.2.6 EXECUTION

Execute the test suite per the instruction in the documentation. The following highlights what needs to be evaluated and reported.

1. The execution instructions are accurate and complete.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. All required tests run.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

3. If there are multiple methods of execution (menu, standalone, etc.), all methods work correctly.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

4. All tests that do not report Pass can be resolved.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

5. The log / journal files are complete, accurate and readable.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

6. There are messages associated with each error or fail message.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

7. The tests produce only the result codes as defined in the 1003.3 standard.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

8. The test suite supports remote execution from src tree.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

9. The test suite uses only a common driver and not more than one referenced library of routines.

YES / NO (Pick one)

EVALUATORS COMMENTS:

10. There is a minimum of dependency between tests, to ease selective running of individual tests.

    YES / NO (Pick one)

    EVALUATORS COMMENTS:

11. The system is tolerant of any unpredictable results from a non-conforming system under test.

    YES / NO (Pick one)

    EVALUATORS COMMENTS:

A.5.2.7 CLEANUP

A good test suite is designed to be re-executable, without filling up directories or leaving scratch files undeleted etc.

1. Setup, executable and scratch files are removed.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

2. Past runs of the suite do not effect subsequent runs.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

A.5.2.8 REPORTS

The reporting that a test suite does is how one evaluates and substantiates its claims.

1. The reports are accurate and reflect the status as logged in the journal files.

   YES / NO (Pick one)

   EVALUATORS COMMENTS:

A.6 TEST CASE TABLES BY AREA

Fill in the test case table for each test case examined and thus identify the validation method used and the result of the test. The full pathname of the test should be specified, however common routines may be referred to by groups In the method column place either a tick or cross in each of the method(s) column(s) used to validate a particular test case depending on whether there is a suspected error. Where a particular method(s) has not been employed to validate the test case then leave the the appropriate column(s) blank. For each suspected errors encountered raise an incident report or reports and insert the forms in this document after this page. Be sure to specify which test method revealed the error and in which test case it is suspected. For guidance a minimum twenty percent of each test group's test cases should be validated by one or more of the defined methods. All three test methods should be used but not necessarily on each test case validated. At least 40 test cases must be verified by review, & strategy check. Use one form for each test group. Copy the test case table as many times as is necessary to document each test case validated.

Note: To save writing it is not necessary to fill in each box individually. For example where consecutive numbers of test cases have been validated identify the first test case, the last test case and link the two vertically with an arrow. Similar technics may be used in the method/results column. The attachment of test logs, annotated or otherwise is encouraged. However it is still a requirement that the test case table be completed in order that test coverage is readily apparent to a reader not familiar with log formats.

TEST CASE AREA VALIDATION TABLE

Test Case Area

Test case Number	Test Method Employed / Result			
	Code Review	Test Journal Analysis	Strategy Check	Static Dynamic Trace

A.7 SUMMARY ALPHA TEST RESULTS

A.7.1 OUTSTANDING FATAL ERRORS.

Insert a copy of all outstanding incident reports classified as fatal errors in this report after this page.

OUTSTANDING FATAL ERRORS

A.7.2 OUTSTANDING CRITICAL ERRORS

Insert a copy of all outstanding incident reports classified as critical errors in this report after this page.

OUTSTANDING CRITICAL ERRORS

A.7.3 OUTSTANDING STANDARD ERRORS.

Insert a copy of all outstanding incident reports classified as standard errors in this report after this page.

OUTSTANDING STANDARD ERRORS

A.7.4 OUTSTANDING ENHANCEMENTS.

Insert a copy of all outstanding incident reports classified as Enhancements in this report after this page.

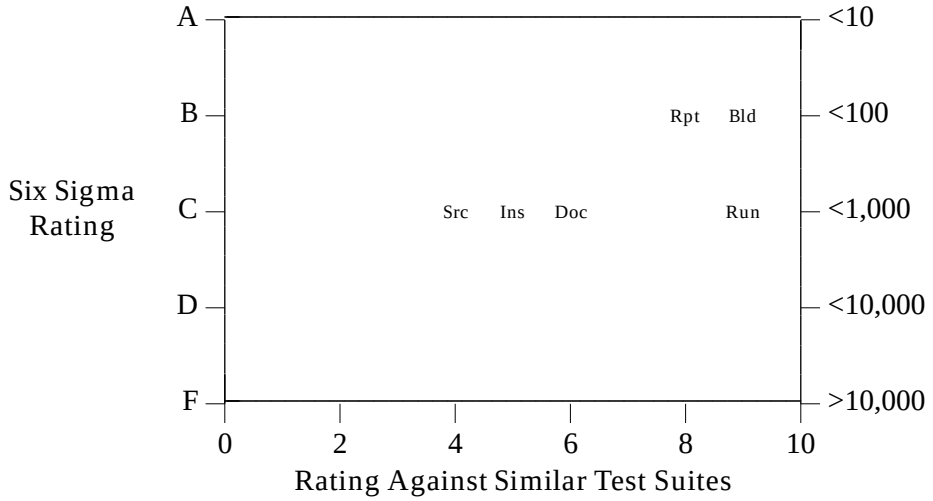
OUTSTANDING ENHANCEMENTS

A.8 METHOD OF DOCUMENTATION

This evaluation is represented based on the following methodology. A letter and a numerical rating is given for each of the applicable test suite areas. A letter grade from A to F represents a defect rating with A being less than 10 defects, B less than 100, C less than 1000, D less than 10,000 and F more than 10,000. A number from 1 to 10 is used to rate the tests against tests suites of similar types, where 1 is very poor and 10 is perfect.

A.8.1 GRAPHICAL REPRESENTATION

For example a test suite might have a rating such as:



In this example the Documentation was rated as having less than 1000 defects per 1,000,000 lines of code, and compared about average to other test suites; C6. The code, however, faired worse with a rating of C4.

CONTENTS