Technology Compatibility Kit User's Guide for SOAP with Attachments API for Java 1.3

For Technology Licensees

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.1-800-555-9SUN1-650-960-1300

Part No:
May 2006

060502@14774

Contents

Preface

This guide describes how to install, configure, and run the Technology Compatibility Kit (TCK) that is used to test the Common SOAP with Attachments API for Java 1.3 (SAAJ 1.3) technology.

The SAAJ TCK is designed as a portable, configurable automated test suite for verifying the compatibility of a licensee's implementation of the SAAJ 1.3 Specification (hereafter referred to as the licensee implementation). The SAAJ TCK uses the JavaTest harness version 3.2.1 to run the test suite

---

Note – Note All references to specific Web URLs are given for the sake of your convenience in locating the resources quickly. These references are always subject to changes that are in many cases beyond the control of the authors of this guide.

Refer to the Java Partner Engineering (`https://javapartner.sun.com`) Web site for answers to frequently asked questions and send questions you may have to your Java Partner Engineering contact.

Who Should Use This Book

This guide is for licensees of Sun Microsystems's SAAJ 1.3 technology to assist them in running the test suite that verifies compatibility of their implementation of the SAAJ 1.3 Specification.

Before You Read This Book

Before reading this guide, you should familiarize yourself with the Java programming language and the SAAJ 1.3 Specification. A good resource for the Java Programming language is the Sun Microsystems, Inc. (`http://java.sun.com`) Web site.

The SOAP with Attachments API for Java 1.3 is based on the SAAJ Specification 1.3. Links to the specification and other product information can be found on the Web at `http://jcp.org/aboutJava/communityprocess/maintenance/jsr067/index2.html`. Sun Microsystems recommends that before you run the tests in the SAAJ TCK you have read and are familiar with the SAAJ 1.3 Specification and the JavaTest User's Guide, which describes the main JavaTest harness. The JavaTest User's Guide is located at `<install_dir>/docs/javatest/javatest.pdf` in the SAAJ TCK distribution.

How This Book is Organized

If you are installing and using the SAAJ TCK for the first time, it is recommended that you read Chapter 1 and Chapter 2 completely for the necessary background information, and then perform the steps outlined in the remaining chapters, while referring to the appendix as necessary.

- Chapter 1 gives an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and describes the SAAJ TCK. It also includes a listing of the basic steps needed to get up and running with the SAAJ TCK.
- Chapter 2 describes the conformance testing procedure and testing requirements.
- Chapter 3 explains how to install the SAAJ TCK.
- Chapter 4 describes how to set up the SAAJ TCK and how to start and configure the JavaTest harness software to be used with the SAAJ TCK.
- Chapter 5 describes how to start the JavaTest harness and use the SAAJ TCK.
- Chapter 6 describes several approaches for dealing with tests that fail to execute properly.
- Appendix A provides answers to frequently asked questions.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Related Documentation

TABLE P–3 Related Documentation Titles

Application	Title
Using JavaTest software	*JavaTest User's Guide* and JavaTest online help (located in the `TS_HOME/docs` directory in the SAAJ TCK distribution)
SAAJ reference	The *SOAP with Attachments API for Java 1.3 Specification*
Using Java technology	*Java Programming Language*, *Java Language Specification Second Edition*, *Java Virtual Machine Specification Second Edition*, *Java 2 Platform*

Accessing Sun Documentation Online

The Java Developer Connection (`http://developer.java.sun.com/developer/infodocs/`)SM Web site enables you to access Java platform technical documentation.

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at `mailto:docs@java.sun.com`.

◆ ◆ ◆

CHAPTER 1

Introduction

This chapter provides an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and describes the SOAP with Attachments API for Java 1.3 TCK (SAAJ TCK 1.3). It also includes a high level listing of what is needed to get up and running with the SAAJ TCK.

1.1 Compatibility Testing

Compatibility testing differs from traditional product testing in a number of ways. The focus of compatibility testing is to test those features and areas of an implementation that are likely to differ across other implementations, such as those features that:

- Rely on hardware or operating system-specific behavior
- Are difficult to port
- Mask or abstract hardware or operating system behavior

Compatibility test development for a given feature relies on a complete specification and reference implementation for that feature. Compatibility testing is not primarily concerned with robustness, performance, or ease of use.

1.1.1 Why Compatibility Testing is Important

Java platform compatibility is important to different groups involved with Java technologies for different reasons:

- Compatibility testing is the means by which Sun Microsystems ensures that the Java platform does not become fragmented as it is ported to different operating systems and hardware environments.
- Compatibility testing benefits developers working in the Java programming language, allowing them to write applications once and then to deploy them across heterogeneous computing environments without porting.

- Compatibility testing allows application users to obtain applications from disparate sources and deploy them with confidence.
- Conformance testing benefits Java platform implementors by ensuring a level playing field for all Java platform ports.

1.1.2 TCK Compatibility Rules

Compatibility criteria for all technology implementations are embodied in the TCK Compatibility Rules that apply to a specified technology. Each TCK tests for adherence to these Rules as described in Chapter 2.

1.1.3 TCK Overview

A TCK is a set of tools and tests used to verify that a licensee's implementation of Sun Microsystems's technology conforms to the applicable specification. All tests in the TCK are based on the written specifications for the Java platform. A TCK tests compatibility of a licensee's implementation of Sun Microsystems's technology to the applicable specification of the technology. Compatibility testing is a means of ensuring correctness, completeness, and consistency across all implementations developed by Sun Microsystems technology licensees.

The set of tests included with each TCK is called the *test suite*. Most tests in a TCK's test suite are self-checking, but some tests may require tester interaction. Most tests return either a Pass or Fail status. For a given platform to be certified, all of the required tests must pass. The definition of required tests may change from platform to platform.

The definition of required tests will change over time. Before your final certification test pass, be sure to download the latest Exclude List for the TCK you are using.

1.1.4 Java Community Process (JCP) Program and Compatibility Testing

The Java Community Process™ (JCP) program is the formalization of the open process that Sun Microsystems, Inc. has been using since 1995 to develop and revise Java technology specifications in cooperation with the international Java community. The JCP program specifies that the following three major components must be included as deliverables in a final Java technology release under the direction of the responsible Expert Group:

- Technology Specification
- Reference Implementation
- Technology Compatibility Kit (TCK)

For further information about the JCP program, go to Java Community Process (`http://jcp.org/en/jsr/detail?id=067`).

1.2 About the SAAJ TCK 1.3

The SAAJ TCK 1.3 is designed as a portable, configurable, automated test suite for verifying the compatibility of a licensee's implementation of Sun Microsystems's SAAJ 1.3 Specification.

1.2.1 SAAJ TCK Specifications and Requirements

This section lists the applicable requirements and specifications.

- **Specification Requirements** — Software requirements for a SAAJ implementation are described in detail in the SAAJ 1.3 Specification. Links to the SAAJ specification and other product information can be found at `http://jcp.org/aboutJava/communityprocess/maintenance/jsr067/index2.html`.
- **SAAJ Version** — The SAAJ TCK 1.3 is based on the SAAJ Specification, Version 1.3.
- **Reference Runtime** — The designated Reference Runtime for conformance testing of implementations based upon the SAAJ Specification 1.3 is the Sun Microsystems SAAJ 1.3 Reference Implementation (RI). See the *SAAJ RI Installation Guide* at `http://java.sun.com/webservices/saaj/index.jsp` for setup instructions.

1.2.2 SAAJ TCK Components

The SAAJ TCK 1.3 includes the following components:

- **JavaTest harness** version 3.2.1 and related documentation. See the `README-javatest.html` file, the *JavaTest Users Guide*, and the `ReleaseNotes-javatest.html` file for additional information.
- **SAAJ TCK signature tests**; check that all public APIs are supported and/or defined as specified in the SAAJ Version 1.3 implementation under test.
- **API tests** for all of the SAAJ API in all related packages:

  `javax.xml.soap`
- **End-to-end tests** that demonstrate sending/receiving SOAP messages to/from a URL endpoint as well tests for WSI compliance..

The SAAJ TCK tests have been tested with the following:

- SAAJ 1.3 Reference Implementation
- Java Standard Edition 5

The SAAJ TCK tests run on the following platforms:

- Solaris 9 Operating System on Sparc and x86
- Solaris 10 Operating System on Sparc and Opteron
- Windows 2003 Server Professional Edition
- Windows XP Professional Edition

- Windows 2000 Professional Edition
- RedHat Linux 9.0
- RedHat Linux AS 3.0 (including Update 4)

1.2.3 JavaTest Harness

The JavaTest™ harness version 3.2.1 is a set of tools designed to run and manage test suites on different Java platforms. The JavaTest harness can be described as both a Java application and a set of compatibility testing tools. It can run tests on different kinds of Java platforms and it allows the results to be browsed online within the JavaTest GUI, or offline in the HTML reports that the JavaTest harness generates.

The JavaTest harness includes the applications and tools that are used for test execution and test suite management. It supports the following features:

- Sequencing of tests, allowing them to be loaded and executed automatically
- Graphic user interface (GUI) for ease of use
- Automated reporting capability to minimize manual errors
- Failure analysis
- Test result auditing and auditable test specification framework
- Distributed testing environment support

To run tests using the JavaTest harness, you specify which tests in the test suite to run, how to run them, and where to put the results as described in Chapter 4.

1.2.4 TCK Compatibility Test Suite

The *test suite* is the collection of tests used by the JavaTest harness to test a particular technology implementation. In this case, it is the collection of tests used by the SAAJ TCK 1.3 to test a SAAJ 1.3 implementation. The tests are designed to verify that a licensee's runtime implementation of the technology complies with the appropriate specification. The individual tests correspond to assertions of the specification.

The tests that make up the TCK compatibility test suite are precompiled and indexed within the TCK test directory structure. When a test run is started, the JavaTest harness scans through the set of tests that are located under the directories that have been selected. While scanning, the JavaTest harness selects the appropriate tests according to any matches with the filters you are using and queues them up for execution.

1.2.5 Exclude Lists

Each version of a TCK includes an Exclude List contained in a `.jtx` file. This is a list of test file URLs that identify tests which do not have to be run for the specific version of the TCK being used. Whenever tests are run, the JavaTest harness automatically excludes any test on the Exclude List from being executed.

A licensee is not required to pass any test—or even run any test—on the Exclude List. The Exclude List file, `<TS_HOME>/bin/ts.jtx`, is included in the SAAJ TCK.

Note – From time to time, updates to the Exclude List are made available on the Java Partner Engineering (`https://javapartner.sun.com`) Web site. You should always make sure you are using an up-to-date copy of the Exclude List before running the SAAJ TCK to verify your implementation.

A test might be in the Exclude List for reasons such as:

- An error in an underlying implementation API has been discovered which does not allow the test to execute properly.
- An error in the specification that was used as the basis of the test has been discovered.
- An error in the test itself has been discovered.
- The test fails due to a bug in the tools (such as the JavaTest harness, for example).

In addition, all tests are run against the Sun Microsystems reference implementations. Any tests that fail when run on a reference Java platform are put on the Exclude List. Any test that is not specification-based, or for which the specification is vague, may be excluded. Any test that is found to be implementation dependent (based on a particular thread scheduling model, based on a particular file system behavior, and so on) may be excluded.

Note – Licensees are not permitted to alter or modify Exclude Lists. Changes to an Exclude List can only be made by using the procedure described in SOAP with Attachments API for Java 1.3 Test Appeals Process.

1.2.6 SAAJ TCK Configuration

You need to set several variables in your test environment, modify properties in the `<TS_HOME>/bin/ts.jte` file, and then use the JavaTest harness to configure and run the SAAJ tests, as described in Chapter 4.

1.3 Getting Started With the SAAJ TCK

This section provides an general overview of what needs to be done to install, set up, test, and use the SAAJ TCK. These steps are explained in more detail in subsequent chapters of this guide.

▼ 1.3.1 To get started with the SAAJ TCK

1 Make sure that the following software has been correctly installed on the system hosting the JavaTest harness:

- Java SE 5
- Apache Tomcat 5.5.*x*
- SAAJ TCK version 1.3
- An implementation of the SAAJ 1.3 specification

See the documentation for each of these software applications for installation instructions.

2 Install the SAAJ TCK 1.3 software.

See Chapter 3 for additional information.

3 Set up the SAAJ TCK software.

See Chapter 4 for details about the following steps.

a. Set up your shell environment.

b. Modify the required properties in the `<TS_HOME>/bin/ts.jte` file.

c. Configure the JavaTest harness.

4 Test the SAAJ 1.3 implementation.

Test the SAAJ implementation installation by running the test suite. See Chapter 5.

◆ ◆ ◆ CHAPTER 2

Procedure for SOAP with Attachments API for Java 1.3 Certification

This chapter describes the compatibility testing process and compatibility requirements for SOAP with Attachments API for Java 1.3. This chapter contains the following sections:

- "2.1 Certification Overview" on page 17
- "2.2 Compatibility Requirements" on page 17
- "2.3 Reference Runtime for SOAP with Attachments API for Java 1.3" on page 24
- "2.4 Specifications for SOAP with Attachments API for Java 1.3" on page 24
- "2.5 Libraries for SOAP with Attachments API for Java 1.3" on page 24

2.1 Certification Overview

The certification process for SOAP with Attachments API for Java 1.3 consists of the following activities:

- Install the appropriate version of the Technology Compatibility Kit (TCK) and execute it in accordance with the instructions in this User's Guide.
- Ensure that you meet the requirements outlined in Compatibility Requirements.
- Certify to Java Licensee Engineering that you have finished testing and that you meet all the compatibility requirements.

2.2 Compatibility Requirements

The compatibility requirements for SOAP with Attachments API for Java 1.3 consist of meeting the requirements set forth by the rules and associated definitions contained in this section.

2.2.1 Definitions

These definitions are for use only with these compatibility requirements and are not intended for any other purpose.

TABLE 2–1 Definitions

Term	Definition
Computational Resource	A piece of hardware or software that may vary in quantity, existence, or version, which may be required to exist in a minimum quantity and/or at a specific or minimum revision level so as to satisfy the requirements of the Test Suite. Examples of computational resources that may vary in quantity are RAM and file descriptors. Examples of computational resources that may vary in existence (that is, may or may not exist) are graphics cards and device drivers. Examples of computational resources that may vary in version are operating systems and device drivers.
Configuration Descriptor	Any file whose format is well defined by a specification and which contains configuration information for a set of Java classes, archive, or other feature defined in the specification.
Conformance Tests	All tests in the Test Suite for an indicated Technology Under Test, as distributed by the Maintenance Lead, excluding those tests on the Exclude List for the Technology Under Test.
Documented	Made technically accessible and made known to users, typically by means such as marketing materials, product documentation, usage messages, or developer support programs.
Exclude List	The most current list of tests, distributed by the Maintenance Lead, that are not required to be passed to certify conformance. The Maintenance Lead may add to the Exclude List for that Test Suite as needed at any time, in which case the updated Exclude List supplants any previous Exclude Lists for that Test Suite.
Libraries	The class libraries, as specified through the Java Community Process (JCP), for the Technology Under Test. The Libraries for SOAP with Attachments API for Java 1.3 are listed at the end of this chapter.

TABLE 2–1 Definitions *(Continued)*

Term	Definition
Location Resource	A location of classes or native libraries that are components of the test tools or tests, such that these classes or libraries may be required to exist in a certain location in order to satisfy the requirements of the test suite. For example, classes may be required to exist in directories named in a `CLASSPATH` variable, or native libraries may be required to exist in directories named in a `PATH` variable.
Maintenance Lead	The Java Community Process member responsible for maintaining the Specification, reference implementation, and TCK for the Technology. Sun is the Maintenance Lead for SOAP with Attachments API for Java 1.3.
Operating Mode	Any Documented option of a Product that can be changed by a user in order to modify the behavior of the Product. For example, an Operating Mode can be binary (enable/disable optimization), an enumeration (select from a list of protocols), or a range (set the maximum number of active threads).
Product	A licensee product in which the Technology Under Test is implemented or incorporated, and that is subject to compatibility testing.
Product Configuration	A specific setting or instantiation of an Operating Mode. For example, a Product supporting an Operating Mode that permits user selection of an external encryption package may have a Product Configuration that links the Product to that encryption package.
Resource	A Computational Resource, a Location Resource, or a Security Resource.
Rules	These definitions and rules in this Compatibility Requirements section of this User’s Guide.

TABLE 2–1 Definitions *(Continued)*

Term	Definition
Security Resource	A security privilege or policy necessary for the proper execution of the Test Suite. For example, the user executing the Test Suite will need the privilege to access the files and network resources necessary for use of the Product.
Specifications	The documents produced through the Java Community Process that define a particular Version of a Technology. The Specifications for the Technology Under Test can be found later in this chapter.
Technology	Specifications and a reference implementation produced through the Java Community Process.
Technology Under Test	Specifications and the reference implementation for SOAP with Attachments API for Java 1.3.
Test Suite	The requirements, tests, and testing tools distributed by the Maintenance Lead as applicable to a given Version of the Technology.
Version	A release of the Technology, as produced through the Java Community Process.

2.2.2 Rules for SOAP with Attachments API for Java 1.3 Products

The following rules apply for each version of an operating system, software component, and hardware platform Documented as supporting the Product:

1. The Product must be able to satisfy all applicable compatibility requirements, including passing all Conformance Tests, in every Product Configuration and in every combination of Product Configurations, except only as specifically exempted by these Rules.

   For example, if a Product provides distinct Operating Modes to optimize performance, then that Product must satisfy all applicable compatibility requirements for a Product in each Product Configuration, and combination of Product Configurations, of those Operating Modes.

   a. If an Operating Mode controls a Resource necessary for the basic execution of the Test Suite, testing may always use a Product Configuration of that Operating Mode providing that Resource, even if other Product Configurations do not provide that Resource. Notwithstanding such exceptions, each Product must have at least one set of Product Configurations of such Operating Modes that is able to pass all the Conformance Tests.

   For example, a Product with an Operating Mode that controls a security policy (i.e., Security Resource) which has one or more Product Configurations that cause Conformance Tests to fail may be tested using a Product Configuration that allows all Conformance Tests to pass.

   b. A Product Configuration of an Operating Mode that causes the Product to report only version, usage, or diagnostic information is exempted from these compatibility rules.

2. Some Conformance Tests may have properties that may be changed. Properties that can be changed are identified in the TCK configuration interview. Apart from changing such properties and other allowed modifications described in this User's Guide (if any), no source or binary code for a Conformance Test may be altered in any way without prior written permission. Any such allowed alterations to the Conformance Tests would be posted to the Java Licensee Engineering web site and apply to all licensees.
3. The testing tools supplied as part of the Test Suite or as updated by the Maintenance Lead must be used to certify compliance.
4. The Exclude List associated with the Test Suite cannot be modified.
5. The Maintenance Lead can define exceptions to these Rules. Such exceptions would be made available to and apply to all licensees.
6. All hardware and software component additions, deletions, and modifications to a Documented supporting hardware/software platform, that are not part of the Product but required for the Product to satisfy the compatibility requirements, must be Documented and available to users of the Product.

   For example, if a patch to a particular version of a supporting operating system is required for the Product to pass the Conformance Tests, that patch must be Documented and available to users of the Product.

7. The Product must contain the full set of public and protected classes and interfaces for all the Libraries. Those classes and interfaces must contain exactly the set of public and protected methods, constructors, and fields defined by the Specifications for those Libraries. No subsetting, supersetting, or modifications of the public and protected API of the Libraries are allowed except only as specifically exempted by these Rules.

   a. The Product may contain a subset of the classes and interfaces for the Libraries.

8. Except for tests specifically required by this TCK to be rebuilt (if any), the binary Conformance Tests supplied as part of the Test Suite or as updated by the Maintenance Lead must be used to certify compliance.
9. The functional programmatic behavior of any binary class or interface must be that defined by the Specifications.

2.2.3 SOAP with Attachments API for Java 1.3 Test Appeals Process

Sun has a well established process for managing challenges to its Java technology Test Suites and plans to continue using a similar process in the future. Because SOAP with Attachments API for Java 1.3 requires one or more subcomponent TCKs, the SOAP with Attachments API for Java 1.3 test appeals process will be consistent with the existing subcomponent TCK test appeals processes. Sun, as SOAP with Attachments API for Java 1.3, will authorize representatives from the Java Licensee Engineering group to be the point of contact for all test challenges. Typically this will be the engineer assigned to a company as part of its SOAP with Attachments API for Java TCK support.

If a test is determined to be invalid in function or if its basis in the specification is suspect, the test may be challenged by any licensee of the SOAP with Attachments API for Java TCK. Each test validity issue must be covered by a separate test challenge. Test validity or invalidity will be determined based on its technical correctness such as:

- Test has bugs (i.e., program logic errors)
- Specification item covered by the test is ambiguous
- Test does not match the specification
- Test assumes unreasonable hardware and/or software requirements
- Test is biased to a particular implementation

Challenges based upon issues unrelated to technical correctness as defined by the specification will normally be rejected.

Test challenges must be made in writing to Java Licensee Engineering and include all relevant information as described in the Test Challenge Form. The process used to determine the validity or invalidity of a test (or related group of tests) is described in "SOAP with Attachments API for Java TCK Test Appeals Steps."

All tests found to be invalid will either be placed on the Exclude List for that version of the SOAP with Attachments API for Java TCK or have an alternate test made available.

- Tests that are placed on the Exclude List will be placed on the Exclude List within one business day after the determination of test validity. The new Exclude List will be made available to all SOAP with Attachments API for Java TCK licensees on the SOAP with Attachments API for Java TCK Web site.
- Sun, as Maintenance Lead has the option of creating alternative tests to address any challenge. Alternative tests (and criteria for their use) will be made available on the SOAP with Attachments API for Java TCK Web site.

Note – Passing an alternative test is deemed equivalent to passing the original test.

▼ 2.2.3.1 SOAP with Attachments API for Java TCK Test Appeals Steps

1. **SOAP with Attachments API for Java TCK licensee writes a test challenge to Java Licensee Engineering contesting the validity of one or a related set of SOAP with Attachments API for Java 1.3 tests.**

   A detailed justification for why each test should be invalidated must be included with the challenge as described by the Test Challenge Form.

2. **Java Licensee Engineering evaluates the challenge.**

   If the appeal is incomplete or unclear, it is returned to the submitting licensee for correction. If all is in order, Java Licensee Engineering will check with the responsible test developers to review the purpose and validity of the test before writing a response as described by the Test Challenge Response Form. Java Licensee Engineering will attempt to complete the response within 5 business days. If the challenge is similar to a previously rejected test challenge (i.e., same test and justification), Java Licensee Engineering will send the previous response to the licensee.

3. **The challenge and any supporting materials from test developers is sent to the specification engineers for evaluation.**

   A decision of test validity or invalidity is normally made within 15 working days of receipt of the challenge. All decisions will be documented with an explanation of why test validity was maintained or rejected.

4. **The licensee is informed of the decision and proceeds accordingly.**

   If the test challenge is approved and one or more tests are invalidated, Sun places the tests on the Exclude List for that version of the SOAP with Attachments API for Java TCK (effectively removing the test(s) from the Test Suite). All tests placed on the Exclude List will have a bug report written to document the decision and made available to all licensees through the bug reporting database. If the test is valid but difficult to pass due to hardware or operating system limitations, Sun may choose to provide an alternate test to use in place of the original test (all alternate tests are made available to the licensee community).

5. **If the test challenge is rejected, the licensee may choose to escalate the decision to the Executive Committee (EC), however, it is expected that the licensee would continue to work with Sun to resolve the issue and only involve the EC as a last resort.**

2.2.3.2 Test Challenge Form

Provide the following information to Java Licensee Engineering:

```
Test Challenger Name and Company:
Specification Name(s) and Version(s):
Test Suite Name and Version:
Exclude List Version:
Test Name:
Complaint (argument for why test is invalid):
```

2.2.3.3 Test Challenge Response Form

Provide the following information in response to a test challenge.

```
Test Defender Name and Company:
Test Defender Role in Defense (e.g., test developer, Maintenance
   Lead, etc.):
Specification Name(s) and Version(s):
```

```
Test Suite Name and Version:
Test Name:
Defense (argument for why test is valid):
[Multiple challenges and corresponding responses may be listed
   here.]
Implications of test invalidity (e.g., other affected tests and test
   framework code, creation or exposure of ambiguities in spec (due
   to unspecified requirements), invalidation of the reference
   implementation, creation of serious holes in test suite):
Alternatives (e.g., are alternate test(s) appropriate?):
```

2.3 Reference Runtime for SOAP with Attachments API for Java 1.3

Designated Reference Implementation for compatibility testing of SOAP with Attachments API for Java 1.3 is as follows:

- Java SE 5 for Solaris OE/SPARC, and Win32
- Sun Reference Implementation of SAAJ Version1.3.
- Sun Solaris/SPARC 2.8 and 2.9, Red Hat Linux 7.2, and Microsoft Windows XP/ 2000

2.4 Specifications for SOAP with Attachments API for Java 1.3

The Specifications for Java SE 5 are found on the Java Partner Engineering (`https://javapartner.sun.com`) web site.

2.5 Libraries for SOAP with Attachments API for Java 1.3

The following is a list of the packages comprising the required class libraries SAAJ 1.3:

`javax.xml.soap`

For the latest list of packages, also see:

`http://java.sun.com/j2se/1.5/docs/api/index.html`

◆ ◆ ◆ CHAPTER 3

Installation

This chapter explains how to install the SOAP with Attachments API for Java TCK 1.3 (SAAJ TCK) software. After installing the software according to the instructions in this chapter, proceed to Chapter 4 for instructions on configuring your test environment.

3.1 Obtaining the SAAJ 1.3 Reference Implementation

You can obtain the Sun Microsystems SAAJ Reference Implementation (RI), Version 1.3 software from the Java Community Process (`http://jcp.org/en/jsr/detail?id=067`) web site. Sun's SAAJ RI software is not necessary for running the SAAJ TCK, but it can be useful as a test base for familiarizing yourself with the TCK before testing your own SAAJ implementation.

3.2 Installing the Software

Before you can run the SAAJ TCK tests, you must install and set up the following software components:

- Java SE 5
- Apache Tomcat 5.5.*x*
- SAAJ TCK version 1.3
- An implementation of the SAAJ 1.3 specification

▼ 3.2.1 To install the SAAJ software

1 **Install the Java SE 5 software, if it is not already installed.**

Download and install the Java SE 5 software from the Java Software (`http://java.sun.com/products`) Web site. Refer to the installation instructions that accompany the software for additional information.

2 **Install the Apache Tomcat 5.5.*x* server software, if it is not already installed.**

Download and install the latest version of Tomcat from Apache.org (`http://www.apache.org`). Refer to the installation instructions that accompany the software for additional information.

3 **Install the SAAJ TCK 1.3 software.**

a. **Copy or download the SAAJ TCK software to your local system.**

You can obtain the SAAJ TCK software from the Java Partner Engineering (`https://javapartner.sun.com`) Web site. The SAAJ TCK software is located in the `OPTPKG-XML/saaj` directory in the Web site's Download Center area.

b. **Change to the directory in which you want to install the SAAJ TCK software:**

```
cd <install_directory>
```

c. **Use the `unzip` command to extract the bundle:**

```
unzip saaj-1_3-tck.zip
```

This creates the `saajtck` directory. The `<install_directory>/saajtck` directory is the test suite home, `TS_HOME`.

4 **Install the SAAJ implementation to be tested.**

Before testing your own SAAJ implementation, it is recommended that you run the SAAJ TCK against the Sun Microsystems SAAJ RI to familiarize yourself with the SAAJ TCK suite and JavaTest software. See Obtaining the SAAJ 1.3 Reference Implementation for more information

◆ ◆ ◆ CHAPTER 4

Setup and Configuration

This chapter describes how to set up the SAAJ TCK and JavaTest harness software. Before proceeding with the instructions in this chapter, be sure to install all required software, as described in Chapter 3.

After completing the instructions in this chapter, proceed to Chapter 5 for instructions on running the SAAJ TCK.

This chapter includes the following topics:

4.1 Configuring Your Environment to Run the SAAJ TCK

This section describes how to configure the SAAJ TCK for your environment. After configuring your environment, continue with the instructions in Using the JavaTest Harness Software.

Note – In these instructions, variables in angle brackets need to be expanded for each platform. For example, `<TS_HOME>` becomes `$TS_HOME` on Solaris/Linux and `%TS_HOME%` on Windows 2000/XP. In addition, the forward slashes (`/`) used in all of the examples need to be replaced with backslashes (`\`) for Windows 2000/XP. Finally, be sure to use the appropriate separator for your operating system when specifying multiple path entries (`;` on Windows, `:` on UNIX/Linux).

▼ 4.1.1 To Configure Your Environment for the SAAJ TCK

1. **Set the following environment variables in your shell environment:**
   1. `JAVA_HOME` **to the directory in which Java SE 5 is installed**
   2. `TS_HOME` **to the directory in which the SAAJ TCK 1.3 software is installed**
   3. `SAAJ_HOME` **to the directory in which the SAAJ RI software is installed**
   4. `CATALINA_HOME` **to the directory in which the Apache Tomcat 5.5.*x* Web Container is installed**
   5. `ANT_HOME` **should *not* be set in your environment. If it is set, either unset it or make sure it is set to** `<TS_HOME>/tools/ant`**. The** `tsant` **script automatically sets** `ANT_HOME` **if it is unset in the environment.**

2. **Edit your** `<TS_HOME>/bin/ts.jte` **file and set the following environment variables:**

   a. **Set the** `webServerHost` **property to the name of the host on which your Web server is running.**

   The default setting is `localhost`.

   b. **Set the** `webServerPort` **property to the port number of the host on which the Web server is running.**

   The default setting is `8080`.

   c. **Set the** `local.classes` **property to point to the SAAJ RI classes/jars. You also need to include the** `servlet-api.jar`**, which you can include from the appropriate location within the reference implementation under test.**

   For the Apache Tomcat 5.5.*x* standalone Web Container your `local.classes` property setting should be:

   ```
   local.classes=${env.SAAJ_HOME}/lib/saaj-api.jar:
   ${env.SAAJ_HOME}/lib/saaj-impl.jar:
   ${env.SAAJ_HOME}/lib/activation.jar:
   ${env.CATALINA_HOME}/common/lib/servlet-api.jar
   ```

   d. **Set the** `webserver.container` **property to the supported web container type used for testing.**

   The supported web container type for standalone web applications is Tomcat via `tomcat.xml`. By default this is set to `tomcat.xml`.

   e. **Set the** `webcontainer.home` **property to the location in which the Web container under test is installed.**

   The supported web container type for standalone Web applications is Tomcat, so this property should be set to where your Tomcat is installed.

- f. **Set the `porting.ts.url.class.1` property to your porting implementation class that is used for obtaining URLs.**

  The default setting for the Sun RI porting implementation is:

  ```
  com.sun.ts.lib.implementation.sun.common.SunRIURL
  ```

3 **Provide your own implementations of the porting package interfaces provided with the SAAJ TCK.**

The porting interfaces are:

- `TSURLInterface.java` — Obtains URL strings for web resources in an implementation-specific manner

API documentation for the `TSURLInterface.java` porting package interface is available in the `<TS_HOME>/docs/api` directory.

4.2 Deploying the SAAJ TCK Tests

▼ 4.2.1 To Deploy the SAAJ TCK Tests

1 **Change to the `<TS_HOME>/dist` directory.**

2 **Copy the SAAJ TCK WAR files to the appropriate container.**

For example, for the Apache Tomcat 5.5.*x* Web container, use the following command:

```
cp *war <CATALINA_HOME>/webapps
```

4.3 Configuring and Starting Your Java Web Server

Complete the following two procedures to configure your Web server environment for Java Web Services Developers Pack.

▼ 4.3.1 To Configure the Java Web Server

1 **Set the following environment variables in your shell environment:**

- a. `JAVA_HOME` **to the directory in which Java SE 5 is installed**
- b. `TS_HOME` **to the directory in which the SAAJ TCK 1.3 software is installed**
- c. `SAAJ_HOME` **to the directory in which the SAAJ RI software is installed**
- d. `CATALINA_HOME` **to the directory in which the Apache Tomcat 5.5.*x* Web Container is installed**

2 **Copy the required jar files to the Web server shared library directory:**

For the Apache Tomcat 5.5.*x* Web Container, use the following commands:

```
cp <TS_HOME>/lib/tsharness.jar <TS_HOME>/lib/saajtck.jar <CATALINA_HOME>/shared/lib
cp <SAAJ_HOME>/lib/*.jar <CATALINA_HOME>/shared/lib
```

▼ 4.3.2 To Start the Java Web Server

◗ **Execute the `<CATALINA_HOME>/bin/startup.sh` script or the `<CATALINA_HOME>/bin/startup.bat` batch file.**

Wait for the Web server to come up. Look at the file `<CATALINA_HOME>/logs/launcher.server.log` and wait until the messages stop printing to the file. After a series of startup and initialization messages, you will see the following message when the web server has completed its startup:

```
[INFO] Http11Protocol - -Starting Coyote HTTP/1.1 on port 8080
[INFO] Catalina - -Server startup in 126511 ms
```

4.4 Using the JavaTest Harness Software

There are two general ways to run the SAAJ TCK test suite using the JavaTest harness software:

- Through the JavaTest GUI; if using this method, please continue on to Using the JavaTest Harness Configuration GUI.
- In JavaTest batch mode, from the command line in your shell environment; if using this method, please proceed directly to Chapter 5.

4.5 Using the JavaTest Harness Configuration GUI

You can use the JavaTest harness GUI to modify general test settings and to quickly get started with the default SAAJ TCK test environment. This section covers the following topics:

- "4.5.1 Configuration GUI Overview" on page 31
- "4.5.2 Starting the Configuration GUI" on page 31
- "4.5.3 The Configuration Interview" on page 32
- "4.5.4 Modifying the Default Test Configuration" on page 33

Note – It is only necessary to proceed with this section if you want to run the JavaTest harness in GUI mode. If you plan to run the JavaTest harness in command-line mode, skip the remainder of this chapter, and continue with Chapter 5.

4.5.1 Configuration GUI Overview

In order for the JavaTest harness to execute the test suite, it requires information about how your computing environment is configured. The JavaTest harness requires two types of configuration information:

- **Test environment** — This is data used by the tests. For example, the path to the Java runtime, how to start the product being tested, network resources, and other information required by the tests in order to run. This information does not change frequently and usually stays constant from test run to test run.
- **Test parameters** — This is information used by the JavaTest harness to run the tests. Test parameters are values used by the JavaTest harness that determine which tests in the test suite are run, how the tests should be run, and where the test reports are stored. This information often changes from test run to test run.

The first time you run the JavaTest harness software, you are asked to specify the test suite and work directory that you want to use. (These parameters can be changed later from within the JavaTest harness GUI.)

Once the JavaTest harness GUI is displayed, whenever you choose *Run Tests->Start* to begin a test run, the JavaTest harness determines whether all of the required configuration information has been supplied:

- If the test environment and parameters have been completely configured, the test run starts immediately.
- If any required configuration information is missing, the configuration editor displays a series of questions asking you the necessary information. This is called the configuration interview . When you have entered the configuration data, you are asked if you wish to proceed with running the test.

4.5.2 Starting the Configuration GUI

Before you start the JavaTest harness software, you must have a valid test suite and Java SE 5 installed on your system.

The SAAJ TCK includes an Ant script that is used to execute the JavaTest harness from the `<TS_HOME>` directory. Using this Ant script to start the JavaTest harness is part of the procedure described in The Configuration Interview.

When you execute the JavaTest harness software for the first time, the JavaTest harness displays a Welcome dialog box that guides you through the initial startup configuration.

- If it is able to open a test suite, the JavaTest harness displays a Welcome to JavaTest dialog box that guides you through the process of either opening an existing work directory or creating a new work directory as described in the JavaTest online help.

- If the JavaTest harness is unable to open a test suite, it displays a Welcome to JavaTest dialog box that guides you through the process of opening both a test suite and a work directory as described in the JavaTest documentation.

After you specify a work directory, you can use the Test Manager to configure and run tests as described in The Configuration Interview.

4.5.3 The Configuration Interview

The answers you give to some of the configuration interview questions are specific to your site. For example, the name of the host on which the JavaTest harness is running. Other configuration parameters can be set however you wish. For example, where you want test report files to be stored.

▼ 4.5.3.1 To configure the JavaTest harness to run the SAAJ TCK tests

Note that you only need to complete all these steps the first time you start the JavaTest test harness. After you complete these steps, you can either run all of the tests by completing the steps in Starting JavaTest or run a subset of the tests by completing the steps in Running a Subset of the Tests.

1. **Change to the `<TS_HOME>/bin` directory and start the JavaTest test harness:**

   ```
   cd <TS_HOME>/bin
   ./tsant gui
   ```

   If the Welcome Screen does not appear do the following, otherwise skip to next step.

2. **Click *File->Open Quick Start Wizard*.**

   The Welcome screen displays.

3. **Click *Start a new test run*, and then click *Next*.**

   You are prompted to Create a new configuration or use a configuration template.

4. **Select *Create a new configuration*, and then click *Next*.**

   You are prompted to select a test suite.

5. **Accept the default suite (`<TS_HOME>/src`), and then click *Next*.**

   You are prompted to specify a work directory to use to store your test results.

6. **Type a work directory name or use the *Browse* button to select a work directory, and then click *Next*.**

   You are prompted to start the configuration editor or start a test run. At this point, the SAAJ TCK is configured to run the default test suite.

7. **Uncheck the *Start the configuration editor* option, and then click *Finish*.**

8 **Click Run Tests->Start.**

The JavaTest harness starts running the tests.

9 **To Reconfigure the JavaTest test harness, click** *Configuration->New Configuration* **or** *Configuration->Change Configuration***.**

10 **Click** *Report->Create Report***.**

11 **Specify the directory in which the JavaTest test harness will write the report, and then click** *OK***.**

A report is created, and you are asked whether you want to view it.

12 **Click** *Yes* **to view the report.**

4.5.4 Modifying the Default Test Configuration

The JavaTest GUI enables you to configure numerous test options. These options are divided into two general dialog box groups:

- **Group 1** — Available from the JavaTest *Configure->Change Configuration* submenus, the following options are displayed in a tabbed dialog box:
  - *Tests to Run*
  - *Exclude List*
  - *Keywords*
  - *Prior Status*
  - *Test Environment*
  - *Concurrency*
  - *Timeout Factor*
- **Group 2** — Available from the JavaTest *Configure->Change Configuration->Other Values* submenu, or by pressing Ctrl+E, the following options are displayed in a paged dialog box:
  - *Environment Files*
  - *Test Environment*
  - *Specify Tests to Run*
  - *Specify an Exclude List*

Note that there is some overlap between the functions in these two dialog boxes; for those functions use the dialog that is most convenient for you. Please refer to the JavaTest Harness documentation or the online help for complete information about these various options.

◆ ◆ ◆ CHAPTER 5

Executing Tests

The SAAJ TCK uses the JavaTest harness to execute the tests in the test suite. For detailed instructions that explain how to run and use JavaTest, see the *JavaTest User's Guide and Reference* in the documentation bundle.

This chapter includes the following topics:

Note – The instructions in this chapter assume that you have installed and configured your test environment as described in Chapter 3 and Chapter 4, respectively.

5.1 Starting JavaTest

There are two general ways to run the SAAJ TCK using the JavaTest harness software:

- Through the JavaTest GUI
- From the command line in your shell environment

Note – The `tsant` command referenced in the following two procedures and elsewhere in this guide is a wrapper around the Ant build tool, which is included in the SAAJ TCK bundle. The `build.xml` file in `<TS_HOME>/bin` contains the various Ant targets for the SAAJ TCK test suite

---

▼ 5.1.1 To Start JavaTest in GUI Mode

◗ **Change to the `<TS_HOME>/bin` directory and execute the `tsant gui` target:**

```
cd <TS_HOME>/bin
./tsant gui
```

▼ 5.1.2 To Start JavaTest in Command-Line Mode

1. **Change to any subdirectory under** `<TS_HOME>/src/com/sun/ts/tests`.
2. **Execute the** `tsant runclient` **target to start the JavaTest run:**

   ```
   <TS_HOME>/bin/tsant runclient
   ```

   For example, to run the SAAJ TCK signature tests, enter the following commands:

   ```
   cd <TS_HOME>/src/com/sun/ts/tests/signaturetest/saaj
   <TS_HOME>/bin/tsant runclient
   ```

5.2 Running a Subset of the Tests

▼ 5.2.1 To Run a Subset of Tests in GUI Mode

1. **Click** *Configure->Change Configuration->Tests to Run* **from the JavaTest main menu.**

   The tabbed Configuration Editor dialog is displayed.
2. **Click** *Specify* **from the option list on the left.**
3. **Select the tests you want to run from the displayed test tree, and then click** *Done***.**

   You can select entire branches of the test tree, or use Ctrl+Click or Shift+Click to select multiple tests or ranges of tests, respectively.
4. **Click** *Save File***.**
5. **Click** *Run Tests->Start* **to run the tests you selected.**

   Alternatively, you can right-click the test you want from the test tree in the left pane of the JavaTest main window, and choose *Execute These Tests* from the popup menu.
6. **Click** *Report->Create Report***.**
7. **Specify the directory in which the JavaTest test harness will write the report, and then click** *OK***.**

   A report is created, and you are asked whether you want to view it.
8. **Click** *Yes* **to view the report.**

▼ 5.2.2 To Run a Subset of Tests in Command-Line Mode

1 **Change to the directory containing the tests you want to run.**

For example, `<TS_HOME>/src/com/sun/ts/tests/signaturetest/saaj`.

2 **Start the test run by executing the following command:**

```
<TS_HOME>/bin/tsant runclient
```

The tests in `<TS_HOME>/src/com/sun/ts/tests/signaturetest/saaj` and its subdirectories are run.

▼ 5.2.3 To Run a Subset of Tests in Batch Mode Based on Prior Result Status

You can run certain tests in batch mode based on the test's prior run status by specifying the `priorStatus` system property when invoking `tsant`.

◗ **Invoke `tsant` with the `priorStatus` property.**

The accepted values for the `priorStatus` property are any combination of the following:

- `fail`
- `pass`
- `error`
- `notRun`

For example, you could run all the SAAJ tests with a status of failed and error by invoking the following commands:

```
cd $TS_HOME/src/com/sun/ts/tests/signaturetest
$TS_HOME/bin/tsant -DpriorStatus="fail,error" runclient
```

Note that multiple `priorStatus` values must be separated by commas.

5.3 Building/Deploying/Running SAAJ TCK Tests Using `tsant`

You can use `tsant`, an implementation of Ant bundled with the SAAJ TCK, to build, deploy, and run the test suite.

▼ 5.3.1 To Configure Your Build Environment

1 **Set the following environment variables in your shell environment to use the build infrastructure that comes with the TCK:**

a. `TS_HOME` **to the directory in which the SAAJ TCK software is installed.**

- *C Shell*

  ```
  setenv TS_HOME /path_to_saajtck
  ```

- *Bourne Shell*

  ```
  TS_HOME=/path_to_saajtck
  export TS_HOME
  ```

b. `ANT_HOME` **to the directory in which the ANT software is installed.**

- *C Shell*

  ```
  setenv ANT_HOME ${TS_HOME}/tools/ant
  ```

- *Bourne Shell*

  ```
  ANT_HOME=${TS_HOME}/tools/ant
  export ANT_HOME
  ```

c. `TS_HOME/bin` **to your** `PATH` **in your command shell.**

- *C Shell*

  ```
  setenv PATH ${TS_HOME}/bin:${PATH}
  ```

- *Bourne Shell*

  ```
  PATH=${TS_HOME}/bin:${PATH}
  export PATH
  ```

d. `JAVA_HOME` **to the directory in which the Java SE 5 software is installed.**

- *C Shell*

  ```
  setenv JAVA_HOME /path_to_jdk15
  ```

- *Bourne Shell*

  ```
  JAVA_HOME=/path_to_jdk15
  export TS_HOME
  ```

2 **Change to the `<TS_HOME>/install/saaj` directory and run ant to create and set up your `<TS_HOME>/bin` directory if it has not yet been set up.**

3 **Change to the `<TS_HOME>/bin` directory and edit the `ts.jte` file to set the required properties needed for using Ant build environment.**

  a. `webserver.home` **to the directory in which the Java Web Services Developers Pack is installed.**

  b. `webserver.host` **to the host on which the Web server is running.**

  c. `webserver.port` **to the port on which the Web server is running.**

  d. `webserver.container` **to supported web container used for testing.**

▼ 5.3.2 To Build the Tests

1 **To build a single test directory, type the following:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api/javax_xml_soap/AttachmentPart
tsant clean build
```

This cleans and builds the tests in the `AttachmentPart` test directory.

2 **To list the classes directory for this test that was built, type:**

```
tsant lc
```

or

```
tsant llc
```

3 **To list the distribution directory of archives for this test that was built, type:**

```
tsant ld
```

or

```
tsant lld
```

4 **To build a subset of test directories, type:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api
tsant clean build
```

This cleans and builds all the test directories under the `api` directory.

▼ 5.3.3 To Deploy the Tests

1. **To deploy a single test directory, type the following:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api/javax_xml_soap/AttachmentPart
tsant deploy
```

This deploys the WAR file built for the `AttachmentPart` test directory to the `webapps` directory.

2. **To deploy a subset of test directories, type:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api
tsant deploy
```

This deploys all the test WAR files for the `api` directory.

▼ 5.3.4 To Run the Tests

1. **To run a single test directory, type the following:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api/javax_xml_soap/AttachmentPart
tsant runclient
```

This runs all tests in the `AttachmentPart` test directory.

2. **To run a single test within a test directory, type:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api/javax_xml_soap/AttachmentPart
tsant runclient -Dtest=getMimeHeader1Test
```

This runs only the `getMimeHeader1Test` in the `AttachmentPart` test directory. You select the test name to run by looking at the `testName` tags in the `URLClient.java` file.

3. **To run a subset of test directories type:**

```
cd <TS_HOME>/src/com/sun/ts/tests/saaj/api
tsant runclient
```

This runs all the test directories under the `api` directory.

5.4 Test Reports

A set of report files is created for every test run. These report files can be found in the report directory you specify. After a test run is completed, the JavaTest harness writes HTML reports for the test run. You can view these files in the JavaTest ReportBrowser when running in GUI mode, or in the web browser of your choice outside the JavaTest interface.

To see all of the HTML report files, enter the URL of the `report.html` file. This file is the root file that links to all of the other HTML reports.

The JavaTest harness also creates a `summary.txt` file in the report directory that you can open in any text editor. The `summary.txt` file contains a list of all tests that were run, their test results, and their status messages.

5.4.1 Creating Test Reports

▼ 5.4.1.1 To Create a Test Report in GUI Mode

1. **Click *Report->Create Report* from the JavaTest main menu.**

   You are prompted to specify a directory to use for your test reports. The default location is `<TS_HOME>/src/com/sun/ts/tests/signaturetests/saaj`.

2. **Specify the directory you want to use for your reports, and then click *OK*.**

   Use the *Filter* drop-down list to specify whether you want to generate reports for the current configuration, all tests, or a custom set of tests.

   You are asked whether you want to view report now.

3. **Click *Yes* to display the new report in the JavaTest ReportBrowser.**

▼ 5.4.1.2 To Create a Test Report in Command-Line Mode

◗ **Specify where you want to create the test report.**

a. **To specify the report directory from the command line at runtime, use:**

```
<TS_HOME>/bin/tsant -Dreport.dir="<report_dir>"
```

Reports are written for the last test run to the directory you specify. The default location is `<TS_HOME>/src/com/sun/ts/tests/signaturetests/saaj`.

b. **To specify the default report directory, set the `report.dir` property in `<TS_HOME>/bin/ts.jte`.**

For example, `report.dir="/home/josephine/reports"`.

c. **To disable reporting, set the `report.dir` property to `"none"`, either on the command line or in `ts.jte`.**

For example:

```
<TS_HOME>/bin/tsant -Dreport.dir="none"
```

5.4.2 Viewing an Existing Test Report

▼ 5.4.2.1 To View an Existing Report in GUI Mode

1. **Click Report->Open Report from the JavaTest main menu.**

   You are prompted to specify the directory containing the report you want to open.

2. **Select the report directory you want to open, and then click *Open*.**

   The selected report set is opened in the JavaTest ReportBrowser.

▼ 5.4.2.2 To View an Existing Report in Command-Line Mode

◗ **Use the Web browser of your choice to view the `report.html` file in the report directory you specified from the command line or in `ts.jte`.**

◆ ◆ ◆ CHAPTER 6

Debugging Test Problems

There are a number of reasons that tests can fail to execute properly. This chapter provides some approaches for dealing with these failures. Please note that most of these suggestions are only relevant when running the test harness in GUI mode.

6.1 Overview

The goal of a test run is for all tests in the test suite that are not filtered out to have passing results. If the root test suite folder contains tests with errors or failing results, you must troubleshoot and correct the cause to satisfactorily complete the test run.

- **Errors** — Tests with errors could not be executed by the JavaTest harness. These errors usually occur because the test environment is not properly configured.
- **Failures** — Tests that fail were executed but had failing results.

The Test Manager GUI provides you with a number of tools for effectively troubleshooting a test run. See the *JavaTest User's Guide* and JavaTest online help for detailed descriptions of the tools described in this chapter.

6.2 Test Tree

Use the test tree in the JavaTest GUI to identify specific folders and tests that had errors or failing results. Color codes are used to indicate status as follows:

- **Green** — Passed
- **Blue** — Test Error
- **Red** — Failed to pass test
- **White** — Test not run
- **Gray** — Test filtered out (not run)

6.3 Folder Information

Click a folder in the test tree in the JavaTest GUI to display its tabbed pane.

Choose the *Error* and the *Failed* panes to view the lists of all tests in and under a folder that were not successfully run. You can double-click a test in the lists to view its test information.

6.4 Test Information

To display information about a test in the JavaTest GUI, click its icon in the test tree or double-click its name in a folder status pane. The tabbed pane contains detailed information about the test run and, at the bottom of the pane, a brief status message identifying the type of failure or error. This message may be sufficient for you to identify the cause of the error or failure.

If you need more information to identify the cause of the error or failure, use the following panes listed in order of importance:

- **Test Run Messages** contains a Message list and a Message pane that display the messages produced during the test run.
- **Test Run Details** contains a two column table of name/value pairs recorded when the test was run.
- **Configuration** contains a two column table of the test environment name/value pairs derived from the configuration data actually used to run the test.

Note – You can set `harness.log.traceflag=true` in `<TS_HOME>/bin/` `ts.jte` to get more debugging information.

6.5 Report Files

Report files are another good source of troubleshooting information. You may view the individual test results of a batch run in the JavaTest Summary window, but there are also a wide range of HTML report files that you can view in the JavaTest ReportBrowser or in the external browser or your choice following a test run. See Section 5.3 "Test Reports" for more information.

6.6 Configuration Failures

Configuration failures are easily recognized because many tests fail the same way. When all your tests begin to fail, you may want to stop the run immediately and start viewing individual test output. However, in the case of full-scale launching problems where no tests are actually processed, report files are usually not created (though sometimes a small `harness.trace` file in the report directory is written).

◆ ◆ ◆ APPENDIX A

Frequently Asked Questions

A.1 Where do I start to debug a test failure?

From the JavaTest GUI, you can view recently run tests using the Test Results Summary, by selecting the red Failed tab or the blue Error tab. See Chapter 6 for more information.

A.2 How do I restart a crashed test run?

If you need to restart a test run, you can figure out which test crashed the test suite by looking at the `harness.trace` file. The `harness.trace` file is in the report directory that you supplied to the JavaTest GUI or parameter file. Examine this trace file, then change the JavaTest GUI initial files to that location or to a directory location below that file, and restart. This will overwrite only `.jtr` files that you rerun. As long as you do not change the value of the GUI work directory, you can continue testing and then later compile a complete report to include results from all such partial runs.

A.3 What would cause tests be added to the exclude list?

The JavaTest exclude file (`*.jtx`) contains all tests that are not required by Sun Microsystems to be run. The following is a list of reasons for a test to be included in the Exclude List:

- An error in a Sun Microsystems reference implementation that does not allow the test to execute properly has been discovered.
- An error in the specification that was used as the basis of the test has been discovered.
- An error in the test has been discovered.