Java Rule Engine API™ JSR-94

Java Community Process

http://java.sun.com/jcp/

Maintenance Lead:
Alex Toussaint, BEA Systems, Inc.

Technical Comments:
jsr-94-comments@jcp.org

Java Rule Engine API Test Compatability Kit (TCK)
Version: 1.0
Status: JCP Final Review
Release: September 15, 2003

Contents

Java Community Process

1 Introduction

The test compatibility kit (TCK) for the Java Rule Engine API is a suite of tests, tools, and documentation that allows the implementor of the Java Rule Engine API to determine if the implementation is compliant with the specification. This TCK includes the following components:

- Test Harness—This section describes the test tools used by the TCK and the information needed to install and run the TCK.
- Test Requirements—This section specifies how the TCK must be used for compatibility testing.
- Appeal Process—This section identifies the appeal process through which challenges to the TCK can be addressed.

The validity and invalidity of individual tests within the test suite is handled by the Technical Interpretation Committee under the direction of the Maintenance Lead. The Technical Interpretation Committee (TIC) is a panel of experts, which can be chosen from among members of the expert group and other JCP member organizations.

The Technical Interpretation Committee consists out of the following expert group members:

Company	Representative
ILOG	Changhai Ke
Fair Isaac	Johan Majoor
IBM	Rainer Kerth

2 Test Harness

The test harness consists out of a series of JUnit tests and uses “Ant” for its execution framework.

The TCK distribution includes both “Ant” version 1.4 and “JUnit” version 3.7. You can find more information on these tools on the following web sites:

http://www.junit.org

http://ant.apache.org/

2.1 Configuring the TCK

Before you run the TCK you have to configure the TCK to make use of the specific rule engine implementation that you want to verify.

2.1.1 jess.jar

You must download `jess.jar` and put it in the `/lib` directory before running tests on the reference implementation from the `/distrib` directory.

2.1.2 Specifying the RuleServiceProvider implementation

Change the RuleServiceProvider setting in the configuration file and specify the RuleServiceProvider implementation you want to verify.

See section “TCK Configuration Options” for more information.

2.1.3 Specifying the rule execution sets

The TCK requires the following two rule execution set definitions to be present:

·Tck_res_1.xml

For stateless rule session tests.

·Tck_res_2.xml

For stateful rule session tests.

The Rule Engine API does not standardize the content of the rule execution sets. You have to provide the two rule execution sets. See section “Overview Rule Execution Set definitions used by the TCK” for a detailed description of the content of the rule execution sets.

After creating the above mentioned rule execution sets, you can configure the TCK's rule execution set location parameter and point it to the location of your rule execution sets.

See section “TCK Configuration Options” for more information.

2.1.4 Specifying the rule engine implementation classes

The TCK requires you to specify where the jar file containing your rule engine API implementation can be located. You have to specify a valid URL in the TCK configuration file. See “TCK Configuration Options” for more information. A URLClassLoader will be constructed for this jar, for those test that require a classloader other then the default.

2.2 How to Run the TCK

You can run the TCK by executing the following command:

ant\bin\ant -f run_tck.xml

This command will start the “Ant” utility and run all TCK tests.

During the test run you will get information about the progress and status of each individual test.

An example of the output of a successfully passed test.

Running org.jcp.jsr94.tck.RuleSessionTest

Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.19 sec

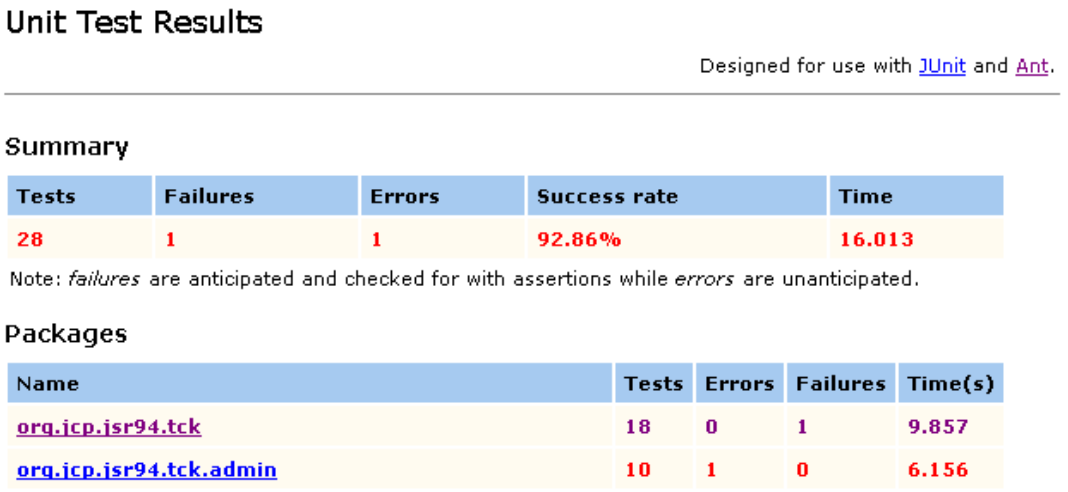
An example of the output of a test that failed:

```
Running org.jcp.jsr94.tck.StatefulRuleSessionTest

Tests run: 1, Failures: 1, Errors: 0, Time elapsed: 1.041 sec

TEST org.jcp.jsr94.tck.StatefulRuleSessionTest FAILED
```

2.3 Verifying the Results

The results of the complete test run can be found in the “reports” directory. This directory is created by the TCK run and contains detailed information about the execution of all tests.

Open the file “index.html” in your browser and verify that all test have completed successfully.



Unit Test Results

Designed for use with JUnit and Ant.

Summary

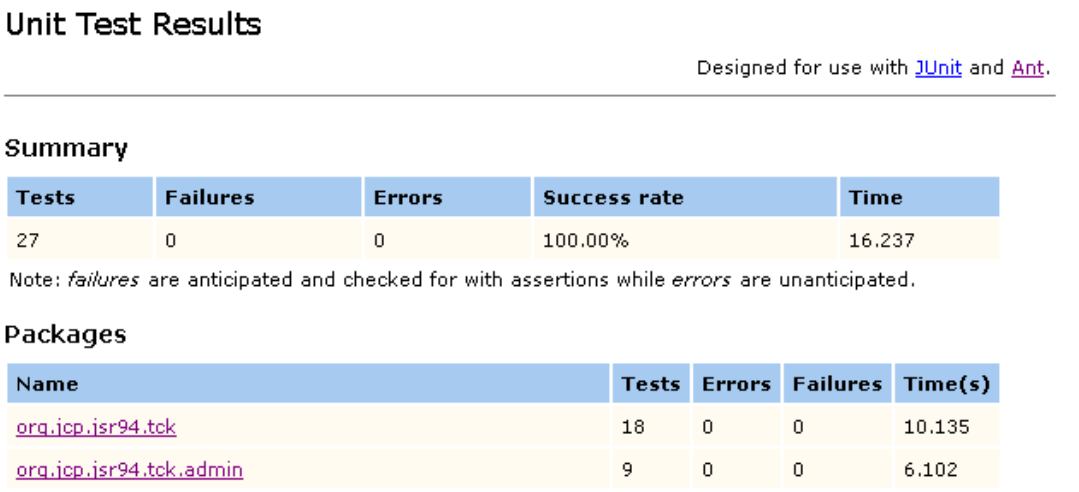
Tests	Failures	Errors	Success rate	Time
28	1	1	92.86%	16.013

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)
org.jcp.jsr94.tck	18	0	1	9.857
org.jcp.jsr94.tck.admin	10	1	0	6.156

Follow the links on the page for detailed information when failures have been reported.



Unit Test Results

Designed for use with JUnit and Ant.

Summary

Tests	Failures	Errors	Success rate	Time
27	0	0	100.00%	16.237

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)
org.jcp.jsr94.tck	18	0	0	10.135
org.jcp.jsr94.tck.admin	9	0	0	6.102

If you develop a compliant implementation of the specification, please e-mail the Maintenance Lead at jsr-94-comments@jcp.org.

3 Test Requirements

3.1 Generic Testing Requirements

The following is an overview of the generic testing requirements for the Java Rule Engine TCK:

- Other than test configuration files, no test source or binary may be modified in any way.

- Every required test must be able to pass on every publicly documented configuration of the Java Rule Engine API.
- The functional programmatic behavior of every binary class and interface must correspond to the Java Rule Engine API Specification.
- No sub-setting, super-setting, or modification of the public or protected API is allowed except as defined in the specification.

3.1.1 Overview of the Rule Execution Set Definitions Used by the TCK

The contents of a `RuleExecutionSet` is not within the scope of the JSR-94. The implementation given in this file is written for the reference implementation. A rule engine vendor verifying their rule engine should modify the files to their specific needs.

The following rule execution sets are used for input to the rule engine. You are allowed to make the necessary changes to the contents of these files:

- Basic Stateless `RuleExecutionSet`

  The definition of this rule execution set can be found in the `tck_res_1.xml` file. This `RuleExecutionSet` will be invoked by the TCK in a stateless manner.

  The rule execution set must have support for the following business object model:

  - Customer Class—The Customer business object is a simple business object that contains a name and credit limit property. The definition of this class can be found in `org.jcp.jsr94.tck.model.Customer`.
  - Invoice Class—The Invoice business object is a simple business object that contains a description, amount, and status property. The definition of this class can be found in `org.jcp.jsr94.tck.model.Invoice`.

  The rule execution set has the following definition:

  - Support Customer and Invoice business objects.
  - Defines One Logical Rule—Rule1: If the credit limit of the customer is greater than the amount of the invoice and the status of the invoice is unpaid, then decrement the credit limit with the amount of the invoice and set the status of the invoice to *paid*.

  Note: Additional physical rules may be defined to accomplish the requirements mentioned above.

The rule execution set has the following semantics:

Input:

- A Customer with a credit limit of 5000.
- An Invoice with an amount of 2000.

The rule execution should produce the following output:

- The credit limit of the customer is 3000.
- The status of the invoice is *paid*.

The rule execution set name `RuleExecutionSet1` may not be changed. This will cause a failure in one of the test cases.

- Basic Stateful `RuleExecutionSet`

  The definition of this rule execution set can be found in the `tck_res_2.xml` file.

  This `RuleExecutionSet` will be invoked by the TCK in a stateful manner. This rule execution set will first be invoked with one set of parameters, process this input, and then keep the state of the execution. A subsequent invocation will add additional information to the rule execution set and the processing will involve both the newly provided information as well as the processed information of the previous execution.

  The rule execution set must have support for the following business object model:

  - Customer Class—The Customer business object is a simple business object that contains a name and credit limit property. The definition of this class can be found in org.jcp.jsr94.tck.model.Customer.
  - Invoice Class—The Invoice business object is a simple business object that contains a description, amount, and status property. The definition of this class can be found in org.jcp.jsr94.tck.model.Invoice.

  The rule execution set has the following definition:

  - Support Invoice and Customer business object as input.
  - Defines One Logical Rule—Rule1: If the credit limit of the customer is greater than the amount of the invoice and the status of the invoice is unpaid then decrement the credit limit with the amount of the invoice and set the status of the invoice to *paid*.

Note: Additional physical rules may be defined to accomplish the requirements mentioned above.

The rule execution set has the following semantics:

The first input to the rule execution set is:

- A Customer with a credit limit of 5000.
- An Invoice with an amount of 2000.

The rule execution should produce the following output:

- The credit limit of the customer is 3000.
- The status of the invoice is paid.

The second input to the rule execution set is:

- An Invoice with an amount of 1500.

The rule execution should produce the following output:

- The credit limit of the customer is 1500.
- The status of the invoices is paid.

The rule execution set name `RuleExecutionSet2` may not be changed. This will cause a failure in one of the test cases.

3.1.2 TCK Configuration Options

All configurable parameters of the TCK are specified in the "tck.conf" configuration file located in the lib directory:

You can change the following settings:

·RuleServiceProvider implementation

<rule-service-provider>java class name</rule-service-provider>

Change the "java class name" to the class that implements the RuleServiceProvider interface.

·Location for the rule execution set definitions

<rule-execution-set-location>./lib</rule-execution-set-location>

Make sure that the directory you specifiy contains the tck_res_1.xml and tck_res_2.xml rule execution set files.

·URL to the jar file containing your rule engine API implementation

<rule-service-provider-jar-url></rule-service-provider-jar-url>

Specify a valid URL to your rule engine API implementation jar file.

Below is the configuration for the default configuration file for the RI.

```
<tck-configuration>
<rule-service-provider>org.jcp.jsr94.jess.RuleServiceProviderImpl
</rule-service-provider>


<rule-execution-set-location>./lib</rule-execution-set-location>

<rule-service-provider-jar-url>lib/jsr94-ri.jar</rule-service-pro
vider-jar-url>

</tck-configuration>
```

3.1.3 Signature Tests

Signature tests are required to verify whether or not modifications to the original specification have been made.

This test compares the signatures of a JSR-94 API implementation under test with the standard signatures produced from the reference implementation compliant with the API specification.

The Sig Test Tool from Sun is used to test the implementation's compatibility with the given JSR-94 API specification.

The following table gives an overview of the test involved.

Test Implementation	JSR-94 Package	Description	Required
`ApiSignatureTest`	`javax.rules`	Verify all API calls of the implementation under test.	Yes
`ApiSignatureTest`	`javax.rules.admin`	Verify all API calls of the implementation under test.	Yes

3.1.4 Public API Test cases

The public API test consists out of the following test cases:

Test	javax.rules	Required
HandleTest	Handle	Yes
ObjectFilterTest	ObjectFilter	Yes
RuleExecutionSetMetadataTest	RuleExecutionSetMetadata	Yes
RuleRuntimeTest	RuleRuntime	Yes
RuleServiceProviderManagerTest	RuleServiceProviderManager	Yes
RuleSessionTest	RuleSession	Yes
StatefulRuleSessionTest	StatefulRuleSession	No
StatelessRuleSessionTest	StatelessRuleSession	Yes

Test	javax.rules.admin	Optional
LocalRuleExecutionSetProviderTest	LocalRuleExecutionSetProvider	
RuleAdministratorTest	RuleAdministrator	
RuleExecutionSetProviderTest	RuleExecutionSetProvider	
RuleExecutionSetTest	RuleExecutionSet	
RuleTest	Rule	

3.1.5 Testing Exception Classes

The exception class tests consists out of the following test cases:

Test	javax.rules	Required
ConfigurationExceptionTest	ConfigurationException	Yes
InvalidHandleExceptionTest	InvalidHandleException	Yes
InvalidRuleSessionExceptionTest	InvalidRuleSessionException	Yes
RuleExceptionTest	RuleException	Yes
RuleExecutionExceptionTest	RuleExecutionException	Yes
RuleSessionCreateExceptionTest	RuleSessionCreateException	Yes
RuleSessionTypeUnsupportedExceptionTest	RuleSessionTypeUnsupportedException	Yes

Test	javax.rules.admin	Required
RuleAdministrationExceptionTest	RuleAdministrationException	Yes
RuleExecutionSetCreateExceptionTest	RuleExecutionSetCreateException	Yes
RuleExecutionSetUnregisterExceptionTest	RuleExecutionSetUnregisterException	Yes

4 Appeal Process

The test appeal process defines the escalation process in which challenges to compatibility test are evaluated and either accepted or rejected.

The Technical Interpretations Committee, whose members have an understanding of both the Java Rule Engine specification and the TCK handles the appeal process. The TIC is responsible for interpreting the specification and making all decisions on test challenges under the direction of the Maintenance Lead.

Any appeal process must adhere to the following assumptions:

- All reviews for TCK challenges go through the Maintenance Lead or its designee.
- TCKs are not restricted in the choice of tools, test suite format, or management of the changes due to appeals.
- All implementors are treated equally in terms of access to changes, pass/fail reporting, and compatibility rules and requirements.

4.1 Challenges to the TCK

- Who can make challenges to the TCK?

  Any party using the TCK as part of an active effort to implement the Java Rule Engine specification.

- What challenges to the TCK may be submitted?

  Only challenges based on the integrity or relevance of individual tests may be submitted. Challenges are written and sent to the Maintenance Lead or its designee contesting the validity of one or a related set of TCK tests. A detailed justification for why each test should be invalidated must be included with the challenge.

- How and by whom are challenges addressed?

  The Maintenance Lead evaluates the challenge and prepares a response. If the appeal is incomplete or unclear, it is returned to the sender for correction. The Maintenance Lead should check the validity of the test before writing a response. The Maintenance Lead will make an attempt to complete a response within 10 business days. If this is not possible, the Maintenance Lead must tell the sender when the response will be completed. If the challenge is identical to a previously rejected challenge, the Maintenance Lead is not required to escalate it to the TIC, but can send the previous TIC response. The Maintenance Lead sends new challenges and responses to the TIC for evaluation. The TIC must make a decision of the test validity or invalidity within 10 business days of receipt of the challenge and its response to the Maintenance Lead. All decisions must be documented with an explanation of why test validity was maintained or rejected.

- How are accepted changes to the TCK managed?

  If the test challenge is approved and one or more tests are invalidated, the Maintenance lead removes the test(s) from the Test Suite or invalidates the test(s) so as to exclude them as part of the testing requirements for all implementors. In addition alternate tests may be developed by the TIC for use by the test challenger. Such a test would be deemed equivalent to the original test for meeting the testing requirements.

4.2 Test Challenge and Response Forms

The test challenge form must include the following information:

- Test name and revision
- Specification involved
- Test Suite used (name and revision)
- Complaint (includes name of complainant and argumentation of why the test is invalid)

The test challenge response form must include the following information:

- Test name and revision
- Specification involved
- Test Suite used (name and revision)
- Defense (defender name and the argumentation of why the test is valid)
- Implications of test invalidity
- Alternatives