SIP Servlet v1.1 TCK Technology Compatibility Kit

User's Guide

Preface

This book introduces the Technology Compatibility Kit for the SIP Servlet v1.1, and explains how to configure and run the TCK test suite. It also provides information for troubleshooting any problems encountered with the test execution.

License

TCK LICENSE AGREEMENT

READ THE TERMS OF THIS (THE "AGREEMENT") CAREFULLY BEFORE VIEWING OR USING THE TECHNOLOGY COMPATIBILITY KIT LICENSED HEREUNDER. BY VIEWING OR USING THE TECHNOLOGY COMPATIBILITY KIT, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE TECHNOLOGY COMPATIBILITY KIT ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN THE UNUSED TECHNOLOGY COMPATIBILITY KIT TO BEA SYSTEMS, INC. (“BEA”).

1.0 DEFINITIONS

1.1 *"*FCS*"* means first commercial shipment of a product.

1.2 "Intellectual Property Rights" means worldwide rights arising under contract, statute or common law, whether or not perfected, and associated with: (a) patents and patent applications; (b) works of authorship, including copyrights, mask works, and moral rights; (c) the protection of trade and industrial secrets and confidential information; (d) any rights analogous to those set forth herein and any other proprietary rights relating to intangible or intellectual property now existing or later recognized in any jurisdiction (excluding trademarks, service marks, trade names, and trade dress); and (e) divisions, continuations, renewals, reissuances, reexaminations, applications, registrations, and any extensions of the foregoing (as applicable), now existing or hereafter filed, issued or acquired.

1.3 “JCP Specification” means the written specification for SIP Servlet Specification v1.1 (JSR-289), Java technology developed pursuant to the Java Community Process.

1.4 “Reference Implementation” means the prototype or “proof of concept” implementation of the JCP Specification developed and made available for license by or on behalf of BEA.

1.5 “Specification License” means the license offered by BEA under certain of its intellectual property rights to create an independent implementation of the JCP Specification.

1.6 *"*Technology Compatibility Kit*" or "*TCK*"* means the documentation, testing tools and test suites associated with the JCP Specification as may be revised by BEA from time to time, that is provided so that an implementer of the JCP Specification may determine if its implementation is compliant with the Specification.

2.0 LICENSE GRANTS

2.1 **License Grant for the TCK.**

(a) Limited Grant. Subject to Your compliance with the restrictions and obligations contained in this Agreement, including but not limited the Additional Limitations set forth in subsection (b) below, BEA hereby grants You, to the extent of BEA's Intellectual Property Rights in the TCK(s), a worldwide, non-exclusive, limited license, with right to sublicense, to use the TCK(s) internally and solely for the purpose of developing and testing Your Products. No license is granted for any other purpose, including any of the activities described in Section 2.1(b).

(b) Additional Limitations. You may not: (i) create derivative works of the TCK(s) or (ii) disassemble or decompile binary portions of the test suite(s) or testing tools or otherwise attempt to derive the source code from such portions or (iii) use any marks, brands or logos associated with the JCP Specification, or otherwise promote or market any product, or code, which implements any portion of the JCP Specification, as being compatible, compliant, conformant or otherwise consistent with the JCP Specification unless such product passes, in accordance with the documentation (including the TCK Users Guide, if any), the most current TCK applicable to the latest version of the JCP Specification and available from BEA one hundred twenty (120) days before FCS of such version of the product; provided, however, that if You elect to use a version of the TCK also provided by BEA that is newer than that which is required under this Section 2.1(b)(iii), then You agree to pass such TCK. Without limiting the generality of the foregoing, you may not modify any portions of the TCK that are provided in source code format.

(c) Testing. You shall self-certify that Your product implementing the JCP Specification passes the applicable TCK as set forth above, if and when such product in fact does so, provided that if BEA reasonably believes that Your product does not pass the applicable TCK, BEA may require You to submit specific test documentation to an independent third party audit facility designated by BEA, for verification of proper compatibility testing.. If the audit determines that such Product does not pass the TCK, (i) You shall be responsible for all reasonable costs of such audit including but not limited to the costs of retesting, and (ii) You shall within 90 days of receipt of notice that the Product does not pass the TCK modify the Product so that it does pass the TCK and resubmit it to the third party auditor for retesting. If you do not comply with subsection (ii) of this Section (c) you shall immediately cease use of any marks, brands or logos associated with the JCP Specification, and shall otherwise cease to promote or market any product, or code, which implements any portion of the JCP Specification, as being compatible, compliant, conformant or otherwise consistent with the JCP Specification.

2.2 Proprietary Rights Notices. You shall not remove any copyright notices, trademark notices or other proprietary legends of BEA or its suppliers contained on or in the TCK, and shall incorporate such notices in all copies of any TCK. You shall comply with all reasonable requests by BEA to include additional copyright or other proprietary rights notices of BEA or third parties from time to time.

2.3 Ownership. You acknowledge and agree that, as between BEA and You, nothing in this Agreement transfers any right, title and interest in and to the TCK, any derivative works

thereof and Intellectual Property Rights (excluding any pre-existing Intellectual Property Rights owned by You) associated therewith, to You.

2.4 No Other Grant. This Agreement does not grant to You any right or license, under any Intellectual Property Rights of BEA or otherwise, except as expressly provided in this Section 2, and no other right or license is to be implied by or inferred from any provision of this Agreement or by the conduct of the parties.

3.0 SUPPORT AND UPGRADES

Nothing in this Agreement shall obligate BEA to provide any technical support, updates, newer versions of or other assistance concerning the TCK to You or to any of Your distributors or customers for Your products. BEA, it its sole option, may elect to make technical support for the TCK available pursuant a separate agreement.

4.0 LIMITED WARRANTY AND DISCLAIMER

THE TCK IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE TCK IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE TCK IS WITH YOU. SHOULD THE TCK PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT BEA OR ANY OF ITS CONTRIBUTORS OR SUPPLIERS) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF THE TCK IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

5.0 LIMITATION OF LIABILITY

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL BEA, OR ANY OF ITS CONTRIBUTORS OR SUPPLIERS BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

6.0 TERM AND TERMINATION

6.1 Term. The Term of this Agreement shall begin on the Effective Date and shall continue unless terminated as provided below. Termination is permitted: (a) by either party for the other party's breach of this Agreement, upon written notice to the other party providing a reasonable opportunity to cure given the nature of the breach, in no event to be less than thirty (30) days; or (b) by BEA upon any action by You alleging that use or distribution of the TCK or an implementation of the JCP Specification by BEA or any of BEA's licensees of the TCK infringes a patent owned by You.

7.0 MISCELLANEOUS

This Agreement represents the complete agreement concerning subject matter hereof. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this Agreement shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this Agreement.

Overview

This guide describes how to install, configure, and run the Technology Compatibility Kit (TCK) that is used to test an implementation of the SIP Servlet v1.1 (JSR 289) specification. The SIP Servlet v1.1 TCK is designed as a portable, configurable automated test execution for verifying the compliance of a licensee's implementation of the SIP Servlet v1.1 Specification.

Who Should Use This Book

This book should be read by anyone who will use the SIP Servlet v1.1 TCK.

Before You Read This Book

The SIP Servlet v1.1 TCK is based on the SIP Servlet v1.1 Specification.
Links to the specification and other product information can be found on the Web at: http://jcp.org/en/jsr/detail?id=289
Before using the SIP Servlet v1.1 TCK, it is recommended that you read and become familiar with the SIP Servlet v1.1 Specification.

How This Book Is Organized

This book contains the following chapters:

- Chapter 1, "Introduction" gives an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and describes the SIP Servlet v1.1 TCK. It also includes a listing of the basic steps needed to get the SIP Servlet v1.1 TCK up and running.
- Chapter 2, "Procedure for SIP Servlet v1.1 Certification" describes the conformance testing procedure and testing requirements.
- Chapter 3, "Configuration and setup" explains how to configure your test environment, including any special setup instructions that need to be completed prior to deploying and/or running selected tests.
- Chapter 4, "Preparing the SIP Servlet v1.1 Implementation" details how to prepare the SIP Servlet v1.1 implementation for TCK testing.
- Chapter 5, "Verifying SIP Servlet v1.1 TCK" describes how to verify that SIP Servlet v1.1 TCK is properly configured.
- Chapter 6, "Testing Your Implementation" describes how to use the SIP Servlet v1.1 TCK to test your implementation.
- Chapter 7, "Troubleshooting" provides some approaches for dealing with these failures.

Related Documents

- The SIP Servlet v1.1 specification (JSR 289)
- The Java Programming Language

Accessing Documentation

When unzipped and installed, the SIP Servlet v1.1 TCK (`sipservlet-1_1-tck.zip`) contains two directories – `tck` and `sipunit`. The `tck` dir is referred to as *TCK_DIRECTORY* and `sipunit` dir is referred to as *SIPUNIT_DIRECTORY* throughout the SIP Servlet v1.1 TCK documentation. The *TCK_DIRECTORY* includes a `docs/` directory that contains this manual in Adobe Acrobat™ PDF format i.e. this SIP Servlet v1.1 TCK User's Guide is located at: TCK_DIRECTORY/docs/ SIP_Servlet_v1.1_TCK_User_Guide.pdf

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`*.

1 Introduction

This chapter gives an overview of the principles that apply generally to all Technology Compatibility Kits (TCKs) and describes the SIP Servlet v1.1 TCK. It also includes a listing of what is needed to run with the SIP Servlet v1.1 TCK.
It contains the following sections:

- *Compatibility Testing*
- *Java Community Process (JCP) Program and Compatibility Testing*
- *The SIP Servlet v1.1 TCK*
- *The SIP Servlet v1.1 TCK-Configuration*
- *Getting Started*

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 by which Sun Microsystems, Inc. ensures that the Java platform does not become fragmented as it is ported to different operating systems and hardware environments.
- Compatibility testing helps ensure that any Java platform implementation you develop will work seamlessly with any other Java platform implementation.
- Compatibility testing enables Java application developers 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 implementers 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 SIP Servlet v1.1 TCK Certification”.

1.1.3 TCK Overview

A TCK is a set of tools and tests used to verify that a licensee’s implementation of SIP Servlet v1.1 technology conforms to the applicable specification. All tests in the TCK are based on the written specifications for the Java platform. This TCK tests compatibility of a licensee’s implementation of SIP Servlet v1.1 technology to the applicable specification of the technology. Compatibility testing is a means of ensuring correctness, completeness, and consistency across all implementations developed by SIP Servlet v1.1 technology licensees.
The set of tests included with each TCK is called the “*test suite*.” Most tests in the TCK 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 passes, 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™ (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 on the JCP program see this URL: http://jcp.org

1.3 The SIP Servlet v1.1 TCK

The SIP Servlet v1.1 TCK is designed as a portable, configurable, automated test suite for verifying the compliance of a licensee's implementation of the SIP Servlet v1.1 Specification. The SIP Servlet v1.1 specification can be found at: http://jcp.org/en/jsr/detail?id=289. The SIP Servlet v1.1 TCK uses the extended SipUnit by BEA Systems, Inc. which is based on SipUnit 0.0.6b and can be found at: http://www.cafesip.org/projects/sipunit/index.html.

1.3.1 SIP Servlet v1.1 TCK Specifications and Requirements

This section lists the applicable requirements and specifications.

- **SIP Servlet Version**. The SIP Servlet v1.1 TCK is based on the SIP Servlet v1.1 Specification.
- **Specification Requirements**. Requirements for a SIP Servlet v1.1 implementation are described in detail in the SIP Servlet v1.1 Specification. Links to the SIP Servlet v1.1 specification and other product information can be found at http://jcp.org/en/jsr/detail?id=289.
- **SipUnit**. The SIP Servlet v1.1 TCK uses extended SipUnit, which is required version 0.0.6b and can be found at: http://www.cafesip.org/projects/sipunit/index.html.
- **Reference Implementation**. The designated Reference Implementation for conformance testing of implementations is based upon SIP Servlet v1.1 Specification.
- **Platform requirements**. The following requirements must be met in order to run the SIP Servlet v1.1 TCK on the host system:
  – **Operating system**. You may use any hardware platform that supports Java SE version 5.0. The SIP Servlet v1.1 TCK has been tested on Linux, Windows NT and Windows XP.
  – **Disk space**. At least 120 Megabytes of free disk space is needed for installation of the SIP Servlet v1.1 TCK, temporary files, and for the creation of report files.
  – **Memory**. At least 64 Mb of RAM is recommended for running the SIP Servlet v1.1 TCK on Windows platforms.

Licensees that need to build the SIP Servlet v1.1 TCK will need Ant 1.6.5 or higher. (Available from http://jakarta.apache.org/ant/)

1.3.2 SIP Servlet v1.1 TCK Components

This section describes the main components that make up the SIP Servlet v1.1 TCK.

SipUnit

SipUnit provides a test environment geared toward unit testing SIP applications. It extends the JUnit test framework to incorporate SIP-specific assertions, and it provides a high-level API for performing the SIP operations needed to interact with or invoke a test target. BEA

Systems, Inc. extended SipUnit based on version 0.0.6b to support SIP Servlet v1.1 features.

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 SIP Servlet v1.1 are described in Chapter 2, “Procedure for SIP Servlet v1.1 Certification.”

Exclude List

Note - You should regularly check the SIP Servlet v1.1 TCK web site for updates to the Exclude List.

Each version of a TCK includes Exclude List functionality. This identifies test method URLs that do not have to be run for the specific version of the TCK being used. Whenever tests are run, the SIP Servlet v1.1 TCK 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 SIP Servlet v1.1 TCK is located in the TCK DIRECTORY/conf/ directory.

Note - From time to time, updates to the Exclude List are made available on the SIP Servlet v1.1 TCK web site. Always use an up-to-date copy of the Exclude List.

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
- Optional APIs are not mandatory, so they can be listed in Exclude List for ensuring the TCK pass rate.

1.4 SIP Servlet v1.1 TCK Configuration

Configuring SIP Servlet v1.1 TCK test runs is described in Chapter 5, “Starting and Configuring the JavaTest Harness.”

1.5 Getting Started

This section provides a general overview of how to install, set up, test, and use the SIP Servlet v1.1 TCK:

1. Make sure that the following software has been correctly installed on the system host:
   - Java software. Java technology Java SE version 5.0 or later is required.

2. Install the SIP Servlet v1.1 TCK on the system host to run TCK.

See Chapter 3, “Configuration and Setup” for more information.

3. Install the implementation of SIP Servlet v1.1 to be tested, and deploy TCK applications.
4. Test the full SIP Servlet v1.1 implementation by running all tests.

See Chapter 6 “Testing Your Implementation”.

2 Procedure for SIP Servlet v1.1 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 describes the compatibility rules and defines the terms used in those rules.

2.2.1 Definitions

These definitions are for use only with the 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™ (JCP™), for the Technology Under Test.

The Libraries for SIP Servlet 1.1 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. BEA Systems, Inc. is the Maintenance Lead for SIP Servlet 1.1.

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 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

The 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 SIP Servlet 1.1.

Test Result

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

Success Rate

A “Success Rate” result indicates the ratio of how many test cases are passed: the compliance percent that SIP Servlet 1.1 implementation is compliant with the SIP Servlet 1.1 specification in the tested area.

Failures

A “Failure” result indicates that the number of the test cases which are failed: the failed cases number that SIP Servlet 1.1 implementation is noncompliant with the SIP Servlet 1.1 specification in the tested area.

Errors

An “Errors” result indicates that the number of test cases which have error to continue the test cases execution: the test cases that need to be debugged.

Test Suite

The requirements, tests, and testing tools distributed by the Maintenance Lead as applicable for a given version of the Technology.

Version

A release of the Technology, as produced through the JCP.

2.2.2 Rules for SIP Servlet 1.1 Products

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

SIPServlet1

The Product must be able to satisfy all applicable compatibility requirements, including passing all Compatibility 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.

SIPServlet1.1

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.

SIPServlet1.2

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.

SIPServlet1.3

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.

SIPServlet2

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 SIP Servlet v1.1 web site and apply to all licensees.

SIPServlet3

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

SIPServlet4

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

SIPServlet5

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

SIPServlet6

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.

SIPServlet7

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.

SIPServlet7.1

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.

SIPServlet8

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.

SIPServlet9

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

2.3 SIP Servlet 1.1 Test Appeals Process

The Maintenance Lead will be the point of contact for all test challenges to the Test Suite for the SIP Servlet v1.1.

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 SIP Servlet v1.1 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:

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 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 “SIP Servlet v1.1 TCK Test Appeals Steps”.

All tests found to be invalid will either be placed on the Exclude List for that version of the SIP Servlet v1.1 TCK or have an alternate test made available as follows:

1. 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 SIP Servlet v1.1 TCK licensees on the SIP Servlet v1.1 TCK web site.
2. 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 SIP Servlet v1.1 TCK website.

2.3.1 SIP Servlet v1.1 TCK Test Appeals Steps

1. SIP Servlet v1.1 licensee writes a test challenge to the Maintenance Lead contesting the validity of one or a related set of SIP Servlet v1.1 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 are 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 SIP Servlet v1.1 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 on the SIP Servlet v1.1 TCK 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?)	

2.4 Specifications for SIP Servlet v1.1

The Specification for SIP Servlet v1.1 is found on the JCP web site at:
http://jcp.org/en/jsr/detail?id=289

2.5 Libraries for SIP Servlet v1.1

- javax.servlet.sip
- javax.servlet.sip.annotation
- javax.servlet.sip.ar
- javax.servlet.sip.ar.spi

The above libraries are all included in the SIP Servlet v1.1 API classes jar, which is available as part of the specification download at the JCP web site.

3 Configuration and Setup

This chapter describes SIP Servlet v1.1 TCK configuration and setup. It contains the following sections:

- Obtaining the TCK Software
- Installing the SIP Servlet v1.1 TCK Software
- SIP Servlet v1.1 TCK Contents
- Configuring SIP Servlet v1.1 TCK
- Setting the Test Environment

3.1 Obtaining the TCK Software

The SIP Servlet v1.1 TCK software is available via web download at http://jcp.org/en/jsr/detail?id=289. The installation instructions are included in this *User Guide*.

3.2 Installing the SIP Servlet v1.1 TCK Software

Complete the following procedure to install the SIP Servlet v1.1 TCK on a system running Linux or Windows operating system.

1. Copy or download the SIP Servlet v1.1 TCK software from the web site, http://jcp.org/en/jsr/detail?id=289.
2. Change to the directory in which you want to install the SIP Servlet v1.1 TCK software and use the unzip command to extract the bundle:

```
cd <install_directory>
unzip sipservlet-1_1-tck.zip
```

This creates the `tck` directory and `sipunit` directory.

3. Check your current ANT_HOME variable and the Ant version is required with 1.6.5 or later.
4. After you complete the installation, follow the directions in Chapter 4 to set up

3.3 SIP Servlet v1.1 TCK Contents

When unzipped and installed, the SIP Servlet v1.1 TCK (`sipservlet-1_1-tck.zip`) contains two directories – `tck` and `sipunit`. The `tck` directory is referred to as *TCK_DIRECTORY* throughout the SIP Servlet v1.1 TCK documentation. The SipUnit installation directory – `sipunit` is referred to as *SIPUNIT_DIRECTORY*.

For example:
If you unzipped the bundle in `/tmp`, then *TCK_DIRECTORY* would be `/tmp/tck` and *SIPUNIT_DIRECTORY* would be `/tmp/sipunit.`

When the SIP Servlet v1.1 TCK is unzipped, several directories are created under the directory where you unzip the TCK (`sipservlet-1_1-tck.zip`). The contents of these directories are as follows (on Win32 platforms assume backslashes in directory paths, instead of forward slashes used here).

File or Directory	Contents
SIPUNIT_DIRECTORY	The directory contains SipUnit source code that has been extended for SIP Servlet v1.1 TCK, license files, libraries etc.
TCK_DIRECTORY	The directory contains documents, source code, configuration files, scripts etc of the SIP Servlet v1.1 TCK distribution. The Release Notes document and README are also contained in this directory.
applications/	Contains applications for Javadoc API Assertion test and Specification Assertion Test.
bin/	Contains scripts for the setting environment variables and executing Signature Test, Javadoc API Assertion Test, Specification Assertion Test. The scripts are provided for Windows and Linux.
conf/	Contains configuration files which should be edited as described in this guide.
dist/	Contains Application Router and applications of SIP Servlet v1.1 TCK to be deployed on the SIP Servlet v1.1 implementation to be tested.
docs/	Contains *`SIP Servlet v1.1 TCK User Guide`*, *`SIP Servlet v1.1 Javadoc API Assertion List`*, *`SIP Servlet v1.1 Spec Assertion List.`*
lib/	Contains libraries used by the SIP Servlet v1.1 TCK.
src/	Contains source code for SIP Servlet v1.1 TCK.

3.4 Configuring SIP Servlet v1.1 TCK

`1. signature.properties`

For the Signature Test, the *TCK_DIRECTORY*/conf/signature.properties file *`must`* be configured correctly. This file has only one entry to set: *`ReferencedJSR289APIJAR`*. Its value indicates your implementation jar file path.

For example:

In Windows:

```
ReferencedJSR289APIJAR=D:/sipserver/lib/sipservlet.jar
```

In Linux:

```
ReferencedJSR289APIJAR=/home/abc/sipserver/lib/sipserver.jar
```

`2. default.properties`

For the Javadoc API Assertion Test and Specification Assertion Test, the *TCK_DIRECTORY*/conf/default.properties *`must`* be configured as follows:

The items in BLUE should be changed according to the actual test environment.

- `server.host` should be the IP Address of SIP Servlet v1.1 implementation hosting machine.

- `server.port` should be the port number of SIP Servlet v1.1 implementation receiving the SIP message.
- `server.http.port` should be the port number of SIP Servlet v1.1 implementation receiving the HTTP request.
- The client side UA properties are for the TCK UA to send and receive SIP messages. You can modify these UA properties; the only restriction being that each of the 4 UA ports *must* be unique.

The following are the contents of *`default.properties`*.

```
#default properties

#server side IP address
server.host=10.0.0.1
server.port=5060
server.http.port=7001
server.application.httproot=apitestapp

#client side transport used to send/receive sip message
transport=udp

#client side UA properties
ua1.username=alice
ua1.displayname=Alice
ua1.port=5071
ua2.username=bob
ua2.displayname=Bob
ua2.port=5072
ua3.username=jim
ua3.displayname=Jim
ua3.port=5073
ua4.username=tom
ua4.displayname=Tom
ua4.port=5074
#wait duration of client side to receive sip message
wait.duration=5000
```

3. `log4j.properties`

The SIP Servlet v1.1 TCK uses Apache log4j for logging. The *`log4j.properties`* can be found in the *TCK_DIRECTORY*/conf/ dir. For log4j configuration, please refer to the web site: http://logging.apache.org/log4j/1.2/manual.html.

```
#log4j.rootLogger= WARN, Console, tck

log4j.appender.Console=org.apache.log4j.ConsoleAppender
```

```
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=(%r ms) [%t] %-5p: %c#%M
%x: %m%n

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%d] %c{1}:%L %-5p - %m%n

log4j.appender.tck=org.apache.log4j.DailyRollingFileAppender
log4j.appender.tck.File=tckServer.log
log4j.appender.tck.DatePattern='.'yyyy-MM-dd
log4j.appender.tck.layout=org.apache.log4j.PatternLayout
log4j.appender.tck.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L -
%m%n


log4j.appender.client=org.apache.log4j.DailyRollingFileAppender
log4j.appender.client.File=tckClient.log
log4j.appender.client.DatePattern='.'yyyy-MM-dd
log4j.appender.client.layout=org.apache.log4j.PatternLayout
log4j.appender.client.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L
- %m%n

log4j.logger.com.bea.sipservlet.tck.apps=debug,tck,stdout
log4j.logger.com.bea.sipservlet.tck.agents=debug,client,stdout
```

3.5 Setting the Test Environment

The *setupENV.bat* and *setupENV.sh* are used to set environment variables for the test environment on Windows and Linux operating systems respectively.
These scripts are run automatically when you execute any of the following scripts to run the TCK test:
runAllTests.bat / runAllTests.sh
runAPITest.bat / runAPITest.sh
runSPECTest.bat / runSPECTest.sh
runSignatureTest.bat / runSignatureTest.sh

4 Preparing the SIP Servlet v1.1 Implementation

This chapter describes the preparation of SIP Servlet v1.1 implementation to perform SIP Servlet v1.1 TCK. It contains the following sections:

- Installing the SIP Servlet v1.1 implementation
- Deploying the TCK Application Router (TCK AR)
- Deploying SIP Servlet v1.1 TCK applications

4.1 Installing SIP Servlet v1.1 Implementation

Install the SIP Servlet v1.1 RI (Reference Implementation) or the SIP Servlet v1.1 implementation you want to run the TCK against.

To install the SIP Servlet v1.1 RI, do the following steps:

1. Download and Install WLS 10.0 MP1 from:
   http://commerce.bea.com/showproduct.jsp?family=WLS&major=10&minor=1
   Considering you installed WLS 10.0 MP1 in the *`C:\jsr289RI`* dir, your BEA_HOME will be: *`C:\jsr289RI`*
2. Install the JSR 289 RI (Reference Implementation) using:
   `java –jar jsr289ri-installer.jar`
   During the RI installation, choose the BEA Home Directory as the one where you installed WLS 10.0 MP1 (i.e. *`C:\jsr289RI`* from step 1).
3. The RI installer will automatically create a domain called *`jsr289riDomain`*. The domain will be created in *`%BEA_HOME%\user_projects\domains\jsr289riDomain`*. You will start the Sip Servlet 1.1 implementation using the *`startWebLogic.cmd`* script in this dir. (On Linux platforms, use the *`startWebLogic.sh`* script.) The default username and password for this domain will be *`weblogic / weblogic`*. You can use these credentials to log on to the WebLogic console accessible by default at: http://localhost:7001/console. Similarly, you can use the same credentials to use WebLogic utilities such as the *`weblogic.Deployer`* for deploying applications.
4. Install the TCK Application Router by setting the *TCK_DIRECTORY* environment variable in the *`jsr289riDomain\bin\startWebLogic.cmd`* script to point to the location of the `tck` directory inside your TCK installation. For example, if you unzipped the TCK bundle (sipservlet-1_1-tck.zip) in `C:\tmp`, then set *TCK_DIRECTORY* to `C:\tmp\tck`
5. Start the SIP Servlet v1.1 RI using the *`%BEA_HOME%\user_projects\domains\jsr289riDomain\startWebLogic.cmd`* script. (On Linux platforms, use the `startWebLogic.sh` script.)
6. Deploy TCK applications to the RI SIP Server using the WebLogic console or the WebLogic deployment utility. For more information on deployment, please refer to WebLogic documentation at: http://e-docs.bea.com/wls/docs100/deployment.html

7. Note: The EAR application will be deployable using the console and *weblogic.Deployer* utility. The individual *sar* modules will not be deployable using the console, use the *weblogic.Deployer* utility instead.

To install a different SIP Servlet v1.1 implementation, refer to vendor specific installation instructions.

If a SIP Servlet v1.1 implementation or SIP Servlet v1.1 RI (Reference Implementation) is already running on your workstation, it is recommended that you shut it down and start a new, clean implementation.

4.2 Deploying TCK Application Router

The TCK AR (Application Router) has been packaged in accordance with the rules specified by the Java SE Service Provider framework and delivered within `sipservlet-1_1-tck.zip` for SIP Servlet v1.1 TCK. The AR jar (*tck-approuter.jar)* can be found under the *TCK_DIRECTORY*/dist dir. The TCK AR jar file contains the TCK specific Application Router implementation. The jar includes *META-INF/services/javax.servlet.sip.ar.spi.SipApplicationRouterProvider* file which points to the TCK specific *SipApplicationRouterProvider* class.
The TCK AR *must* be deployed on the SIP Servlet v1.1 implementation before executing any TCK tests. Each SIP Servlet v1.1 implementation may have a proprietary AR deployment mechanism. But for SIP Servlet v1.1 TCK, it is recommended that the SIP Servlet v1.1 implementation support the deployment mechanism for the TCK AR as specified in JSR 289 specification Section 15.4.2.
As per the specification, the following steps *must* be followed for deploying the AR:

1. Include the *tck-approuter.jar* in the system classpath of the SIP Servlet v1.1 implementation to test.
2. Set the following system property "*javax.servlet.sip.ar.spi.SipApplicationRouterProvider*" defined in JSR 289 specification to point to the specific provider class in the *tck-approuter.jar* while starting the SIP Servlet v1.1 implementation as follows:

-Djavax.servlet.sip.ar.spi.SipApplicationRouterProvider=com.bea.sipservlet.tck.ar.TckApplicationRouterProvider

com.bea.sipservlet.tck.ar.TckApplicationRouterProvider is the concrete subclass of `javax.servlet.sip.ar.spi.SipApplicationRouterProvider`, and is used by SIP Servlet v1.1 implementation to load the AR for SIP Servlet v1.1 TCK tests.

The tested SIP Servlet v1.1 implementation may also provide its own deployment mechanism for the application router. For executing SIP Servlet v1.1 TCK cases, you may also deploy the TCK AR using the implementation's proprietary way, as long as the TCK AR is deployed successfully and works well within the SIP Servlet v1.1 container.

4.3 Deploying SIP Servlet v1.1 TCK Applications

The applications for SIP Servlet v1.1 TCK tests are all included in *TCK_DIRECTORY*/dist dir of `sipservlet-1_1-tck.zip.`

The dist dir contains both SAR and EAR application archives. You can deploy either *`tck-apps.ear`* or deploy each SAR archive separately on the SIP Servlet v1.1 implementation.

The following application archives are included:

- *`tck-apps.ear`*
- *`apitestapp.sar`*
- *`uas.sar`*
- *`uac.sar`*
- *`proxy.sar`*
- *`b2bua.sar`*
- *`ar-continue.sar`*
- *`ar-internalroutemodifier.sar`*
- *`ar-reverse.sar`*
- *`ar-suburiregion.sar`*
- *`ar-success.sar`*

5 Verifying SIP Servlet v1.1 TCK

After installing the SIP Servlet v1.1 implementation and SIP Servlet v1.1 TCK, you may want to quickly verify your installation by running the Javadoc API Assertion Test of the TCK. This procedure is optional and complete configuration instructions are provided in Chapter 3, “Configuration and Setup.”

Execute *runAPITest.bat/ runAPITest.sh*

The following output indicates a successful test:

```
Buildfile: build.xml

init:
     [echo] ANT_HOME=D:\Work\eclipse\plugins\org.apache.ant_1.6.5

check-tckjunit:

check-junit:

check-sipunit:
     [echo] The sipunit.jar exists in the TCK lib.

clean-api-reports:
     [echo] delete D:\Work\dev\jsr\289\tck/report/api_reports
[delete] Deleting directory D:\Work\
dev\jsr\289\tck\report\api_report

run-api-tests:
    [mkdir] Created dir: D:\Work\dev\jsr\289\tck\report\api_report
 [tckjunit] Running
com.bea.sipservlet.tck.agents.api.javax_servlet_sip.AddressTest
 [tckjunit] Tests run: 12, Failures: 0, Errors: 0, Time elapsed:
37.679 sec
……
```

If there are a number of Failures or Errors after running the AddressTest, cancel the test and check the configuration, or go to Chapter 7 “Troubleshooting”.

To check the test’s results, view the test report from *TCK_DIRECTORY*/report/api_report/index.html. The report from the AddressTest run will be available in: *TEST-com.bea.sipservlet.tck.agents.api.javax_servlet_sip.AddressTest.xml.*

6 Testing Your Implementation

This chapter describes how to use the SIP Servlet v1.1 TCK to test your implementation. It contains the following sections:

- Testing a SIP Servlet v1.1 Implementation
- Excluding Tests
- Requirements and Constraints
- Test Reports

This chapter assumes that you have verified that your TCK is properly configured as described in Chapter 5, “Verifying the SIP Servlet v1.1 TCK.”

6.1 Testing a SIP Servlet v1.1 Implementation

For performing the full testing of TCK on a target SIP Servlet v1.1 implementation, the following steps introduced in these chapters must be done firstly:

- Chapter 3, “Configuration and Setup”.
- Chapter 4, “Preparing the SIP Servlet v1.1 Implementation”.
- Chapter 5, “Verifying SIP Servlet v1.1 TCK”.

Then the subsequent steps are:

1. Start the target SIP Servlet v1.1 implementation, i.e. the tested SIP servlet container.
2. Start the deployed SIP applications, if necessary. (Some SIP servlet containers require the deployed applications be started in order to receive incoming requests.)
3. Select the excluded test cases. Please refer to Section 6.2.
4. Start the TCK test by running a script. The TCK provides DOS batch scripts (*.bat) and Bash scripts (*.sh). These scripts can be found in *TCK_DIRECTORY*/bin. The scripts are listed below:

- runAllTests.bat/runAllTests.sh runs all the test cases in sequence, including the signature test, API test, and specification test.
- runSignatureTest.bat/runSignatureTest.sh runs the signature test.
- runAPITest.bat/runAPITest.sh runs the API test
- runSPECTest.bat/runSPECTest.sh runs the specification test.
- XXX_solaris.sh is used to run TCK under solaris OS.

The signature test requires additional configuration; see Chapter 3 for more information.

Additionally, setupEnv.bat/setupEnv.sh is used to setup environment, and it is called by other scripts internally. It is not necessary to run it directly.

5. Check the results. The reports are saved in different directories.

Test results are displayed by the test reports, which can be found in *TCK_DIRECTORY*/report.

6.2 Excluding Tests

TCK allows some test cases to be optional and not be tested. The optional cases are specified in the *TCK_DIRECTORY*/conf/excludeList property file. The property file is in a simple, line-oriented format, and each line specifies an optional test package, class, or method. Any listed packages, classes, and methods are not tested by TCK.
The excluded list is predefined by TCK. The cases contained in it are used to test optional features of JSR 289.
The file can be modified in order to customize the exclusion list. If some excluded entry is no longer wanted, insert a "#" (comment delimiter) at the beginning of the line.

NOTE: the directory of the excludeList.properties must be set into the CLASSPATH. If the excludeList.properties can't be found by the class loader all the cases will executed by default. In this case the following warning will be printed:

****
********** ***Can't find the excludeList.properties file!*** **********
******Please make sure the file's directory is set into the CLASSPATH.*******
****

The script of *TCK_DIRECTORY*/bin/setupENV.sh/bat has done this job, so you don't care about the CLASSPATH issue if you want to execute the TCK by running the scripts. But if you want to execute the TCK manually by ANT, which is described in section 7.6, and also want to exclude some test cases at the same time, you must set the directory of the excludeList.properties (i.e. *TCK_DIRECTORY*/conf) into CLASSPATH by yourself.

6.3 Requirements and Constraints

- **Only those applications and AR (Application Router) provided by TCK should be deployed on the tested SIP servlet container.**

The deployment environment should be clean. Any other applications and AR should not be deployed in order to avoid the potential conflict among the applications and ARs which will probably affect the test results.

- **Don't run more than one script at a time.**

The SIP Servlet v1.1 TCK can be executed concurrently, but the tested SIP servlet container's support for concurrent process is unknown. For the test result reliability it is recommended to run the tests in sequence.

- **Don't deploy SIP applications provided by TCK in a cluster environment.**

Some un-serializable objects are used as attributes by SIP session in TCK applications, so some cases will fail in distributed environment.

- **Don't run TCK by TCP.**

Because of some constraints of JAIN-SIP (the SIP stack used by TCK client side) the test cases about proxy will fail if the TCP is used. Keep the 'transport' configuration as 'udp' in the TCK_DIRECTORY/conf/default.properties.

- **Don't configure IPV6 address(es) for TCK.**

TCK doesn't support IPV6 currently in both client side and server side. If IPV6 address is configured some unexpected result might be produced.

6.4 Test Reports

A set of report files will be generated after the TCK is executed.

- *TCK_DIRECTORY*/report/api_report: the API test report.
- *TCK_DIRECTORY*/report/spec_report: the specification test report.

The API and specification test reports are html-based format and in framed style, just like general Javadoc format. Open the index.html in the api_repot/spec_report directory to view the report. The report includes the following information:

  - Environment information: system class path, working directory, some system environment variables such as JAVA_HOME, JDK version, and referenced library, etc.
  - Tests result summery information: total tests number, errors number, failures number, success rate and the time spent to run the tests.
  - Detailed tests result information: for every test class, all the test methods, and their corresponding assertion IDs, test results, and the error information if the test methods failed.

- *TCK_DIRECTORY*/report/sig_report: the signature test report.

The signature report is a text file named "***jsr289SignatureReport.txt***". The report lists the tested differences between the reference API and the implementation under test, such as missing classes, added methods, etc.

7 Troubleshooting

There are a number of reasons that tests can fail to execute properly. This chapter provides some approaches for dealing with these failures.

7.1 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.
A test can fail with two results: error and failure.

- Error. A test with error means the test execution is broken because some error occurs, such as unexpected exceptions.
- Failure. A test whose result is failure means it is executed but the result is not as expected.

SIP servlet v1.1 TCK is a client-server mode program, so client and server can both cause test failure. Several tools can be helpful to locate the problem: test report, client and server side logs, and JAIN-SIP logs for client side SIP stack.

7.2 Checking Test Report

Test report should be the first thing to check when one test fails. The report will give a summary of the test result. The report is like the normal JUnit framed html-based report generated by Apache ANT which is very popular in Java world. The only difference to the generic JUnit is in every test class result page there is an “Assertion” column which indicates the assertion IDs corresponding to the test method.
The test report also contains the basic test environment information and the error or failure information which would be useful for trouble shooting.
Please refer to the JUnit web site (http://www.junit.org) and Ant web site (http://ant.apache.org) for more information.
If almost all the test case fail, and the “Type” column in the report indicates that the client SIP message is not able to be sent to the server, or that the server side SIP message can’t be received by the client, the most possible cause is that the configuration is wrong. Section 7.3 will describe how to check the configuration.

7.3 Checking Configurations

This section describes some common problems caused by wrong configuration. Chapter 3 describes TCK configuration in detail.

- Server or client address and port are not correct.

The most important configuration file is *default.properties* which locates in *TCK_DIRECTORY*/conf/. Server and client addresses and port numbers are configured in it. If the addresses or the port numbers are wrong, the SIP message will not be sent or received properly. Make sure they are all correctly configured.

- Some client UAs are missing.

The TCK needs 4 UAs (User Agents) in the client side. Normally a *NullPointerException* will be thrown during initializing the client if any one of them is missing in the configuration. And the UA name *must* be ua1, ua2, ua3 and ua4, for example, ua1.username=alice, or ua3.port=5073. Please check *default.properties* for it.

- The path of SIP servlet implementation library is not correct in signature test.

The signature test needs to configure the path of your SIP servlet implementation library in *TCK_DIRECTORY*/conf/*signature.properties*. If the *ReferencedJSR289APIJAR* refers to a non-exist file or directory, the signature test will report such information: “The XXXX does not exist.” If the *ReferencedJSR289APIJAR* refers to an invalid jar file or directory, the signature test will run but the report will say that all the classes are missing.

- Set a proper waitDuration.

waitDuration item in *default.properties* describes how long the client side waits for the messages coming from the server side.

Some test cases test callback interfaces/methods such as SipSessionListener. JSR289 doesn’t define how to implement those interfaces/methods, so containers can implements them in single-thread or multi-thread style. If the tested container use multi-thread technology (i.e. the asynchronous style) to implement those interfaces/methods, it is hard to know how long to wait those methods being invoked, so it is also difficult to know when the messages initiated by those callback methods arrive to the client side. Additionally the performance of the hardware and software is various, which will affect the wait duration too. In a word you should adjust the *waitDuration* according your own condition.

7.4 Checking Exclude List

If some test cases are not run, and they do not exist in the test report, they might be excluded. Check the *excludeList.properties* in the *TCK_DIRECTORY*/conf/.

The *excludeList.properties* has been predefined by TCK, and normally user does not need to modify it. TCK supports to exclude test cases on method level and class level. If some test methods are excluded, they will not be executed and not be recorded in the report. If some test classes are excluded, all of their test methods will not be executed, and zero tests will be recorded in the report.

If you want to include an excluded case into TCK, you can delete that case from the *excludeList.properties*.

7.5 Checking Log Files

TCK will generate log files in both server side and client side. JAIN-SIP will also generate detailed SIP stack logs at client side at the same time.

TCK uses log4j to generate logs. The original log4j.properties can be found in *TCK_DIRECTORY*/conf/, and it specifies both server side and client side log behavior, including the log file name, log file format, and log level. It will be copied to *TCK_DIRECTORY*/build/classes and be loaded by the TCK client there. For the server side, you ***must*** copy it to a directory in which it can be loaded by the tested SIP servlet container. If you want to modify the log4j.properties, *DON'T FORGET* to copy it to the server side directory and *TCK_DIRECTORY*/build/classes, and the server may need to restart.
Log4j is a frequently used software in Java project, you can refer to its web side for more help: http://logging.apache.org/log4j/index.html

- Client side log file: it is generated in *TCK_DIRECTORY/,* named with tckClient.log.
- Server side log file: it is named tckServer.log, and its location depends on your SIP servlet container's implementations.
- JAIN-SIP's log file: TCK use JAIN-SIP as its client side SIP stack and JAIN-SIP will produce logs for every UA. In *TCK_DIRECTORY/*, you can find these logs. The log file names are listed below for your reference:
  - testAgent1_debug.txt: the detailed stack's debug information for UA1.
  - testAgent1_log.txt: the SIP message sent and received by UA1.
  - testAgent2_debug.txt: the detailed stack's debug information for UA2.
  - testAgent2_log.txt: the SIP message sent and received by UA2.
  - testAgent3_debug.txt: the detailed stack's debug information for UA3.
  - testAgent3_log.txt: the SIP message sent and received by UA3.
  - testAgent4_debug.txt: the detailed stack's debug information for UA4.
  - testAgent4_log.txt: the SIP message sent and received by UA4.

Testing process information is all recorded in those log files which are very helpful for troubleshooting.

7.6 Running TCK with ANT

Besides running the TCK by the scripts in *TCK_DIRECTORY*/bin, you can also use ANT to execute TCK. Please visit the website http://ant.apache.org/ for more information about ANT. There is a build.xml which is an ant script in *TCK_DIRECTORY*/. Suppose you have been in the *TCK_DIRECTORY*/ and your operating system is UNIX, then you can type the command "***ant usage":***

```
% ant usage
Buildfile: build.xml
usage:
     [echo] Usage: ant target [options]
     [echo]  build-all : Build all apps to tck-apps.ear.
     [echo]              which located in tck/dist directory.
     [echo]              Build the application router to tck-
approuter.jar.
     [echo]              which located in tck/dist directory.
```

```
     [echo]              Build all tests.
     [echo]              which located in tck/build.
     [echo]  clean-all : Clean the tck-apps.ear, tck-approuter.jar,
all tests classes
     [echo]              and all reports.
     [echo]  test-all  : Run the signature Test,API Assertion Tests
and the SPEC Assertion Tests
     [echo]              and generate the reports.
     [echo]              All reports are located in tck/report.
     [echo]              The Signature Test report are located in
tck/report/sig_report.
     [echo]              The API Assertion Tests report are located
in tck/report/api_report.
     [echo]              The SPEC Assertion Tests report are located
in tck/report/spec_report.
```

There are still some other targets which are useful for troubleshooting. They are summarized as below:

- General targets

build-apps: Build all applications.
build-ar: Build TCK application router.
clean-ar: Clean TCK application router.
clean-all: Clean all applications and client tests.
clean-apps: Clean all applications.
clean-tests: Clean all client tests.

- Specification Assertion Test

build-spec-apps: Build the Specification Assertion Test applications.
clean-spec-apps: Clean the Specification SPEC Assertion Test applications.
build-spec-tests: Build the Specification SPEC Assertion Test client files.
clean-spec-tests: Clean the Specification SPEC Assertion Test client files.
clean-spec-reports: Clean the Specification SPEC report files.
run-spec-testcase: Execute and generate the report. This target needs one more parameter: *-Dcase=<testName>*, which indicates the test class name.

- Javadoc API Assertion Test

build-api-app: Build the Javadoc API Assertion Test application.
clean-api-app: Clean the Javadoc API Assertion Test application.
build-api-tests: Build all client tests for Javadoc API Assertion Test.
clean-api-tests: Clean all client tests for Javadoc API Assertion Test.
run-api-tests: Run all tests and generate the report.
run-api-testcase: Execute and report a specified test case. This target needs one more parameter:
-Dcase=<testName>: which indicates the test class name.

- Signature Test

signature-test: Run signature test and generate the report.

Among those ANT targets, probably the most useful targets for troubleshooting are *run-spec-testcase* and *run-api-testcase.* Running all the tests in the TCK is time consuming and is not convenient for troubleshooting and debugging, so if you want to debug one class in the API test (for example the class is *ProxyBranchTest*), you can invoke the *run-api-testcase* ant task as follows:

% **ant run-api-testcase -Dcase=ProxyBranchTest**

Then only the ProxyBranchTest will be executed, which will save much time.

NOTE: Occasionally some cases will fail because the UAC can't receive expected responses. The reason of such failures might be (1) the constraints of the SIP stack (JAIN-SIP and SIPUnit) on the client side; (2) UAS sends SIP messages too frequently and the UAC cannot process the messages in time. For such failures you can re-run the failed cases only using the ant task above and in most cases they will succeed.