OSGi Service Platform, TCK Manual Release 1

July 7, 2006 4:50 pm

OSGi Service Platform TCK Manual

Release 1
August 2006

OSGi™ Alliance

Feedback

This specification can be downloaded from the OSGi web site:

```
http://www.osgi.org
```

Comments about this specification can be mailed to:

```
speccomments@mail.osgi.org
```

OSGi Board of Directors and Officers

Table Of Contents

1 Introduction

This document describes the OSGi Reference Implementation (RI) and Test Case Kit (TCK) for Release 4. The purpose of this document is to be used by companies that want to certify their OSGi Release 4 implementations. These implementations can be:

- Core R4.0
- Compendium R4.0
- Mobile R1.0

2 Test Setup

All TCKs have been written for the OSGi test harness. It is therefore necessary to setup an environment where both a target framework runs as well as the director framework. Setting up this harness is described in the Director Test Tool.

The necessary files to support the self certification are distributed in different ZIP files. The `osgi.tck.zip` file contains the harness. For each of the possible TCKs, a separate ZIP file is available. These files are separated separately due to licensing reasons.

1.1 The Execution Environment

It is advised to run both the director as well as the target framework on J2SE 1.4, this is the environment that has been used by the OSGi to verify the test suites. However, most tests are possible to run on smaller execution environments but not all. This information is contained in the about-xxx.html file, which is part of the specific TCK.

1.2 Installing the TCK Harness

The TCKs Harness is stored in the `osgi.tck.zip` zip file. This file should be downloaded and unzipped in an empty directory (called the base directory and denoted with "."). The ZIP archive contains the following files:

Table 1-1 *Contents of the osgi.tck.zip*

about.html	Describes content and license issues
director	Directory containing the harness files
build.xml	An ant script that runs the tests from the harness site.
osgi.properties	Contains the combined properties
doc	Documentation and coverage XML files
tck-manual.pdf	This manual
licenses.txt	License information, please read before usage
licenses	License subdirectory
org.osgi.impl.framework.eclipse.jar	The Eclipse Equinox Framework implementation used to run the harness.
osgi.director.jar	The harness code, a.k.a. the Director

Table 1-1 Contents of the osgi.tck.zip

osgi.jar	OSGi specification JAR
osgi.min.jar	Minimum services used to run the Director
osgi.target.jar	Needs to be installed on the target device. Will contact the director and execute the tests.
target	Directory to install the target in. It is advised to copy this directory and adapt the files to the implementation under test.
build.xml	Example build file that can run the tests on the target. The example is written for the OSGi RI. This RI is licensed separately. This build file has the same targets as the build file in the director directory.
keystore	Contains a test certificate for signed bundles. Used in diverse tests. The used certificate alias is test and the passwords are testtest.
osgi.properties	Properties for the target framework
policy	Security policy file

1.3 Actual TCKs

1.4 Core TCK

The Core TCK is always included in any other additional TCK. The core adds the following files:

Table 1-2 Contents of the osgi.core.ri-tck.zip

osgi.core.jar	Specification JAR, only contains the framework and system service APIs
osgi.core.tests.jar	All tests for the core. This JAR will automatically install all contained tests. Is intended to run on the Director. This file can be installed on the Director GUI.
osgi.core.impls.jar	Reference Implementation. Intended to be used to verify that the setup is correct.
director	
core.script	Script file to run the core tests from run-core in the Director
doc	
osgi.core.xml	Coverage report. This report can be opened in modern browsers. They will use an XML stylesheet on bundles.osgi.org.

The Core RI requires the properties defined in table Table 1-3, "Core RI Properties," on page 3.

The ant target to run the core TCK in the director and target directories is:

```
run-core
```

Table 1-3 Core RI Properties

`scr.bundle.name`	Name of the bundle implementing the declarative services
`eclipse.ee.install.verify`	Verifies the minimum execution environment. Must be true for the tests to succeed.
`osgi.support.signature.verify`	Verify signatures, must be true to succeed
`osgi.framework.keystore`	URL to the Keystore used by the Framework

1.5 Cmpn TCK

The compendium TCK adds the following files:

Table 1-4 Contents of the osgi.cmpn.ri-tck.zip

osgi.cmpn.jar	Specification JAR, only contains the framework and system service APIs
osgi.cmpn.tests.jar	All tests for the compendium. This JAR will automatically install all contained tests. Is intended to run on the Director. This file can be installed on the Director GUI.
osgi.cmpn.impls.jar	Reference Implementation. Intended to be used to verify that the setup is correct.
director	
cmpn.script	Script file to run the compendium tests from run-cmpn in the Director
doc	
osgi.cmpn.xml	Coverage report. This report can be opened in modern browsers. They will use an XML stylesheet on bundles.osgi.org.

The ant target to run the core TCK in the director and target directories is:

```
run-cmpn
```

1.6 Mobile TCK

The Mobile TCK includes the core TCK and adds the following files:

Table 1-5 Contents of the osgi.mobile.ri-tck.zip

osgi.mobile.jar	Specification JAR, only contains the framework and system service APIs
osgi.mobile.tests.jar	All tests for the mobile specification. This JAR will automatically install all contained tests. Is intended to run on the Director. This file can be installed on the Director GUI.
osgi.mobile.impls.jar	Reference Implementation. Intended to be used to verify that the setup is correct.
director	
mobile.script	Script file to run the mobile tests from run-core in the Director
doc	

Table 1-5 Contents of the osgi.mobile.ri-tck.zip

osgi.mobile.xml	Coverage report from static analysis
delivered	A directory with files that are required by the deployment tests. This directory *must* be available on the target. Its location is set by the osgi.properties in the target directory.
doc	
osgi.mobile.xml	Coverage report. This report can be opened in modern browsers. They will use an XML stylesheet on bundles.osgi.org.

There are a number of Mobile TCK specific properties that must be set. These properties are actually set in the TCK harness osgi.properties files. However, these properties are bound to the OSGi reference implementation. The delivered JAR files use these properties to create the appropriate implementation. This is not part of the OSGi specification and future version may choose this differently.

Table 1-6 Mobile RI Properties

`org.osgi.vendor.application`	The Implementation package for the org.osgi.service.application specification. ### is this still needed?
`org.osgi.vendor.application.ApplicationDescriptor`	The implementation class for the Application Descriptor
`org.osgi.vendor.application.ApplicationHandle`	The implementation class for the Application Handle
`osgi.dev`	??
`org.osgi.vendor.deploymentadmin`	Implementation package for the org.osgi.service.deploymentadmin specification.
`org.osgi.vendor.mobile.UserPromptCondition`	Implementation class for the UserPromptCondition
`org.osgi.impl.service.deploymentadmin.deliveredarea`	Points to the directory that is included in the ZIP file named delivered. It contains deployment packages necessary for the deployment admin test.
`org.osgi.impl.service.deploymentadmin.userprompt`	true means the implementation of the UserPromptCondition must do a real user prompt, otherwise it is assumed to run in a regression test and should not consult the user.
`org.osgi.impl.service.deploymentadmin.keystore.file`	Place where the keystore is located
`org.osgi.impl.service.dmt.DigestDelegate`	Class that implements the Digest Delegate

The ant build target for the director and target directories is:

```
run-mobile
```

1.7 Running the TCK against the RI

The following directions assume that you have "java" and "ant" in your PATH variable. I.e., when you type "java", it will execute the java VM. The example is tested against Java 1.4 and Java 1.5 and ant 1.6.

There are two directories for running the tests. The first directory is the target directory which contains an ant build.xml script that runs the target. The target is the device under test. The other directory is the director directory that has a build.xml to run the TCK in the director. The ant scripts must be started in different shells but the `osgi.properties` assume the same that they run on the same machine. The target should be started before the director.

In the director directory, please fill in the appropriate properties for your environment. ###

The following targets are supported in the director and target build:

- *director* – Run the GUI
- *run-core* – Core Framework and Framework services
- *run-cmpn* — Compendium services
- *run-mobile* – Mobile services

The script can fail if the appropriate TCK is not installed.

The director script generates a file called "run-xxxx.xml" in the director directory. This file must be mailed to the certification authority for compliance.

1.8 Sample Run

Most of the details are abstracted in the ant files. In principle a framework must be started in the target directory as well as the director directory. Both directories support the run-xxxx target.

First the target framework must be started. Start a shell (both a windows shell or a Lunix/Unix shell should work) and cd to the ./target directory. Run `ant run-core` in this directory.

```
$ cd ./target
$ ant
Buildfile: build.xml
run-core:
-run:
     [echo] Testing file:../osgi.core.impls.jar
     [java] Bundle id is 9435
     [java] Bundle id is 9436
     [java] Bundle id is 9437
     [java] No files specified  in Group-Load manifest tag:
```

The default target is to only run the core RI (which you will need a license for). The last line of the output contains a warning that can be ignored. The target will run until it is quit (possibly by the director) or when no activity has taken place for a certain time. This timeout prevents target frameworks from running forever and polluting the system.

Now a director framework must be started. Start a new shell and cd to the ./ director directory. Also run ant run-core.

```
$ cd ../director/
$ ant
Buildfile: build.xml

director:
 [java] Bundle id is 2215
 [java] Bundle id is 2216
 [java] Bundle id is 2217
 [java] osgi> Discovery starts.
```

It is possible that there is a warning

```
 [java] DatagramSockeet for target discovery already
      in use (2001)
```

This warning can be ignored, it means that there is another director running, likely the OSGi TCK Eclipse plugin, which is available to members. The default configuration of the properties always assumes the localhost as where the target is running. You can change this property. ###

The build file will run a script in the directory. There are scripts for each TCK. The scripts normally delay for a number of seconds to finish installation and then sequence the test cases. The output can look like this for the target:

```
[java] Target: Accepting connection from /127.0.0.1
[java] Target: Installing test.framework.classloading_TBC
[java] Log: [begin=testAllServiceListener001]
[java] Log: #Checking if an event is delivered for a bundle
       which imports a assignable service interface
[java] Log: #Checking if an event is delivered for a bundle
       which imports a non-assignable service interface
[java] Log: [end=testAllServiceListener001]
...
[java] Log: [end=testBundleContextGetAllServiceReferences001]
```

The output of the Director shell looks like:

```
    [java] Listening on port 2201 ...
    [java] ----------------------------------------------
    [java] test.framework.classloading
    [java] ----------------------------------------------
    [java] test.framework.classpath
    [java] ----------------------------------------------
    [java] test.framework.div
    [java] ----------------------------------------------
    [java] test.framework.dynpkgimport
    [java] ----------------------------------------------
    [java] test.framework.filter
```

When the Director is finished, you can find a run-xxxx.xml file in the director directory. This file can be viewed in a modern browser. It is formatted with XSL; it gets its style sheet from bundles.osgi.org. This works only when connected.

2 Adapting To Your Framework

The normal way to run the compliance program is to create a setup that is similar to the RI. This means your implementation of the Framework must be started running the `osgi.target.jar` file. This will automaticaly allow the director to control it. This implementation is carefully written to rely on as little Java and OSGi as possible.

The Director harness has many options that can be found in its manual to adapt to your environment. ###

1.1 Certificate Handling

One point that must be taken care of is certificate handling. The RI uses a Java keystore that is included in the target directory. However, certificate handling is not standardized in OSGi and framework vendors must therefore add the "test" certificate to their own certificate repository for these tests to pass.

Usually it will take a number of rounds before the TCK runs correctly on your Framework. Running all the tests for each correction can be cumbersome. Thre Director harness is quite easy to use in GUI mode. Running `ant director` script in the director directory will start it in GUI mode without any test cases. The manage button allows you to install a single test case and run it singularly. See the manual for more information how the Director can be used to debug.

2 Test Harness

The Director is a test harness for OSGi TCKs. It runs on an OSGi Framework and controls a target implementation framework via the network. The Director can be used interactively in GUI mode or via a script for regression testing. The tool has a number of options that can be set using properties.

The output of a test run is colleced in an XML file. The XML file can be mailed, posted, as XML or HTML, or stored in a standard place.

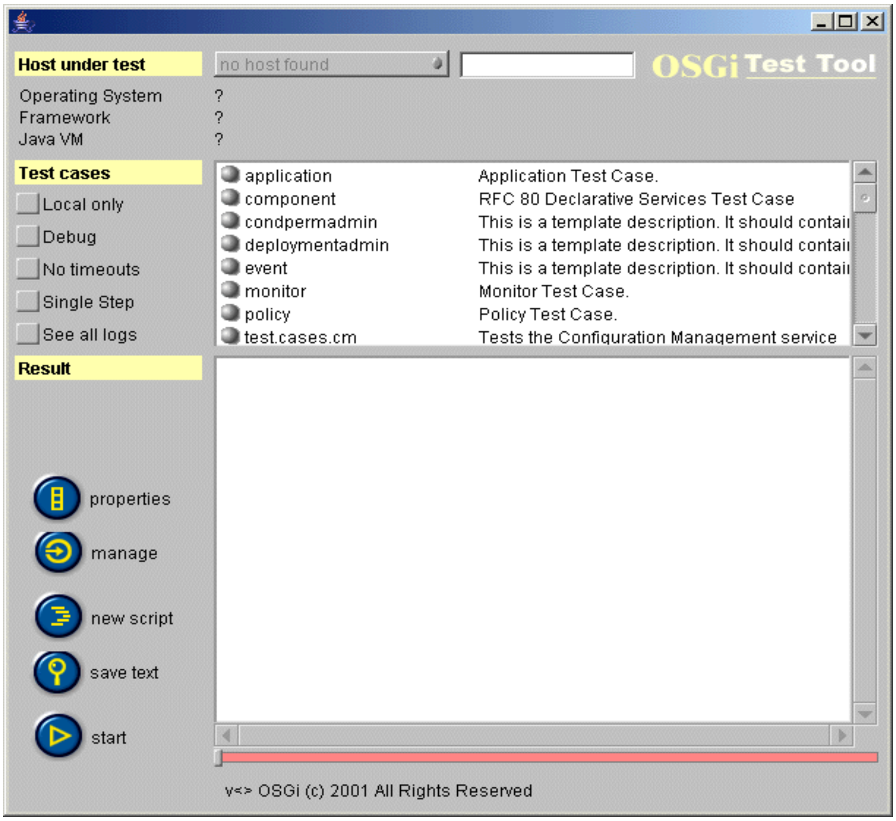
The complete harness is packed with the compliance ZIP file. A limited form of the Director is also available as an Eclipse plugin.

The target is a minimal bundle that announces itself on the network so that Director harnesses can find it and that can collaborate with the Director to run a TCK.

1.1 The GUI

The UI is written with IFC. This is a different UI than Windows or Swing but should be quite straightforward to use for most computer users.

Figure 1-1 *OSGi Test Tool Director GUI*



application	Application Test Case.
component	RFC 80 Declarative Services Test Case
condpermadmin	This is a template description. It should contai
deploymentadmin	This is a template description. It should contai
event	This is a template description. It should contai
monitor	Monitor Test Case.
policy	Policy Test Case.
test.cases.cm	Tests the Configuration Management service

v<> OSGi (c) 2001 All Rights Reserved

1.1.1 Selecting a target

Goto the menu on the top of the page. The menu shows all the active targets that are available on your network. Select the one needed. The UI will remember your choice, even when targets go away or come.

Sometimes a network contains multiple targets and you are only interested in the local target. In that case, there is a checkbox "local only" that will help. It will remove all targets from other computers than your local computer.

1.1.2 Selecting a "challenged" target

Some OSGi environments are not capable of broadcasting the announcement messages or they are on another network where broadcasts messages are not forwarded. There solutions to this. First, on the top right site there is a text field. Typing the name of the host (or IP number) and hitting enter/return will create a target to that host. The other solution is to set the broadcast address of the target to a single address of your host computer. This is done by setting the system property: "org.osgi.test.broadcast" to the destination system (or network).

1.1.3 Selecting test cases

Select the testcases you want to execute. No selected testcases is the same as selecting all. Multiple testcases can be selected with selecting the first and then keep the mouse pressed and dragging it to include more. Unfortunately, the standard select + shift/control does not work.

1.1.4 Starting a test run

A test run is started with pressing the start key. The key will become red to indicate a run is in progress. The UI, however, remains active and responds to other presses. You can still scroll through the testcases and messages. Pressing the same button again (now called stop) will attempt to stop the run. This can cause many exceptions on the target because it requires the cooperation of the test case itself. It is therefore also not always direct.

1.1.5 Setting the Test Properties

The UI allows the selection of a properties file that is downloaded to the target. The "properties" button opens a file chooser that selects the appropriate properties file. This function will only set the system property org.osgi.test.properties.file to the file path. For each test run, this will cause the Director to download the given properties to the target.

1.1.6 Debugging a test run

There are two support options for debugging a testcase. First option is to select the debug checkbox. This will log more information. The second option is to select the single step checkbox. Every time the test case sends log information, the start/stop button will change and the progress halts. This allows you to inspect the target framework and look at the bundles state. Time outs might step in if the session takes too long. In that case, use the disable timeouts checkbox option.

1.1.7 How to see the log

All information that is logged from the target is compared to a reference log. Due to the size of the information it is not useful to see it when everything is passing. However, during debugging it is useful to see this information. First, it is always displayed on the console of the target framework. Second, checking the ' 'See all logs' checkbox will show the result in the message window.

1.1.8 Progress

Some test cases can take a lot of time to execute. Well written test cases should indicate progress (they can do this with the TestRun object they receive) by sending the percentage of work done. Alas, not all test cases are well written. Therefore, the UI will show the progress but also a '.' for each received log entry.

1.1.9 Managing the director framework

The director framework can be managed when the 'manage' button is pressed. A small window opens that shows the state of all bundles in the framework (this is the director, not the target!). There are buttons to start/stop/update/uninstall and install bundles. This can be very useful sometimes. For example, it allows one to install the test bundles.

The text line at the bottom of the window is a filter of the bundle symbolic name. Only bundles starting with the given prefix are visible. This prefix is default test. for test cases, however, this can potentially hide newly installed bundles when the bundle symbolic name does not start with test. Just remove the prefix and all bundles will be visible.

1.1.10 Running a script

Scripts are XML files with the script DTD. A script can be selected from the UI with the 'script' button. You are asked to open a file. A reference to this file is placed in the Framework's registry as if it was a normal testcase. This allows scripts to be executed as normal testcases.

Scripts cannot be removed.

1.1.11 Saving the messages content

The message window can be saved with the 'save text' button, or it can be copied to the clipboard. Copying is achieved by selecting the text in the message window and typing ˆC (Control key + C key at the same time).

1.1.12 Seeing the test result

After a test run, the message window and the status line will show where the results can be found. The easiest way to direct your browser is to copy the URL (ˆC) and paste it in the appropriate place of your favorite browser.

1.2 Director Properties

The Director is normally instructed with a number of System properties. These are explained in the following table:

Table 1-7 Director Properties

Property	Default	Description
org.osgi.test.batch.target	localhost:3191	This the host and port of the target where a script will be executed on.
org.osgi.test.batch.script		A URL (mandatory, no filename alone) of a script. The script should contain an XML script.

Table 1-7 Director Properties

org.osgi.test.batch.out	<date & time >.xml	A URL or a file name (if the name does not have ':' between the 3 and 10th character it is assumed to be a file name) of a script. The URL is used in the output mode. There exist many URLs that work that way: mailto:cpeg@mail.osgi.org?sub-ject=Test+Result Will send a mail, on some systems this requires the Java mail.host System property to be set! ftp://cpeg:password@membercvs.osgi.org/testresult.xml Will only work on systems that implement output on the FTP stream handler. http://membercvs.osgi.org/servlet/testre-sult.xml - Will do a POST on the given server result.xml - Assume normal file name result.html - URLs or filenames ending in HTML are stored as HTML and not the default XML
org.osgi.test.batch.ui	script ? false : true	Defines if there is a UI. If the property is set to 'false', the UI is not started. The default tries to be intelligent. If a script is run, there will be no UI. Without a script there is a UI. The UI fol-lows the script run if it is open and can be used for debugging.
org.osgi.test.batch.quittarget	'false'	This property defines if the target should be exited at the end of the script test run.
org.osgi.test.compliance.applicant		This property can identify the applicant for the compliance program. This is expected to become an identification number but is not yet defined.
org.osgi.test.compliance.program		Specifies the compliance program this test run was done for. Needs to be specified further
org.osgi.test.compliance.campaign		The campaign id for the compliance program. Needs to be further specified.
org.osgi.test.properties.file		The value of this property is used a file name of a properties file. If this property is set and the given file exists, it is loaded into a property object and send to the target. The target will use these properties to override the target's system properties.

1.2.1 Target Properties

The following properties are for the target framework.

Table 1-8 Target Properties

org.osgi.test.target.port	3191	This property is intended for the target framework. Normally, the target framework listens at 3191 and this property can change that. The target informs the director of this port through a broadcast message so no property is necessary for the director.
org.osgi.test.broadcast	255.255.255.255	The target uses this property to broadcast the announcement package. The default works fine and works for most systems. It should be used when the broadcast should be limited to one network, or it is limited to a single host. For a host the actual IP can be used, for a network, network broadcast address should be used which is for example 172.16.255.255. On some systems the broadcast address is 0.0.0.0 so then this property can also be used.
org.osgi.test.seewait	false	Testing is not easy and writing test cases is harder. This property sets a simple animation when the target is waiting for input, indicating that it is ready to listen to the director. There are two animations: • |/-\ for normal operation • ?!?! when there is an exception (usually security) during the sending of the announcement.
org.osgi.test.testcase. timeout	60000	Timeout used for communication between director and target for testcase communication when the testcase is based on the classes in org.osgi.test.cases.util
org.osgi.test.testcase. watchdog	10	If this property is set, the target will exit after 30s * this value seconds. Deault, after 5 minutes the watchdog will exit. This is intended to be used when the test is run in a batch file every couple of hours and there are test cases that make the target hang.

1.3 The Script language

The director implements a simple script language for controlling the test cases. Though it is not a script language that will have many afficianados, it makes it very easy to parse. A script can look like the following example:

```
<script name="compliance" version="1.0">
   <delay duration="10" type="sec"/>
   <istart>
      <url>http://membercvs.osgi.org/project/sp-r3/bundles/
org.osgi.test.cases.useradmin.tc1</url>
```

```
        <url>http://membercvs.osgi.org/project/sp-r3/bundles/
org.osgi.test.cases.useradmin.tc2</url>
        <url>http://membercvs.osgi.org/project/sp-r3/bundles/
org.osgi.test.cases.useradmin.tc3</url>
    </istart>
    <testcase>useradmin.tc2</testcase>
    <waitfor max="10000"
        count="1" poll="500">
    (&(objectclass=org.osgi.test.TestCase)
        (name=permissions.tc))</waitfor>
    <updateFramework/>
    <reboot status="0"/>
</script>
```

In detail:

Construct	Attributes	Description
<script>		This is the outer marker of the script.
	name	Identifying name of the script
<waitfor>		Wait for a number of services to appear in the service registry for a maximum time. The value of the tag is a filter expression for the Framework.
	max='10000'	Maximum time to wait in ms
	count='1'	Number of services that should come out of getServiceReferences with the associated filter.
	poll='500'	Number o ms between checks of the registry.
<delay>		Wait for a number of seconds.
	duration='10'	Time to wait, unit depends on type.
	type='sec'	Unit, can be week, hour, day, minute, sec, ms
<istart>		Install and start a bundle from a url.
<url>		URL for a bundle to be started.
<testcase>		Run a testcase. The body of the tag is trimmed of spaces and looked up in the registry. If such a test case exists, it is executed in a new session.
<properties>	location	Set the `org.osgi.test.properties.file` System property so that the properties file is downloaded for every test case and its content is merged with the Target's system properties.
<updateFramework>		This function will update the framework on the target. It will call Bundle.update() on the bundle with ID=0. This will of course terminate the session, but if a delay is inserted, the next line of the script could run on the same target after it is restarted.

<reboot>		This function will exit the framework on the target. It will call Bundle.stop() on the bundle with ID=0. This will of course terminate the session. However, if the target framework is restarted with a script that starts it again, a small delay can be used to continue the test run. Note that the exit status can be used by the script on the target machine to stop the framework or restart.
	status='0'	Set the exit status of the target framework.

1.4 The Command Line

The director supports the CommandProvider interface. It will register a CommandProvider with the Framework when it starts. The `org.osgi.tools.console.jar` bundle can be run on the director Framework at the same time. This console will listen on port 2011 (or a next port if it is busy) for a telnet session. The director adds the following commands to the console:

Table 1-9

Command	Options	Description
startrun		Starts a test run. This command must precede any testcase or exec commands.
	-host localhost	Defines the host where the target runs
	-port 3191	The port where the target runs.
	-forever	Will not time out on messages.
	-debugging	Generate extra debugging information
exec	<script name>	Execute a script. This can only be done after a startrun is given.
exec	<script name>	Execute a script. This can only be done after a startrun is given.
	-output <file name>	Places the output in a file.
testcase	<testcase name>	This will lookup the test case with the given name and executes it in a new session.
stoprun		Stop the current run and return the name of the result file. This is an XML file stored in the history directory of the bundle. It can also be found on the web on http://host:port/test/director

2 References

[1] *Bradner, S., Key words for use in RFCs to Indicate Requirement Levels* http://www.ietf.org/rfc/rfc2119.txt, March 1997.

[2] *OSGi Service Gateway Specification 1.0, May 2000* http://www.osgi.org/resources/spec_download.asp

[3] *OSGi Service Platform, Release 2, October 2001*
http://www.osgi.org/resources/spec_download.asp

[4] *OSGi Service Platform, Release 3, March 2003*
http://www.osgi.org/resources/spec_download.asp

[5] *OSGi Service Platform, Release 3, March 2003*
http://www.osgi.org/resources/spec_download.asp

End Of Document