JAIN SLEE TCK 1.0 User's Guide

November 26, 2004

Contents

Preface

0.1 License Terms

0.2 Overview

This guide describes how to install, configure, and run the Technology Compatibility Kit (TCK) that is used to test an implementation of the JAIN Service Logic Execution Environment 1.0 specification, and how to develop the components required for running the TCK.

The JAIN SLEE TCK 1.0 is designed as a portable, configurable automated test suite for verifying the compliance of a licensee's implementation of the JAIN SLEE 1.0 Specification (hereafter referred to as a licensee implementation). The JAIN SLEE TCK 1.0 uses the JavaTest harness version 3.x to run the test suite.

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.

0.3 Who Should Use This Book

This guide is for licensees of JAIN SLEE 1.0 technology to assist them in running the test suite that verifies compliance of their implementation of the JAIN SLEE 1.0 Specification.

0.4 Before You Read This Book

Before reading this guide, you should familiarize yourself with the Java™ programming language and the JAIN SLEE 1.0 Specification. A good resource for the Java programming language is the Sun Microsystems, Inc. web site, located at: java.sun.com

The JAIN SLEE TCK 1.0 is based on the JAIN SLEE Specification 1.0. Links to the specification and other product information can be found on the Web at: www.jcp.org/jsr/detail/22.jsp

Before using the JAIN SLEE TCK 1.0, it is recommened at you read and become familiar with the JAIN SLEE Specification 1.0 and also the *JavaTest User's Guide* describing the main JavaTest harness. It is located at:

TCK_DIRECTORY/doc/javatest/javatest.pdf

in the JAIN SLEE TCK 1.0 distribution.

0.5 How This Book Is Organized

Chapter 1, "Introduction" gives an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and de-

scribes the JAIN SLEE TCK 1.0. It also includes a listing of the basic steps needed to get up and running with the JAIN SLEE TCK 1.0.

Chapter 2, "Procedure for JAIN SLEE 1.0 Certification" describes the conformance testing procedure and testing requirements.

Chapter 3, "Writing the JAIN SLEE TCK 1.0 Resource Adaptor" details the requirements for the JAIN SLEE TCK 1.0 resource adaptor needed to test the JAIN SLEE 1.0 implementation.

Chapter 4, "Installing the JAIN SLEE TCK 1.0" describes JAIN SLEE TCK 1.0 installation procedures.

Chapter 5, "Starting and Configuring the JavaTest Harness" describes loading the JavaTest harness software and basic JAIN SLEE TCK 1.0 set up and configuration.

Chapter 6, "Preparing the SLEE" details how to prepare the JAIN SLEE 1.0 implementation for TCK testing.

Chapter 7, "Verifying the JAIN SLEE TCK 1.0" describes how to start the JAIN SLEE TCK 1.0 and verify that it is properly configured.

Chapter 8, "Using the JAIN SLEE TCK 1.0" describes how to use the JAIN SLEE TCK 1.0 to test your implementation.

Chapter 9, "Testing API Signatures" describes how to use and run the signature test.

Chapter 10, "Test-Specific Information" provides information about the individual tests in the test suite.

Chapter 11, "Debugging Test Problems" describes some methods that can be used to trouble-shoot tests that fail.

Appendix A, "Frequently Asked Questions" provides answers to frequently asked questions.

0.6 Related Books

- *JavaTest User's Guide* and *JavaTest online help* (the JavaTest User's Guide is located at `doc/javatest/javatest.pdf` in the JAIN SLEE TCK 1.0 distribution)
- *The JAIN Service Logic Execution Environment 1.0 specification*
- *The Java Programming Language*

- *The Java Language Specification Second Edition*
- *The Java Virtual Machine Specification 2nd Edition, Java 2 Platform*

0.7 Accessing Documentation Online

When unzipped and installed, the JAIN SLEE TCK 1.0 includes a `doc/` directory that contains this manual and the *JavaTest User's Guide*, both in Adobe Acrobat™ PDF format.

The JavaTest User's Guide is located at:
TCK_DIRECTORY`/doc/javatest/javatest.pdf`

This JAIN SLEE TCK 1.0 User's Guide is located at:
TCK_DIRECTORY`/doc/sleetck-user-guide.pdf`

You may also access other Java technology related documentation online at: java.sun.com.

0.8 Typographic Conventions Used in This Book

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

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
AaBbCc123	What you type, when contrasted with on-screen computer output	**% su** `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename*.

Note that the top-most JAIN SLEE TCK 1.0 installation directory, is referred to as *TCK_DIRECTORY* throughout the JAIN SLEE TCK 1.0 documentation.

0.9 Notes on the text

This user guide is based on the Java™Technology Compatibility Kit User's Guide Template For Technology Licensees, release 1.1.1, written by Sun Microsystems, Inc. Sections have been added, removed and modified to cover the JAIN SLEE TCK 1.0.

Chapter 1

Introduction

This chapter gives an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and describes the JAIN SLEE TCK 1.0. It also includes a listing of what is needed to get up and running with the JAIN SLEE TCK 1.0. It contains the following sections:

- Compatibility Testing 1.1
- Java Community Process (JCP) Program and Compatibility Testing 1.2
- The JAIN SLEE TCK 1.0 1.3
- JAIN SLEE TCK 1.0 Configuration 1.4
- Getting Started 1.5

1.1 Compatibility Testing

Java technologies are *"cross-platform,"* meaning that they run on different hardware platforms and operating systems. Compatibility testing is the process of testing a technology implementation to make sure that it operates consistently with each platform, operating system, and other implementations of the same Java technology specification.

Therefore, compatibility testing differs from traditional product testing in a number of ways because 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 to ensuring 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.
- Compatibility 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, "Procedure for JAIN JAIN SLEE 1.0 Certification".

1.1.3 TCK Overview

A TCK is a set of tools and tests used to verify that a licensee's implementation of a 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 the technology to the applicable specification of the technology. Compatibility testing is a means of ensuring correctness, completeness, and consistency across all implementations developed by licensees of the technology.

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 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.

1.2 Java Community Process (JCP) Program and Compatibility Testing

The Java Community Process^{SM} (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^{SM} 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 on the JCP program see this URL: jcp.org

1.3 The JAIN SLEE TCK 1.0

The JAIN SLEE TCK 1.0 is designed as a portable, configurable, automated test suite for verifying the compliance of a licensee's implementation of the JAIN SLEE 1.0 Specification. The JAIN Service Logic Execution Environment 1.0 specification can be found at: www.jcp.org/jsr/detail/22.jsp. The JAIN SLEE TCK 1.0 uses the JavaTest harness version 3.x to run the test suite.

1.3.1 JAIN SLEE TCK 1.0 Specifications and Requirements

This section lists the applicable requirements and specifications.

- **SLEE Version**. The JAIN SLEE TCK 1.0 is based on the JAIN SLEE Specification version 1.0.
- **Specification Requirements**. Requirements for a JAIN SLEE 1.0 implementation are described in detail in the JAIN SLEE 1.0 Specification. Links to the JAIN SLEE 1.0 specification and other product information can be found at www.jcp.org/jsr/detail/22.jsp(previosuly said hardware and software requirements)
- **JavaTest harness.** The JAIN SLEE TCK 1.0 requires version 3.x of the JavaTest harness.
- **Reference Implementation**. The designated Reference Implementation for conformance testing of implementations based upon JAIN SLEE Specification 1.0.
- **Platform requirements**. The following requirements must be met in order to run the JAIN SLEE TCK 1.0 on the host system:
  - **Operating system**. You may use any hardware platform that supports Java. The JAIN SLEE TCK 1.0 has been tested on Linux, Solaris, Windows NT and Windows XP.
  - **Disk space**. At least 60 Megabytes of free disk space available for installation of the JAIN SLEE TCK 1.0, temporary files, and for the creation of report files.

– **Memory**. At least 64 Mb of RAM is recommended for running the JAIN SLEE TCK 1.0 on Windows platforms.

Ant Licensees wishing to build the JAIN SLEE TCK 1.0 will need Ant 1.5.1 or higher. (Available from http://jakarta.apache.org/ant/)

1.3.2 JAIN SLEE TCK 1.0 Components

This section describes the main components that make up the JAIN SLEE TCK 1.0.

JavaTest Harness

The JavaTest™harness version 3.x 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 on-line within the JavaTest GUI, or off-line 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 5, "Starting and Configuring the JavaTest Harness.

The JavaTest harness runs tests on a vendor's JAIN SLEE 1.0 implementation or the JAIN SLEE 1.0 Reference Implementation running on a platform system.

TCK Compatibility Rules

Compatibility criteria for a technology implementation are identified in the TCK Compatibility Rules. The TCK tests a technology implementation to make sure that it adheres to those rules.

The compatibility rules that apply to JAIN SLEE 1.0 are described in Chapter 2, "Procedure for JAIN SLEE 1.0 Certification."

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 JAIN SLEE TCK 1.0 to test an implementation of the JAIN SLEE 1.0 implementation.

The tests are designed to verify that a licensee's run-time 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.

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 included in the JAIN SLEE TCK 1.0 is located in the `testsuite/` directory and has a file name based on the corresponding JAIN SLEE TCK version number. For example, the Exclude List for version 1.0 of the JAIN SLEE TCK is *TCK_HOME*`/testsuite/jain-slee-tck-1_0.jtx`.

Note —- From time to time, updates to the Exclude List are made available on the JAIN SLEE TCK 1.0 web site. You should always make sure you are using an up-to-date copy of the Exclude List before running the JAIN SLEE TCK 1.0 to verify your implementation.

A test might be included in an 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 also tested against the JAIN SLEE 1.0 Reference Implementation. Any tests that fail when run on the JAIN SLEE 1.0 Reference Implementation 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 "JAIN SLEE 1.0 Test Appeals Process" on page 15

1.4 JAIN SLEE TCK 1.0-Configuration

Use the JavaTest harness Graphic User Interface (GUI) to configure test runs, as described in Chapter 5, "Starting and Configuring the JavaTest Harness."

1.5 Getting Started

This section provides an general overview of what needs to be done to install, set up, test, and use the JAIN SLEE TCK 1.0:

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

   - **Java software**. Make sure that the Java technology J2SE™version 1.4.1 or later is installed.
   - **JavaTest harness version 3.x.**

   Consult the documentation for each of these software application for installation instructions.

2. **Install the JAIN SLEE TCK 1.0 on the system hosting the JavaTest harness.**
   See Chapter 4, "Installing the JAIN SLEE TCK 1.0" for more information.

3. **Install the implementation of JAIN SLEE 1.0 that is under test, such that the JavaTest harness can access it via RMI.**

4. **Start up and configure the JavaTest harness.**
   Use the JavaTest harness configuration editor interview to enter the basic configuration parameters it needs. (See Chapter 5, "Starting and Configuring the JavaTest Harness.")

5. **Verify the JavaTest harness Installation.**
   Test that the JavaTest harness is running correctly with a simple series of tests. (See "Verifying Installation and Setup" on page 40.)

6. **Test the Full JAIN SLEE 1.0 Implementation**
   Test the full JAIN SLEE 1.0 implementation by running the entire test suite on the JAIN SLEE 1.0 implementation. (See "Using the JAIN SLEE TCK 1.0 to Test a JAIN SLEE 1.0 Implementation" on page 44.)

Chapter 2

Procedure for JAIN SLEE TCK 1.0 Certification

This chapter describes the compatibility testing procedure and compatibility requirements.

2.1 Certification Overview

- 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," below.

2.2 Compatibility Requirements

This section described the compatibility rules and defines the terms used in those rules.

2.2.1 Definitions

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

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.

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^{SM} (JCP^{SM}), for the Technology Under Test.

The Libraries for JAIN SLEE 1.0 are listed at the end of this chapter.

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 JCP member responsible for maintaining the Specification, reference implementation, and TCK for the Technology. Open Cloud and Sun Microsystems, Inc are the Maintenance Leads for JAIN SLEE 1.0.

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 of a Runtime can be binary (enable/disable optimization), an enumeration (select from a list of localizations), or a range (set the initial Runtime heap size).

Product
A licensee product in which a Runtime 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 Runtime supporting an Operating Mode that permits selection of an initial heap size might have a Product Configuration that sets the initial heap size to 1 Mb.

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.

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 JCP 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 JCP.

Technology Under Test
Specifications and the reference implementation for JAIN SLEE 1.0.

Test Result
Each test returns a Test Result, which falls into of the following categories:

Test Passed
A "Test Passed" result indicates that the test was successful: the JAIN SLEE 1.0 implementation was found to be compliant with the JAIN SLEE 1.0 specification in the tested area.

Test Failed
A "Test Failed" result indicates that the test was un successful: the JAIN SLEE 1.0 implementation was found to be non-compliant with the JAIN SLEE 1.0 specification in the tested area.

Test Error
A "Test Error" result indicates that the relevant behaviour could not be tested due to an error.

A Test Error result may indicate an error either in the test, the JAIN SLEE 1.0 implementation, or a configuration error.

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 JCP.

2.3 Rules for JAIN SLEE 1.0 Products

For each version of an operating system, software component, and hardware platform Documented as supporting the Product:

SLEE1.
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.

SLEE1.1
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.

SLEE1.2
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.

SLEE1.3
A Product may contain an Operating Mode that selects the Edition with which it is compatible. The Product must meet the compatibility requirements for the corresponding Edition for all Product Configurations of this Operating Mode. This Operating Mode must affect no smaller unit of execution than an entire Application.

SLEE2.

Some Conformance Tests may have properties that may be changed. Apart from changing such properties 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 JAIN SLEE TCK 1.0 web site and apply to all licensees.

SLEE3.

The testing tools supplied as part of the Test Suite or as updated by the Maintenance Lead must be used to certify compliance.

SLEE4.

The Exclude List associated with the Test Suite cannot be modified.

SLEE5.

The Maintenance Lead can define exceptions to these Rules. Such exceptions would be made available to and apply to all licensees.

SLEE6.

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.

SLEE7.

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 in 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.

SLEE7.1

If a Product includes Technologies in addition to the Technology Under Test, then it must contain the full set of combined public and protected classes and interfaces. The API of the Product must contain the union of the included Technologies. No further subsetting, supersetting, or modifications to the APIs of the included Technologies are allowed.

SLEE8.
Except for tests specifically required by this TCK to be recompiled (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.

SLEE9.
The functional programmatic behavior of any binary class or interface must be that defined by the Specifications.

2.4 JAIN SLEE 1.0 Test Appeals Process

The Maintenance Lead will be the point of contact for all test challenges to the Test Suite for JAIN SLEE 1.0. Correspondence related to test challenges should be sent to slee-tck@opencloud.com.

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 JAIN SLEE TCK 1.0. 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:

1. Test has bugs (i.e., program logic errors)
2. Specification item covered by the test is ambiguous
3. Test does not match the specification
4. Test assumes unreasonable hardware and/or software requirements
5. 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 the Maintenance Lead and include all relevant information as described in the Test Challenge form below. The process used to determine the validity or invalidity of a test (or related group of tests) is described in "JAIN SLEE TCK 1.0 Test Appeals Steps" on page 16.

All tests found to be invalid will either be placed on the Exclude List for that version of the JAIN SLEE TCK 1.0 or have an alternate test made available as follows:

- 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 JAIN SLEE TCK 1.0 licensees on the JAIN SLEE TCK 1.0 web site.

- The 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 JAIN SLEE TCK 1.0 web site.

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

2.4.1 JAIN SLEE TCK 1.0 Test Appeals Steps

1. **JAIN SLEE 1.0 licensee writes a test challenge to the Maintenance Lead contesting the validity of one or a related set of JAIN SLEE 1.0 tests.**

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

2. **The Maintenance Lead evaluates the challenge.**

   If the appeal is incomplete or unclear, it is returned to the submitting licensee for correction. If all is in order, the Maintenance Lead will check with the test developers to review the purpose and validity of the test before writing a response. The Maintenance Lead 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), the Maintenance Lead 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, the Maintenance Lead places the tests on the Exclude List for that version of the JAIN SLEE TCK 1.0 (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 on the JAIN SLEE TCK 1.0 web site. If the test is valid but difficult to pass due to hardware or operating system limitations, the Maintenance Lead 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 the Maintenance Lead to resolve the issue and only involve the EC as a last resort.**

Test Challenge Form	
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)	

Test Challenge Response Form	
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) -can be iterative-	
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., is an alternative test appropriate?)	

2.5 Specifications for JAIN SLEE 1.0

The Specifications for JAIN SLEE 1.0 are found on the JCP web site, at www.jcp.org/jsr/detail/22.jsp

2.6 Libraries for JAIN SLEE 1.0

```
javax.slee
javax.slee.connection
javax.slee.facilities
javax.slee.management
javax.slee.nullactivity
javax.slee.profile
javax.slee.resource
```

`javax.slee.serviceactivity`
`javax.slee.usage`

The above libaries are all included in the JAIN SLEE 1.0 API classes jar, which is available as part of the specification download at the JCP web site, and is included at `lib/slee.jar` in this distribution.

Chapter 3

Writing the JAIN SLEE TCK 1.0 Resource Adaptor

In order to run the JAIN SLEE TCK 1.0 against a JAIN SLEE 1.0 implementation, a resource adaptor must be written to support the JAIN SLEE TCK 1.0.

The JavaDoc documentation for the JAIN SLEE TCK 1.0 Resource Adaptor API details the implemenation requirements.

This documentation may be built using the 'javadoc' build target in the supplied build file `sleetck-ant.xml`.

The JAIN SLEE TCK 1.0 Resource and its interfaces are defined in the `com.opencloud.sleetck.lib.resource` package and sub-packages. The main interfaces used by the JAIN SLEE TCK 1.0 Resource Adaptor implementation are defined in the `com.opencloud.sleetck.lib.resource.adaptor` package. This documentation for these interfaces details the Resource Adaptor's obligations.

The JAIN SLEE TCK 1.0 Resource Adaptor provided for the JAIN SLEE 1.0 Reference Implementation provides a working example of a JAIN SLEE TCK 1.0 Resource Adaptor, and the source files for this Resource Adaptor provide a reference point for developers writing the JAIN SLEE TCK 1.0 Resource Adaptor for their JAIN SLEE 1.0 implementation. The resources for testing the JAIN SLEE 1.0 RI are packaged as `slee-ri-tck-resources.zip` in the base directory of the JAIN SLEE TCK 1.0 installation. The source package for the Resource Adaptor implementation is: com.opencloud.slee.resources.tck. Unbundling the `slee-ri-tck-resources.zip` file will put these sources at `tck/com/opencloud/slee/resources/tck`, relative to the installation directory used for `slee-ri-tck-resources.zip` (which should be the base directory of the JAIN SLEE 1.0 RI installation).

As a brief summary, the Resource Adaptor implementation must:

1. Instantiate the JAIN SLEE TCK 1.0 resource using the TCKResourceFactory.

2. Implement the TCKResourceEventHandler interface and register the implementation with the TCK Resource.

3. Fulfil all the requirements outlined in the com.opencloud.sleetck.lib.resource.adaptor package documentation

4. The Resource Adaptor must be depoyed as an implementation of the JAIN SLEE TCK 1.0 resource adaptor type. The relevant deployment descriptors are at `com/opencloud/sleetck/lib/resource/META-INF`

Chapter 4

Installing the JAIN SLEE TCK 1.0

This chapter describes JAIN SLEE TCK 1.0 installation procedures. It contains the following sections:

- Obtaining the Software 4.1
- Installing the JAIN SLEE TCK 1.0 software 4.2
- JAIN SLEE TCK 1.0 Contents 4.3

4.1 Obtaining the Software

The JAIN SLEE TCK 1.0 software is available via web download at www.jainslee.org/tck/download.html. The installation instructions are included in the installation archive.

4.2 Installing the JAIN SLEE TCK 1.0 Software

The following procedure details how to set up the JAIN SLEE TCK 1.0 for testing against the JAIN SLEE 1.0 Reference Implementation.

The installation procedure for testing other JAIN SLEE 1.0 implementations will vary in parts, but the installation procedure for testing the RI serves as a model for the installation process that JAIN SLEE 1.0 vendors may use.

The path separatory used in this document is '/'. Replace '/' with '\' for windows equivalents. Windows users should substitute .sh scripts with the .bat equivalents.

Some configuration steps depend on whether JavaTest will be invoked in GUI mode or batch mode. Note that either or both modes may be used, and is simply a matter of user preference. The GUI provides more intuitive control over test suite configuration and control, while the batch mode allows for more automated testing.

- Unzip the JAIN SLEE 1.0 RI archive
  % unzip jain-slee-1.0-ri.zip
  This will create a subdirectory `jain-slee-1.0-ri` under the current directory, which contains the JAIN SLEE 1.0 RI software. This new directory is the base directory of the JAIN SLEE 1.0 RI installation, and will be referred to as *SLEE_HOME* throughout this chapter.

- Unzip the JAIN SLEE TCK 1.0 archive
  % unzip jain-slee-1.0-tck.zip
  This will create a subdirectory `jain-slee-1.0-tck` under the current directory, which contains the JAIN SLEE TCK 1.0 software. This new directory is the base directory of the JAIN SLEE TCK 1.0 installation, and will be referred to as *TCK_HOME* throughout this chapter.

- Set environment variables
  Set the `JAVA_HOME` environment variable to reference a valid Java VM installation. Java 2 Standard Edition (J2SE^{TM}), version 1.4.1 or later is required. Set the `SLEE_HOME` environment variable to reference the base directory of the JAIN SLEE 1.0 RI installation.

- Unzip the JAIN SLEE TCK 1.0 resources archive into the JAIN SLEE 1.0 RI installation directory
  From *SLEE_HOME*: **% unzip *TCK_HOME*/slee-ri-tck-resources.zip**
  This creates a subdirectory `tck`

- Create and edit the configuration files for the JAIN SLEE 1.0 TCK resources
  For each template file in *SLEE_HOME*`/tck/config` with `.template` file extension, create a copy of the file with the `.template` extension removed, as per the instructions in the template file.

- Update `mlet.conf`
  Append the contents of *SLEE_HOME*`/tck/config/mlet.conf` to *SLEE_HOME*`/mlet/mlet.conf`, e.g.:
  % cat *SLEE_HOME*/tck/config/mlet.conf >> *SLEE_HOME*/mlet/mlet.conf
  The updated `mlet.conf` file should now contain an entry for the JAIN SLEE TCK 1.0 plugin.

- Create and edit JAIN SLEE TCK 1.0 configuration files
  For each template file in *TCK_HOME*`/config` with `.template` file extension, create a copy of the file with the `.template` extension removed, as per the instructions in the template file.
  Note that the `sleetck.jti.template` file may not need to be used (see the comments in that file).

- Set file permissions for executable scripts
  If necessary, grant execute access for the executable scripts in the *SLEE_HOME*`/bin`,*SLEE_HOME*`/tck/bin` and *TCK_HOME*`/bin` directories, e.g.: **% chmod +x *SLEE_HOME*/bin/*.sh *SLEE_HOME*/tck/bin/*.sh *TCK_HOME*/bin/*.sh**

- Insert an interview checksum
  If the `sleetck.jti.template` file was used to create a `sleetck.jti` configuration file, the file must be processed to insert a valid checksum, using *TCK_HOME*`/bin/insert-interview-checksum.sh` (or `.bat`):
  % *TCK_HOME*/bin/insert-interview-checksum.sh

- Start rmiregistry
  The rmiregistry must be started with the appropriate classpath and

port settings, using the `start-rmiregistry.sh` (or `.bat`) script:
% cd *TCK_HOME*; ./bin/start-rmiregistry.sh
Note that the rmiregistry must be run on the machine(s) on which the SLEE's MBeanServer, and the JAIN SLEE TCK 1.0 Resource will be running. These components will usually be run on the same machine as the SLEE. If these components will not be running on the same machine as the JAIN SLEE TCK 1.0, the `start-rmiregistry.sh` (or `.bat`) script should be copied to these machines, along with the referenced configuration and library files, and run on those machine(s).

- Start Cloudscape
  The Cloudscape database must be started prior to starting the RI: **% $J2EE_HOME/bin/cloudscape -start**

- Start the JAIN SLEE 1.0 RI
  % cd $*SLEE_HOME*; ./bin/start.sh

- Set up the JAIN SLEE TCK 1.0 Resource Adaptor
  % cd $*SLEE_HOME*/tck; ./bin/resource-setup.sh -i
  The `resource-setup.sh` (or `.bat`) script installs the JAIN SLEE TCK 1.0 Resource Adaptor type and the JAIN SLEE TCK 1.0 Resource Adaptor, creates and activates the resource adaptor entity, then binds the entity to link name 'slee/resources/tck'. (The JAIN SLEE 1.0 vendor should do the equivalent when installing their JAIN SLEE TCK 1.0 Resource Adaptor)

  Next steps:

  If you will be using JavaTest in GUI mode, the next step will be to invoke JavaTest using *TCK_HOME*`/bin/runtck.sh` (or `.bat`), to set the configuration options (see Chapter 5)

  If you will be using JavaTest in batch mode, and have configured sleetck.jti, the next will be to invoke JavaTest using *TCK_HOME*`/bin/runtck-batch.sh` (or `.bat`), to run the test suite (see Chapter 8)

4.3 JAIN SLEE TCK 1.0 Contents

The top most JAIN SLEE TCK 1.0 installation directory, is referred to as *TCK_DIRECTORY* throughout the JAIN SLEE TCK 1.0 documentation. You can name this directory whatever you want.

Once the JAIN SLEE TCK 1.0 is installed, several directories will be created under *TCK_DIRECTORY*/. The contents of these directories are as follows (on Win32 platforms assume backslashes in directory paths, instead of forward slashes used here). The JavaTest documentation is also

available in PDF format, for easy online viewing or for printing. This file is named `javatest.pdf` and it resides in the *TCK_DIRECTORY*`/doc/javatest/` directory

Table 4.1: JAIN SLEE TCK 1.0 directory contents

File or Directory	Contents
TCK_DIRECTORY	The top level directory of the installation contains copyright and license files, README files and release notes for the products contained in the distribution.
`bin/`	Contains scripts for the starting the JavaTest software, and scripts for JAIN SLEE TCK 1.0 specific tools
`config/`	Contains configuration files which should be edited as described in this guide.
`doc/`	This directory and its subdirectories contain documentation for JAIN SLEE TCK 1.0 and JavaTest. The files matching the pattern JAIN_SLEE_1.0_*_Assertions.* contain information about the JAIN SLEE 1.0 assertions covered by the JAIN SLEE TCK 1.0.
`doc/sleetck-user-guide.pdf`	Contains the JAIN SLEE TCK 1.0 User's Guide (this document)
`doc/javatest/`	Contains JavaTest-specific documentation, including `javatest.pdf`, the JavaTest User's Guide and Reference
`dtd/`	Contains the DTD files for JAIN SLEE 1.0
`jars/`	Contains the Jar libraries built by the JAIN SLEE TCK 1.0 build scripts

Table 4.1: (Continued)

File or Directory	Contents
`lib/`	Contains the JAR archive libraries used by the JAIN SLEE TCK 1.0
`lib/javatest.jar`	JAR archive for JavaTest software and JavaTest libraries. This file must be specified in the CLASSPATH shell environment variable in order to start the JavaTest software.
`src/sigtest`	Contains some of the source files from the Signature Test tool used to test binary compatibility with the JAIN SLEE 1.0 API
`src/tck/`	Contains the JAIN SLEE TCK 1.0 source files, deployment descriptors, and build files for JAIN SLEE components specific to tests. The directory structure of the `com/opencloud/sleetck/lib/testsuite` sub-directory matches the structure of the test descriptions directory (`testsuite/tests`).
`testsuite`	Contains the main testsuite file (`testsuite.jtt`, the current exclude file `jain-slee-tck-1_0.jtx`, and the JAIN SLEE TCK 1.0 test description DTD and template files)
`testsuite/testsuite.jtt`	File that contains information required by the JavaTest harness about the test suite.
`testsuite/tests/`	Contains all of the test descriptions (`.xml`) for the JAIN SLEE TCK 1.0. The test hierarchy begins with this directory (the JavaTest software refers to this directory as the RootURL for the JAIN SLEE TCK 1.0).

Table 4.1: (Continued)

File or Directory	Contents
`slee-ri-tck-resources.zip`	This file contains resources used for testing the JAIN SLEE 1.0 Reference Implementation. These resources can be used as a model for the equivalent resources which a JAIN SLEE 1.0 vendor will have to provide to enable testing using JAIN SLEE TCK 1.0
`sleetck-ant.xml`	This is an ANT build file for the JAIN SLEE 1.0 Technology Compatibility Kit
`tcktools-ant.xml`	Build file for the JAIN SLEE TCK extension classes, including the slee management tools

There may be additional directories related to the JavaTest harness that are not found under the *TCK_DIRECTORY*. These are listed in the table below.

Table 4.2: User-defined JAIN SLEE TCK 1.0 directory contents

File or Directory	Contents
user_defined_work_dir	Work directories are user definable and are used to store test result files (`.jtr`).
user_defined_report_dir	The report directory is user-definable and is used to store the `harness.trace` file and other report files.

Chapter 5

Starting and Configuring the JavaTest Harness

This chapter describes basic JAIN SLEE TCK 1.0 execution and configuration. It contains the following sections:

- Executing the JavaTest Harness Software 5.1
- JavaTest Harness Configuration 5.2
- JAIN SLEE TCK 1.0 Configuration Options 5.3

In order for the JAIN SLEE TCK 1.0 to run, the JavaTest harness variables have to be properly set. These are set through the configuration interview as described in "JavaTest Harness Configuration" on page 33.

5.1 Executing the JavaTest Harness Software

Before executing the JavaTest harness software, you must have a valid test suite and J2SE SDK 1.4.1 or greater installed on your system.

When executing the JavaTest harness, you can include arguments at the end of the command string that specify how it starts. These command-line options are described in your JavaTest documentation that is located at *TCK_DIRECTORY*`/doc/javatest/javatest.pdf`

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.

To execute the harness, run the Startup Script. You can use one of the startup scripts provided with the JAIN SLEE TCK 1.0 as described in "Executing the JAIN SLEE TCK 1.0 Script" on page 33.

When you execute the JavaTest harness for the first time it displays a Welcome to JavaTest dialog box.

- 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 Chapter 8, "Using the JAIN SLEE TCK 1.0."

5.1.1 Executing the JAIN SLEE TCK 1.0 Script

The JAIN SLEE TCK 1.0 provides scripts that can be used to execute the JavaTest harness on Linux, Solaris, and Windows platforms from the *TCK_DIRECTORY*. The scripts are named `runtck.sh` and `runtck.bat`, and are self-documented. They are found in the *TCK_DIRECTORY*`/bin` directory.

5.2 JavaTest Harness Configuration

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, you choose Run Tests > Start to begin a test run. At that point 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 a *configuration interview*. When you have entered the configuration data, you are asked if you wish to proceed with running the test.

5.2.1 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 JavaT-

est harness is running. Other configuration parameters can be set however you wish. For example, where you want test report files to be stored.

At any point in the interview you can use the File > Save As, or File > Save menu items to save your answers and your position in the interview. By default, the editor automatically saves the interview to a file named `default.jti`. It is possible to save the interview to a different file and create multiple configurations as described in the JavaTest online help.

5.3 JAIN SLEE TCK 1.0 Configuration Options

To configure the JAIN SLEE TCK 1.0, you can load the configuration from config/sleetck.jti, edit values based on guessed suggestions, or create it from scratch using the interview.

The options:

Load the default. This option assumes that you created a `sleetck.jti` file from the `sleetck.jti.template` file. After creating a work directory, choose Configure > Load Configuration... from the menu, the select the `sleetck.jti` file.

Edit from guessed values. To start using default values as guessed by the TCK, choose Configure > Edit Configuration from the menu.

Create a new configuration from scratch. To create a new configuration with no default values set, choose Configure > New Configuration from the menu

Additional notes for interview questions:

RMI Port The RMI port specified in the configuration interview must match that specified in the MLet configuration file `mlet.conf`, and RMI_PORT value set in the `start-rmiregistry` script.

The default, recommended value is 4099. If another value is chosen, the port number must be altered in all three places.

Chapter 6

Preparing the SLEE

PropertyPermission	‘‘*’’,‘‘read’’
RuntimePermission	‘‘queuePrintJob’’
SocketPermission	‘‘*’’,‘‘connect’’

Table 6.1: SLEE permissions expected by Sbbs

This chapter details how to prepare the JAIN SLEE 1.0 implementation for JAIN SLEE TCK 1.0 testing, and some non-obvious assumptions made by the of the JAIN SLEE 1.0 implementation by the JAIN SLEE TCK 1.0. It contains the following sections:

- Setting SLEE permissions 6.1
- Configuring the SLEE TCK MLet 6.2
- Installing the JAIN SLEE TCK 1.0 Resource Adaptor 6.3
- Requirements for the JAIN SLEE 1.0 Implementation 6.4

6.1 Setting SLEE permissions

The JAIN SLEE 1.0 implementor must set security permissions such that SBBs will have only the correct permissions. The relevent test is: tests/runtime/security/Security

tests/runtime/security/SecurityPermissionsTest.xml tests that the SBB is granted and denied security permissions as detailed in the Runtime Environment chapter of the SLEE specification.

As a summary, the SLEE should only grant the Sbb the permissions from table 6.1

6.2 Configuring the SLEE TCK MLet

The JAIN SLEE TCK 1.0 accesses the JAIN SLEE's management interface via a plug-in which must be installed by the vendor.

The MLet must be deployed in the MBeanServer which hosts the SLEE management MBeans, and must be reachable via RMI from the test engine.

Installation will usually be a case of adding an entry to an MLet configuration file which references and configures the plug-in. The `mlet.conf` file which is contained in the `slee-ri-tck-resources.zip` file provides the model for the configuration requirements, and contains the default configuration values expected by the TCK.

The `mlet.conf` file must reference the correct JAIN SLEE TCK 1.0 archive and RMI port number.

The RMI port must be set to match the RMI port used by the RMI registry used by JAIN SLEE TCK 1.0 components.

6.2.1 SLEE TCK MLet

6.3 Installing the JAIN SLEE TCK 1.0 Resource Adaptor

The JAIN SLEE TCK 1.0 Resource itself is NOT distributable - ie it is single VM. It must be reachable on single host via RMI. Distributed JAIN SLEE 1.0 implementations may be tested using a distributed resource adaptor, which connects to a standalone JAIN SLEE TCK 1.0 resource.

All resource adaptors must deployed on the same host, and must share the same rmi registry (ie be bound to the same port). but they can reside in different jvms.

The JAIN SLEE TCK 1.0 resource adaptor must be deployed in the JAIN SLEE 1.0 implementation under test. As the resource management API is not specified in the JAIN SLEE Specification 1.0, the installation steps will differ across implementations. The JAIN SLEE TCK 1.0 Resource installation is expected to involve the following steps:

1. Install the JAIN SLEE TCK 1.0 Resource Adaptor Type
2. Install the JAIN SLEE TCK 1.0 Resource Adaptor
3. Create the JAIN SLEE TCK 1.0 Resource Entity
4. Activate the JAIN SLEE TCK 1.0 Resource Entity
5. Bind the JAIN SLEE TCK 1.0 Resource Entity to the following name:

   ```
   slee/resources/tck
   ```

6.4 Requirements for the JAIN SLEE 1.0 Implementation

This section details requirements for the JAIN SLEE 1.0 implementation which are not covered in either the JAIN SLEE 1.0 specification or other sections in this user guide. Note that extra preparation of the SLEE may be required to meet any unsatisfied requirements.

The JAIN SLEE TCK 1.0 requires the following:

The JAIN SLEE TCK 1.0 tests assume that the JAIN SLEE TCK 1.0 resource will be active and have it's event handler set, be bound in the RMI registry, and be available to services via jndi throughout the test cycle, whenever the SleeProvider is in the started state.

- The JAIN SLEE TCK 1.0 requires that the vendor installs no components with vendor name "`jain.slee.tck*`" other than those introduced by the JAIN SLEE TCK 1.0.
- The JAIN SLEE TCK 1.0 requires that the vendor creates no standard address profile tables or resource info tables with names beginning with "`tck`".
- SBBs may access the JAIN SLEE TCK 1.0 resource adaptor from any state (including the pooled state).
- The SLEE must be able to be stopped and started during the test cycle via the MBean interface.

Chapter 7

Verifying the JAIN SLEE TCK 1.0

This chapter describes how to start the JAIN SLEE TCK 1.0 and verify that it is properly configured. It contains the following sections:

- JAIN SLEE TCK 1.0 Operating Assumptions 7.1
- Verifying Installation and Setup 7.2

7.1 JAIN SLEE TCK 1.0 Operating Assumptions

- J2SE SDK version 1.4.1 or later is installed on the system hosting the JavaTest harness.
- You are using JavaTest harness version 3.x.
- An implementation of JAIN SLEE 1.0 is installed on the system hosting the JavaTest harness or on a system that the JavaTest host can access.

7.2 Verifying Installation and Setup

Once the JAIN SLEE TCK 1.0 has been installed, you should set up and verify test configurations in incremental steps in order to simplify any necessary troubleshooting. It is a good idea to *first* set up the JavaTest harness and pass a small test to make sure that you have the correct JavaTest harness configuration and that the JavaTest harness is running correctly.

7.2.1 Verifying JavaTest Harness Configuration

1. **Start the SLEE**
2. **Execute the JavaTest harness software**
3. **Select simple test to run**
   Use the JavaTest harness configuration editor to choose a small, simple test to run. For example, `tests/preparatory/sleestate/SleeStateTest.xml` or `tests/events/handlerinvocation/EventHandlerInvocationTest.xml`. Consult the JavaTest online help for information on how to choose tests.
4. **Specify Directories**
   Use the JavaTest harness configuration editor to chose the report and work directories you wish to use.
5. **Start the JavaTest harness**
   Choose Run Tests > Start to begin the test. If configuration information is incomplete, you will be asked to supply the missing data.

The JavaTest status bar grows while JavaTest tracks statistics relative to the files done, tests found, and tests done.
Should you encounter any errors after clicking Start, refer to Part IV, Troubleshooting, in the *JavaTest User's Guide and Reference* (located at *TCK_DIRECTORY*`/doc/javatest/javatest.pdf` in the JAIN SLEE TCK 1.0 distribution).

6. **Check the results**
   Check the test results as displayed by the JavaTest harness to make sure that everything is functioning correctly. Consult the JavaTest online help for information on test results.

Chapter 8

Using the JAIN SLEE TCK 1.0

This chapter describes how to use the JAIN SLEE TCK 1.0 to test your implementation. It contains the following sections:

- Using the JAIN SLEE TCK 1.0 to Test a JAIN SLEE 1.0 Implementation 8.1
- Requirements and Constraints 8.2
- Monitoring Test Results 8.3
- Test Reports 8.4

This chapter assumes that you have verified that your TCK is properly configured as described in Chapter 7, "Verifying the JAIN SLEE TCK 1.0".

8.1 Using the JAIN SLEE TCK 1.0 to Test a JAIN SLEE 1.0 Implementation

As a brief summary:

- Start the RMI registry
- Start the SLEE
- Start JavaTest using `./bin/runtck.sh`
- Use the GUI to start a test run

To perform full testing of the JAIN SLEE 1.0 implementation on the target implementation, follow the steps listed below.

1. **Start the RMI registry** Run an RMI registry on a user chosen port with the required classes in the classpath. The script `./bin/start-rmiregistry.sh` (or .bat) provides the easiest way of configuring the RMI registry.
2. **Start the SLEE** Start the JAIN SLEE 1.0 implementation to be tested.
3. **Execute the JavaTest harness software.**

   Described in "Executing the JavaTest Harness Software" on page 32.
4. **Select the tests to be run.**

   Use the JavaTest harness configuration editor to choose which tests you wish to run.

   Consult the JavaTest online help for information on how to choose tests.

5. **(Optional) Specify directories.**

   Use the JavaTest harness configuration editor to chose the report and work directories you wish to use.

6. **Start the test run.**

   Choose Run Tests > Start to begin the test run. If configuration information is incomplete, you are asked to supply the missing data.

   The JavaTest status bar grows while JavaTest tracks statistics relative to the files done, tests found, and tests done.

7. **Check the results.**

   Test progress and results are displayed by the JavaTest harness.

8.2 Requirements and Constraints

- **Don't install any SLEE components (e.g. SLEE services) during a test suite run.**
- **Don't run more than one test at a time.**
  JavaTest prompts for a "Concurrency" value (the maximum number of tests that should be run concurrently). Choose the default value of "1". *The JAIN SLEE TCK 1.0 does not support concurrent execution of tests.*

8.3 Monitoring Test Results

After the test run begins, you can track its progress using the test progress display fields, Progress Monitor dialog box, test tree, and information tabbed panes in the Test Manager window.

8.4 Test Reports

A set of report files is created for every test run.
Test result reports include the following kinds of information:

- Environment information—tester login, working directories, and other identification information
- Date and time for each event
- Copies of configuration files
- List of tests or symbolic names that can be expanded to the exact list of the tests

- The list of excluded tests as specified by the Exclude List.
- A pass/fail report, including any return status, return values, or return message
- Any system messages, including exceptions and errors, generated by the tests.

There are a number of files generated in the report directory.

Within that directory, the JavaTest harness creates a test result hierarchy similar to the test suite hierarchy that contains your tests. Result files for the tests in your test run appear under the appropriate subdirectory within this result hierarchy.

After the test run is completed, the JavaTest harness writes HTML reports for the test run. You can open these files in a web browser to view or print their contents.

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.

8.4.1 Generating Reports Using the Script

Test reports may also be generated using the batch scripts provided with the TCK (`./bin/runtck-batch.sh and ./bin/runtck-batch.bat`).

Test reports are generated by default following a test run, for the chosen tests.

To generate a set of reports for a previous test run, use the `-nt` command line option.

E.g. the following would create a set of reports in `myReportDir` for the specified configuration: `./bin/runtck-batch.sh -nt -c config/sleetck.jti -d myWorkDir -r myReportDir -excludeList testsuite/jain-slee-tck-1_0.jtx`

See the documentation in the script for configuration options.

Chapter 9

Testing API Signatures

This chapter describes how to run a diagnostic test to compare the signatures of the public and protected methods, constructors, and fields in the jar file containing classes for the RI against an exact list of signatures from the API Specification. It contains the following sections:

- Overview 9.1
- Classes required by the API Signature Tests 9.2
- Setting the `CLASSPATH` for API Signature Testing 9.3
- Running Signature Test 9.4

Running the Static Signature test is required.

9.1 Overview

It is physically impossible to verify API signatures using the JAIN SLEE 1.0 implementation in the absence of reflection capabilities. However, API libraries typically have prototypes that follow the standard class file format. In this case it is possible to explore the class files of the prototype to gain confidence that the API libraries that are actually present on the target platform comply with the specification. The JAIN SLEE TCK 1.0 contains a tool called Static Signature test exactly for this purpose, and its use is covered in this chapter.

9.2 Classes required by the API Signature Tests

The JAIN SLEE 1.0 specification requires the following APIs to be made available to SBBs:

- Java 2 Platform, Standard Edition, v1.3 (J2SE) APIs
- JAIN SLEE 1.0 APIs
- JNDI 1.2 Standard Extension
- JAXP 1.0
- JDBC 2.0 Standard Extension (support for row sets only)

Implementation classes for the above APIs must be accessible from the `CLASSPATH` while running the API signature tests, as detailed in the following section.

9.3 Setting the CLASSPATH for API Signature Testing

Signature Test checks binary compatibility of an implementation by testing provided classes against a signature file, using Java™reflection capabilities. To do this, the relevant classes must be accessible from the `CLASSPATH`.

The `CLASSPATH` variable can be modified to include the implementation classes in one of two ways:

- **Add the required libraries to the VENDOR_LIB path (recommended)**
  If your JAIN SLEE TCK 1.0 installation been set up using a `setup-tck` script, you can add the required libraries to the VENDOR_LIB variable by editing the *TCK_DIRECTORY*`/config/config_variables` file.
  Append your libraries to the "`VENDOR_LIB=slee-implementation.jar`" line.

- **Edit the `runtck` script in the `bin` directory for your platform.**
  The `CLASSPATH` is set in this script, on a line of the form "`CLASSPATH=jar1.jar:jar2.jar:...`". You can append the required libraries to the end of this line to include them in the `CLASSPATH`.
  Paths are relative to *TCK_DIRECTORY*.
  E.g. if you are running on Linux or Solaris and want to add `implementation-classes.jar` to the `CLASSPATH`, edit *TCK_DIRECTORY*`/bin/runtck.sh`

9.4 Running Signature Test

The signature tests included in the JAIN SLEE TCK 1.0 test suite delegate to the Signature Test tool—no direct interaction with Signature Test is required. Once the `CLASSPATH` is set as detailed above, the tests can be run in the normal way.

The API signature tests are located under the *TCK_DIRECTORY*`/tests/api/signaturetest` directory.

Chapter 10

Test-Specific Information

This chapter provides information required to configure, set up, and run specific JAIN SLEE TCK 1.0 tests.

10.1 SleeSignatureTest

10.1.1 Setup

The default API signature (slee-1_0.sig) was created using the JMX 1.0 API, but a variant was created using JMX 1.1 (slee-1_0-with-jmx1_1.sig) which is supported by this test. If your implementation uses JMX 1.1:

1. In the lib directory under the JAIN SLEE TCK 1.0 base directory, replace `jmxri.jar` with `jmxri1.1.jar`

2. In the test description file (`testsuite/tests/api/signaturetest/SleeSignatureTest.xml`), use the supplied variant of the `executeClass` element which references `slee-1_0-with-jmx1_1.sig` (see the notes in the test description file).

10.2 Security Permissions Test

10.2.1 Setup

Security permissions granted to the SBBs must match those specified in the Runtime Environment chapter of the JAIN SLEE 1.0 specification. See "Setting SLEE permissions" 6.1.

10.3 javax/slee/management/DeployableUnitDescriptor/Test37

Test3742Test tests the DeployableUnitDescriptor.getDeploymentDate() method, by recording the time before and after the install() call, and checking that the date returned by DeployableUnitDescriptor.getDeploymentDate() is between these times. If the SLEE and test engine are running on different machines, their clocks may beed to be synchronized to pass this test.

The test description file is located at javax/slee/management/DeployableUnitDescriptor/Test3742Test.xml

Chapter 11

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. It contains the following sections:

11.1 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 window provides you with a number of tools for effectively troubleshooting a test run.

Consult *JavaTest User's Guide* and JavaTest online help for detailed descriptions of the tools described in this chapter.

11.2 Test Tree

Use the test tree 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)

11.3 Folder Information

Click a folder in the test tree 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.

11.4 Test Information

To display information about a test, 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.

11.5 Assertion Details

If the test has returned a Fail result, the test result message will include an assertion ID which identifies the assertion from the JAIN SLEE 1.0 specification that was not satisfied. To find details about the assertion (including the text of the assertion itself), use the assertion spreadsheets which are provided in the `doc` directory of the JAIN SLEE TCK 1.0 specification (as JAIN_SLEE_1.0_Assertions.sxc, JAIN_SLEE_1.0_Assertions.xls, and JAIN_SLEE_1.0_TechSpec_Assertions.csv/JAIN_SLEE_1.0_JavaDoc_Assertions.csv).

Note: Assertion -1 is used for the preparatory tests which are not related to any particular assertion in the JAIN SLEE 1.0 specification.

11.6 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.

There are different `.html` format report files that you can view manually following a test run. When you began your test run, you entered a value in the "where to put results" screen. All these files may be found in that designated report directory. By default, the report directory is *workdir*`/report`.

Another important resource for debugging test failures is the work directory report called `failed.html`.

11.7 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).

11.8 Failures due to a Timeout

If a test failed due a timeout, simply due to the SLEE being slow to respond, the user may increase the timeout in the 'Default time out' question in the configuration interview (see section 5.2).

11.9 Test Source Code

If the cause of the failure or error is not apparent from the test messages or reports, it may be necessary to view the test source code to find the cause of the error.

The source files may be located using the following steps:

1. The test description file for the failing test will be stated in the GUI or the test report
2. The class name of the main test class is referenced in the `executeClass` element of the test description file
3. If the test will install components into the SLEE, this test description file will usually reference the location of the deployable-unit jar to install. To find the sources uses in the component, search for a matching deployable-unit jar reference in a `custom-components.xml` file in the

testsuite, which will indicate the deployment descriptors used by the component

4. The deployment descriptors will reference the main classes used by the component

5. The remaining classes can be found by following the references in the source files

The JAIN SLEE TCK 1.0 source files are located at `src/tck` under the base installation directory.

The test descriptions files are located at `testsuite/tests` under the base installation directory.

The test suite source directory is located at `src/tck/com/opencloud/sleetck/lib/testsuite` under the base installation directory. It contains source files for test classes, and test specific deployment descriptors.

The build files for the JAIN SLEE components used by the tests are named `custom-components.xml`, and are placed in the sub-directories of the test suite source directory.

11.9.1 Suspected Test Bug

If the cause of the failure or error is a fault appears to be in the test, and the test is considered to be invalid, then the JAIN SLEE TCK 1.0 licensee may use the appeals process to challenge the test, as detailed in section 2.4.

Appendix A

Frequently Asked Questions

A.1 JavaTest Harness

Q: Is there a JavaTest harness User's Guide available?
A: On line help is available for the JavaTest harness. You can select the "?" icon or Help->User's Guide from the JavaTest harness menu. There is also a PDF format version of the on line help at *TCK_DIRECTORY*`/doc/javatest/javatest.pdf` that can be viewed on-screen or printed out.

Q: How do you move a set of reports to a new location without breaking the links?
A: The JavaTest harness 3.x provides a utility program to reconnect the links after moving the reports.

Q: What factors affect how fast JavaTest runs?
There are many factors, but the key ones are:

- Size of test suite.
- GUI mode vs. batch mode. The GUI has more overhead, so it will be slower.
- Size of work directory—empty or full.
- Using, or not using, a binary test finder (see the top of the `harness.trace` file).

A.2 JavaTest Harness Configuration

Q: Since `.jti` files are checksummed, how can I use build scripts to dynamically configure configuration settings for (automated) batch runs?
A: There are two ways to do this. The `editJTI` utility allows automated "offline" modification of the interview file. The batch command allows you to modify the interview answers when invoking the harness in batch mode.

Q: How can we avoid having to re-answer all the interview questions every day?
A: You can save the interview in a `.jti` file and then reload that file when doing the interview for the next build. **Q: Multiple people use similar settings for running the TCK. Do we all need to answer the interview individually and keep track of our own `.jti` files?**
You can setup a central repository of `.jti` files, from which people can "prime" their interview. (The files in this repository should be made read-only so that each user does not alter the master copy.) Each user can load the shared master copy, make alterations for their particular task and then save the interview in a new file. This saves everyone from having to answer all the common questions.

A.3 Testing an Implementation

Q: Where do I start to debug a test failure?
A: From the JavaTest GUI, you can view recently run tests using the Test Results Summary (click the fourth icon), by selecting the red Failed tab or the blue Error tab. You can also take a look at the raw *testname*`.jtr` file located in the Work Directory. The `.jtr` file lists all the environment variables and echoes the evaluated command strings. Taking these evaluated strings and running them outside of JavaTest harness, on the command line, can often help to debug problems in your environment. See Chapter 11, "Debugging Test Problems," for more information.

Q: How do I restart a crashed test run?
A: 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.

Q: What results should I expect when I execute the tests within the TCK?
A: When the JAIN SLEE TCK 1.0 tests are run with the exclude file (`.jtx`), all tests should pass. This means that from the JavaTest GUI, the Test Results Summary window will show no tests listed on the Failed tab. Remember, to examine all test results, particularly if you have run the JAIN SLEE TCK 1.0 in separate pieces, you will need to run a final report so that all result files (`.jtr`) are examined.

Q: Why are there so many tests in the Exclude List?
A: The JavaTest exclude file (`*.jtx`) contains all tests that are not required to be run. The following is a list of reasons for a test to be included in the Exclude List:

- An error in the JAIN SLEE 1.0 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.