

# Compatibility Testing

---

By Stan Avzan and Bruce Hartford

PRELIMINARY

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms.

Sun, Sun Microsystems, Java, JavaTest and JavaHelp are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

---

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

L'utilisation est soumise aux termes de la Licence.

Sun, Sun Microsystems, Java, JavaTest et JavaHelp sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.



# Table of Contents

---

Java Compatibility	5
Ensuring Compatibility	6
Java Specifications and Tests	7
Compatibility and Quality Tests	8



# Compatibility Testing

---

**Summary:** Java™ technology applications run on a wide variety of operating systems and devices. This requires that Java platforms be mutually compatible with each other and provide consistent environments within which applications can be run. “Compatibility” testing is the means by which the Java community ensures such consistency. Therefore, compatibility testing is an essential aspect of implementing the various Java technologies on different platforms and devices.

Compatibility testing differs from quality testing in that compatibility testing ensures that a particular Java technology is compatible with all other implementations of that technology on each platform. Quality testing ensures that an implementation performs correctly and that it operates on the designated platforms within the specified parameters.

---

## Java Compatibility

The essential value of Java technology can be summed up by “Write Once, Run Anywhere.”

Java is both a device-independent programming language and a multi-device platform on which a huge array of products and applications can be run. At the present time, over 5 million software developers use the Java programming language and platform to create products for a wide range of devices, computers, and networks. Today, Java powers more than 800 million PCs, over a billion mobile phones and other handheld devices, a billion and a half smart cards, plus set-top boxes, printers, web cameras, games, car navigation systems, lottery terminals, medical devices, parking payment stations, and so on.

Java's "Write Once, Run Anywhere" portability and versatility mean that applications written in the Java programming language can not only run on any Java platform, but also provide cross-platform interoperability. For "Write Once, Run Anywhere" to be true, Java platforms have to be mutually compatible with each other and provide consistent environments within which applications can be run.

---

## Ensuring Compatibility

The Java Community Process™ (JCP™) is a multi-company program that extends and expands Java technologies by cooperatively creating new specifications and updating old ones. The JCP also ensures that all implementations of a specification are mutually compatible regardless of platform. This requires that:

- **Implementations must conform to the applicable specification.** When Java technologies conform to their specifications, they operate consistently on different platforms and operating systems, and with other Java technologies.
- **Implementations must be tested to ensure that they are compatible with the applicable specification or specifications.** Without testing, there is no way to be sure that a given implementation conforms to the corresponding specification.

Every Java technology specification developed under the JCP program includes the following:

- **Technology specification** – a written specification for some aspect of the Java technology. This includes the language, virtual machine, Platform Editions, Profiles, and application programming interfaces.
- **Reference Implementation (RI)** – the prototype or "proof of concept" implementation of a Specification.
- **Technology Compatibility Kit** – the suite of tests and tools that allows an organization to determine if its implementation is compliant with the specification and compatibility rules.

A Technology Compatibility Kit (TCK) is a complete, ready-to-use, suite of tests and tools to ensure that each implementation conforms to the specification and meets the requirements of the compatibility rules. A TCK tests all aspects of a specification that impact how compatible a particular implementation of that specification is, such as the public API, all mandatory elements of the specification, and all optional features defined by the specification. A vendor's implementation of a specification is only considered "compatible" if the implementation fully and completely passes the specification's TCK.

Because all implementations of a Java specification have to pass the *same* TCK tests, application developers can be confident that any application they create based on that technology will work on all Java platforms that provide the technology.

---

## Java Specifications and Tests

Java technology specifications are written descriptions of some aspect of the Java technology covering the language, virtual machine, platform editions, profiles, and application programming interfaces. Many different vendors implement a given specification on a wide range of devices.

Software developed in conformance to a Java specification requires two different kinds of testing. Compatibility tests are run on Java technology implementations to ensure that they conform to their corresponding Java specifications. Quality tests are run on implementations to ensure that they operate as intended on the designated platforms.

Both kinds of tests – compatibility and quality – must be run during the course of development. **Compatibility testing, therefore, is an *essential*, part of overall testing.**

An implementation that passes all of its TCK tests is considered to be “compatible,” meaning that it works as expected according to the Java specification, and behaves in a consistent manner across all platforms on which it is implemented. An implementation that passes its quality tests is assumed to function as intended on its designated devices.

TABLE 1 summarizes some of the essential differences between compatibility and quality tests:

**TABLE 1** Summary – Compatibility Tests Compared to Quality Tests

	Compatibility Tests	Quality Tests
<b>Name:</b>	“Compatibility tests” or “TCK tests.”	“Quality” or “Q/A” tests.
<b>Purpose of tests:</b>	To ensure that a particular Java technology is compatible with all other implementations of that technology on each platform.	To ensure that an implementation performs as specified, that it’s features work correctly, and that it operates on the designated platforms within the specified parameters.
<b>Tests apply to:</b>	All implementations of a particular Java technology specification by all vendors.	A particular implementation from a particular platform vendor or team of vendors.

**TABLE 1** Summary – Compatibility Tests Compared to Quality Tests (*Continued*)

	Compatibility Tests	Quality Tests
<b>Tests created by:</b>	Under the JCP process, TCK tests are created by the multi-company task force that is responsible for creating or improving Java specifications.	An individual platform vendor or team of vendors.
<b>Tests approved by:</b>	TCKs are approved by vote of the entire JCP consortium as an integral part of adopting a new or updated specification.	Created and approved by the vendor.
<b>Test structure:</b>	Technology Compatibility Kit (TCK).	Determined by the vendor.
<b>Tests must be run by:</b>	All vendors implementing the Java technology.	Only the vendor.
<b>Pass-Fail criteria:</b>	An implementation <i>must</i> pass <i>all</i> TCK tests in order to be certified as “Java compatible.”	Pass-Fail criteria is determined by each vendor.
<b>Scope limitations:</b>	Compatibility tests do not ensure quality.	Quality tests do not ensure implementation compatibility.

## Compatibility and Quality Tests

The two kinds of tests – compatibility and quality – are different as listed in TABLE 2.

**TABLE 2** Detailed Comparison of Compatibility and Quality Tests

	Compatibility Tests	Quality Tests
<b>Tests cover:</b>	Conformance to a Java technology specification and compatibility with other implementations of that technology	Functionality of features Performance parameters (benchmarks) Component integration tests Stress tests Regression tests Usability and user-interface tests Customer satisfaction tests
<b>Test applicability</b>	Compatibility tests must be runnable on a wide spectrum of implementations. They are used on potentially different operating systems with varying requirements for resources.	Quality tests are typically written for a specific implementation.
<b>Test quality:</b>	TCKs are themselves subject to rigorous testing and include detailed documentation.	Varies according to the needs and requirements of each vendor.

**TABLE 2** Detailed Comparison of Compatibility and Quality Tests (*Continued*)

	<b>Compatibility Tests</b>	<b>Quality Tests</b>
<b>Test validation:</b>	Compatibility tests are validated by running them against a Reference Implementation (RI) that is known to be a valid implementation of the specification. A TCK test that fails against an RI is excluded from the TCK and implementors are not required to run that test against their implementations.	Quality tests are not validated by running them against an implementation, rather implementations are validated by running quality tests, TCK tests, and possibly other tests against them. Since quality tests are specific to a particular implementation from one company, there is no requirement that test bugs be corrected or otherwise addressed.
<b>Test approach:</b>	Compatibility tests are behavior based. They test specified behavior with a minimum knowledge or concern for the underlying structure of the code. “Black-box,” “functional testing,” “data-driven,” and “input-output-driven testing” are all common industry terms for this kind of testing.	Quality tests are structure and behavior-based. They rely on a knowledge of the logic of the underlying code to guide the selection of test data. “White-box” and “internals-based testing” are common industry terms for this kind of testing.
<b>Test design:</b>	The most common type of TCK tests are unit tests. Unit tests verify the semantics of specific API calls or VM instructions. TCKs may also include functional tests to verify functionality that spans across APIs, such as RMI, GUI rendering, or networking.	Quality tests usually test complete implementations. For example, tests to ensure backward compatibility in a newer version, or to stress the platform in a way that real applications do.
<b>When tests are run:</b>	TCKs assume a minimum level of implementation quality needed for them to execute. For example, they include tool support to pass parameters to configurable tests. Therefore, TCKs may not be runnable during the early stages of development.	Quality tests often need to be started at the earliest stages of the development process. For example, there are typically a set of “smoke tests” to check the basic functioning of early versions of the implementation.
<b>Stress and performance tests:</b>	TCKs usually do not contain any stress tests or performance measurement code. Performance and other related limitations can vary with the particular implementation of the technology. An exception is when the related Java technology specification clearly defines certain limitations or requirements on resources. In this case the TCK must test these assertions accordingly.	Quality tests attempt to make full use of all available resources. They attempt to reach certain predefined limitations on the particular implementation under test. For example, stress tests might include opening a large number of windows, files or sockets, or launching a large number of threads or processes. They might also attempt to allocate all or most of the available memory.

**TABLE 2** Detailed Comparison of Compatibility and Quality Tests (*Continued*)

	<b>Compatibility Tests</b>	<b>Quality Tests</b>
<b>Coverage goals:</b>	Compatibility tests verify testable assertions in the Java specification. A main measure of quality for a TCK is assertion coverage including both breadth and depth. Breadth is what percentage of the specification assertions are covered by at least one test. Depth is how well each assertion is tested.	Quality tests have as their goal to ensure that the code works correctly. Thus, their quality is measured by what percentage of the code blocks or branches in are tested.
<b>Test configurability:</b>	Predefined configuration parameters are typically beyond the scope of the Java technology specification so no assumptions can be made about them in TCK tests. As a result, TCK tests need to use the smallest possible amount of resources to verify a specific assertion. When this isn't feasible, tests need to be configurable by parameters describing specific implementation under test.	Quality tests can make assumptions about how the implementation is configured and run based on consistent values such as those specified by configuration files and command-line parameters. The implementation itself typically has predefined requirements, such as a specific underlying OS or CPU, or specific memory or disk space allocations. Quality tests can make use of this specific information.
<b>Test evolution:</b>	TCK tests are bound by the specification and its evolution through the JCP process. All TCK tests must be strictly based on the specification. Even if there is an obvious problem in the specification, the TCK still needs to wait for an official specification update before the corresponding tests can be updated.	Quality tests enjoy absolute freedom in regards to test evolution. Since they are strictly internal to a particular implementation by a single vendor, there are no restrictions on test evolution.
<b>Test outcomes:</b>	The results of running TCK tests are binary: either Pass or Fail. An implementation must pass <i>all</i> TCK tests for it to be certified as compatible.	Quality test outcomes are at the discretion of the vendor. Pass-Fail requirements are determined by the vendor. Vendors can choose to add, modify or remove tests as they see fit.