



Java™ Device Test Framework Getting Started Guide

Version 2.4

Sun Microsystems, Inc.
www.sun.com

May 2009

Submit comments about this document by clicking the Feedback[+] link at: <http://docs.sun.com>

Copyright 1996-2009 Sun Microsystems, Inc. All Rights Reserved.

DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.

This code is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 only, as published by the Free Software Foundation. Sun designates this particular file as subject to the "Classpath" exception as provided by Sun in the LICENSE file that accompanied this code.

This code is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License version 2 for more details (a copy is included in the LICENSE file that accompanied this code).

You should have received a copy of the GNU General Public License version 2 along with this work; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.

Please contact Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA or visit www.sun.com if you need additional information or have any questions.

Contents

Preface	xi
1. Concepts	1
Test Framework	1
Test Device	4
Test	4
Test Types	5
Runtime Tests	5
Benchmark Tests	6
Over-the-Air Provisioning Tests	6
Central Installation	6
Administrator and Tester Harnesses	7
Relay	7
Work Directory, Configuration, and Template	7
2. Installation	9
Installation Options	9
Individual Installation	9
Prerequisites	10
Verify Prerequisites	11

Prepare Configurer	12
Run Configurer	13
Edit Application Server Permissions	14
Deploy Relay	15
Unzip Developer's Kit	16
Install NetBeans Platform Plugin	16
Install Basic Test Packs	17
Group Installation	20
Central Installation and Relay Installation	20
System Requirements	21
Recommended Components	23
Verify Prerequisites	24
Prepare Configurer	24
Run Configurer	26
Edit Application Server Permissions	26
Deploy Relay	26
Install Basic Test Packs	27
Tester Installation	27
Prerequisites	27
Verify Prerequisites	28
Prepare Configurer	28
Run Configurer	30
Post-installation Reconfiguration	30
3. Launching the Administrator Harness	31
4. Installing a Test Pack	33
5. Connecting Test Devices	35
Test Device Requirements	35

Test Device Connection Options	36
Test Bundle Transfer	36
HTTP Bundle Transfer	36
Local Link Bundle Transfer	37
Test Result Disposition	38
Specifying the Transmission of Bundles and Results	38
6. Running Tests	43
Preparation	43
Selecting Tests	44
Configuring the Test Run	44
Configuring Test Packs	45
Configuring Test Cases	45
Setting the View Filter	45
Starting the Test Run	46
7. Inspecting Results	47
8. Creating Reports	51
9. Sharing Data	53
Sharing Results	53
Sharing Templates	54
10. Writing Tests	57
11. Uninstalling	59
Index	61

Figures

FIGURE 1-1	Test Framework Main Components and Interactions	2
FIGURE 1-2	Administrator Harness	3
FIGURE 1-3	Simple Test Pack Under Development	4
FIGURE 1-4	Test Hierarchy in Harness Test Tree	5
FIGURE 1-5	Configuration Editor	8
FIGURE 2-1	Administrator Harness After Installation	18
FIGURE 2-2	Test Pack Installer First Screen	19
FIGURE 2-3	Test Tree After Basic Test Pack Installation	20
FIGURE 3-1	Administrator Harness Initial Display	32
FIGURE 4-1	Installing a Test Pack Help	34
FIGURE 5-1	Test Bundle Transfer Options - HTTP	37
FIGURE 5-2	Test Bundle Transfer - Local Link	37
FIGURE 5-3	Test Result Disposition Options	38
FIGURE 7-1	Package Test Results	48
FIGURE 7-2	Failed Test Results	49
FIGURE 8-1	Create Report Dialog	52
FIGURE 9-1	Report Converter Wizard First Screen	54

Tables

TABLE 2-1 Relay and Central Installation Host Requirements 21

Preface

The *Java™ Device Test Framework Getting Started Guide* is the first document new test framework users should read. It covers installation and essential use topics.

Using UNIX Commands

This document might not contain information about basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices. Refer to the following for this information:

- Software documentation that you received with your system
- Solaris™ Operating System documentation, which is at:

<http://docs.sun.com>

Shell Prompts

Shell	Prompt
C shell	<i>machine-name%</i>
C shell superuser	<i>machine-name#</i>
Bourne shell and Korn shell	\$
Bourne shell and Korn shell superuser	#

Related Documentation

The following table lists the documentation for this product.

Title	Format	Location
Harness help	Help	Administrator harness Help menu
NetBeans software plugin help	Help	NetBeans software Help menu
<i>Release Notes</i>	HTML	Install directory - ReleaseNotes.html
<i>Documentation Overview</i>	HTML	Install directory - index.html
Test Framework Web Site	HTML	https://jdtf.dev.java.net

The following table lists the Java Device Test Suite documentation that is related to the test framework. See the *Documentation Overview* for more information.

Concepts

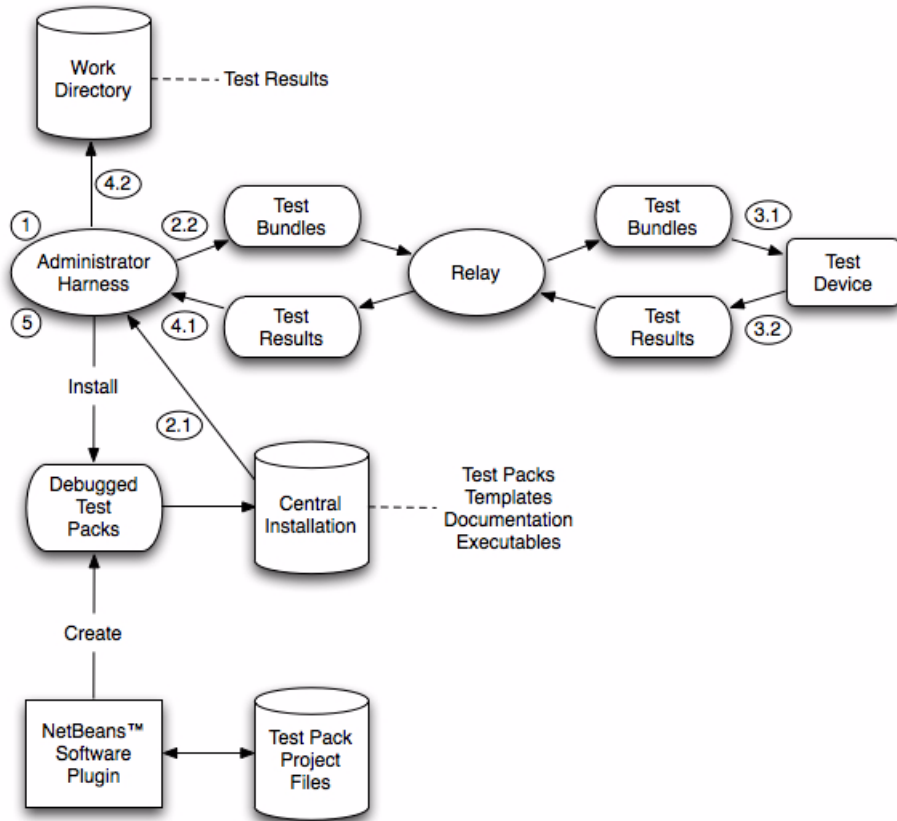
This chapter describes these concepts:

- [Test Framework](#)
- [Test Device](#)
- [Test](#)
- [Central Installation](#)
- [Administrator and Tester Harnesses](#)
- [Relay](#)
- [Work Directory, Configuration, and Template](#)

Test Framework

The Java Device Test Framework (test framework) is an environment for running and writing tests that exercise Java technology installed in test devices (typically mobile phones). [FIGURE 1-1](#) shows the main components and how they interact.

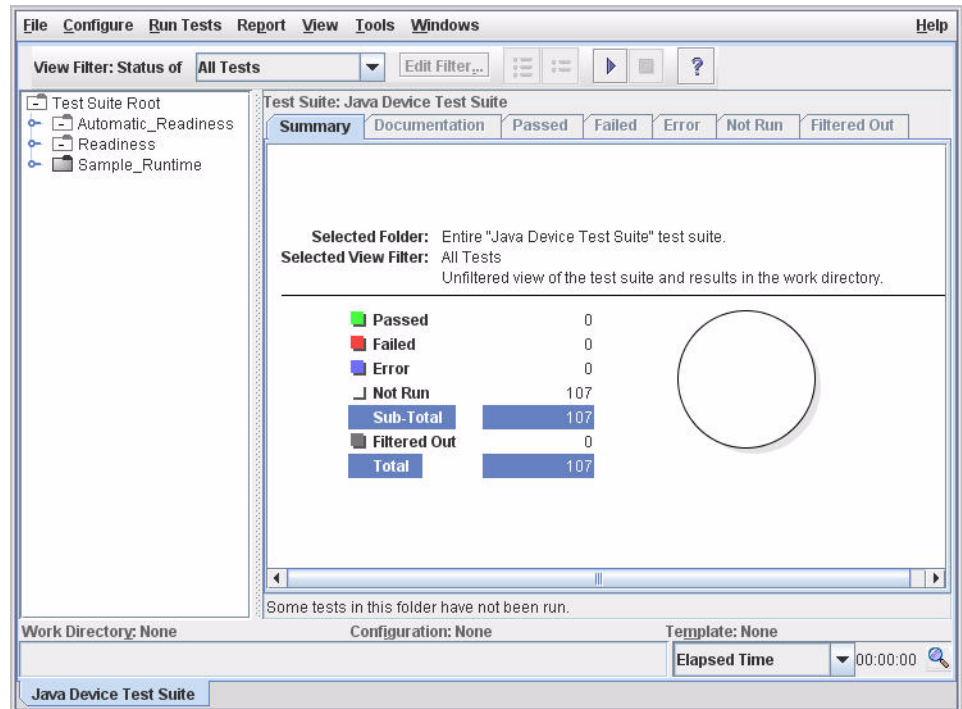
FIGURE 1-1 Test Framework Main Components and Interactions



A simple use case proceeds as follows.

1. A user launches the administrator harness, selects and configures some tests, and clicks the harness's Start button. [FIGURE 1-2](#) shows the administrator harness.

FIGURE 1-2 Administrator Harness



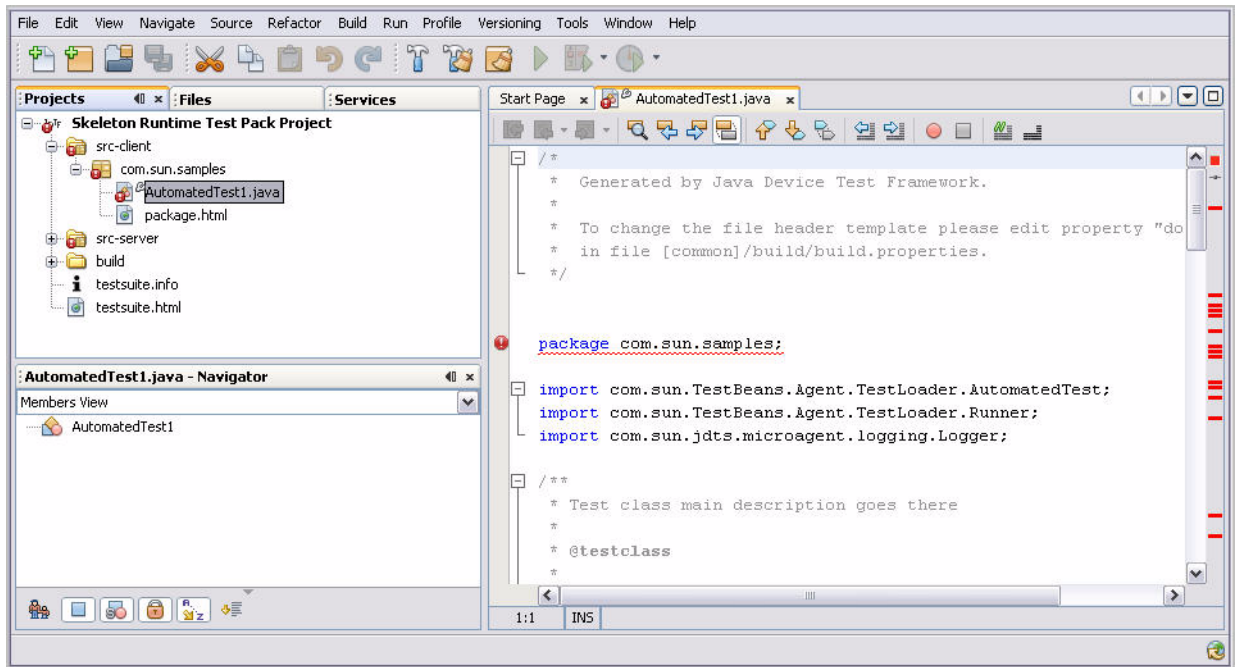
2. The harness reads tests from the Central Installation, packages them into bundles (MIDlet suites), and sends them by HTTP to the Relay.

The Relay is an administrative convenience when a firewall stands between the test device and the test framework.

3. The test device fetches a test bundle from the Relay by HTTP, executes the tests, and returns results (status, logs) to the Relay.
4. The harness fetches the results from the Relay and stores them in the work directory.
5. When it receives the results from the final test bundle, the harness ends the test run, and the user can inspect results, create reports, start another test run, and so on.

Test development is orthogonal to running tests, though a developer runs tests to debug them. The main unit of test development is called a test pack, and a test pack is created or modified with a test framework plugin to the NetBeans™ integrated development environment. When a new or modified test pack is ready for deployment, the developer packages it into a zip file, which can be installed by the administrator into any Central Installation. FIGURE 1-3 shows a test pack under development.

FIGURE 1-3 Simple Test Pack Under Development



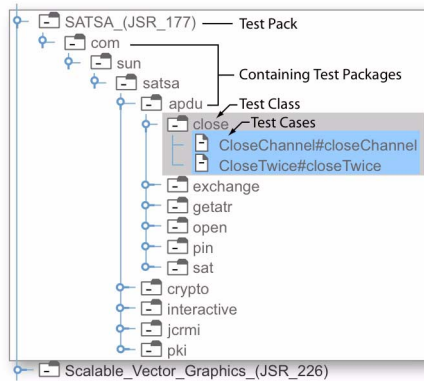
Test Device

A test device is a mobile phone that includes an implementation of Java Platform Micro Edition technology. Using the test framework to test such a device can reveal problems that prevent applications from running well on it. Test device requirements and connection options are described in [“Connecting Test Devices” on page 35](#).

Test

To users, tests (more formally test cases) appear as the lowest level (leaf) nodes in the harness’s test tree. The higher level nodes are called test classes, test packages, and test packs. [FIGURE 1-4](#) shows an example of the hierarchy.

FIGURE 1-4 Test Hierarchy in Harness Test Tree



A test has a status, which is Not Run, Error, Failed, or Passed.

To a developer, a test case is a method in a test class.

Test Types

There are three main types of test, each of which has subtypes. A test pack contains tests of a single type.

Runtime Tests

Runtime tests execute on the test device under the control of a Java Device Test Framework component called the MicroAgent. The MicroAgent is usually included in the test bundles that users download and install in devices being tested. It is also possible to install an agent in some test devices' read-only memory. A test bundle is a collection of tests plus the MicroAgent packaged as a MIDlet suite, a unit that a test device's application management system (AMS) must be able to download, install, and execute.

There are two main kinds of runtime tests, automated and interactive. An automated runtime test determines its result (passed or failed) and returns the result to the MicroAgent, which returns it to the harness via the Relay.

An interactive runtime test relies on you to determine if the test passed or failed. When an interactive test begins to run, the harness displays instructions for you to operate the device and inspect the device's response (for example, what it displays).

The instruction window includes Passed and Failed buttons. If the device behaves as the test instructions say it should, you click the Passed button. Otherwise, you click the Failed button.

Benchmark Tests

From a tester's perspective, benchmark tests are nearly identical to runtime tests. Benchmark tests return performance measurements called metrics, that the tests compute and the harness displays. Benchmark tests can also pass or fail. The pass or fail decision is made by comparing the test device's performance with the performance of a reference device on the same test. The reference device's performance constitutes a threshold that the test device must meet or exceed for the test to pass.

Over-the-Air Provisioning Tests

Over-the-air (OTA) provisioning tests verify the quality of a device's over-the-air application provisioning implementation. This includes obtaining, installing, and removing applications (MIDlet suites), and enforcing security requirements.

Unlike runtime and benchmark tests, OTA provisioning tests do not run on the test device. They run on an emulated provisioning server that is implemented as a servlet. Each OTA test has an associated application (a MIDlet suite) that you download from the provisioning server and install and launch on the test device.

Central Installation

The Central Installation is a directory containing shareable test framework components, such as installed test packs, and the administrator harness. Sharing the Central Installation is optional. Each user can have a personal Central Installation, or one Central installation can be shared over a local network by multiple users. In this guide, the directory containing the Central Installation is referred to as *CentralInstallDir*.

Administrator and Tester Harnesses

There are two versions of the test harness. The administrator harness is included in the Central Installation. An administrator or individual user can use this harness to create and maintain shareable resources, such as test packs and templates and to run tests. The administrator harness is a superset of the tester harness. A developer can use the administrator harness to test tests.

The tester harness is somewhat simpler than the administrator harness. It is intended for running tests using shared resources, rather than administering those shared resources. A group installation consists of a shared Central Installation, a shared Relay, and one or more tester harnesses. The tester harnesses use a local network to access resources stored in the Central Installation, and HTTP to communicate with the Relay. Unless sharing is important, there is no reason to install the tester harness.

Relay

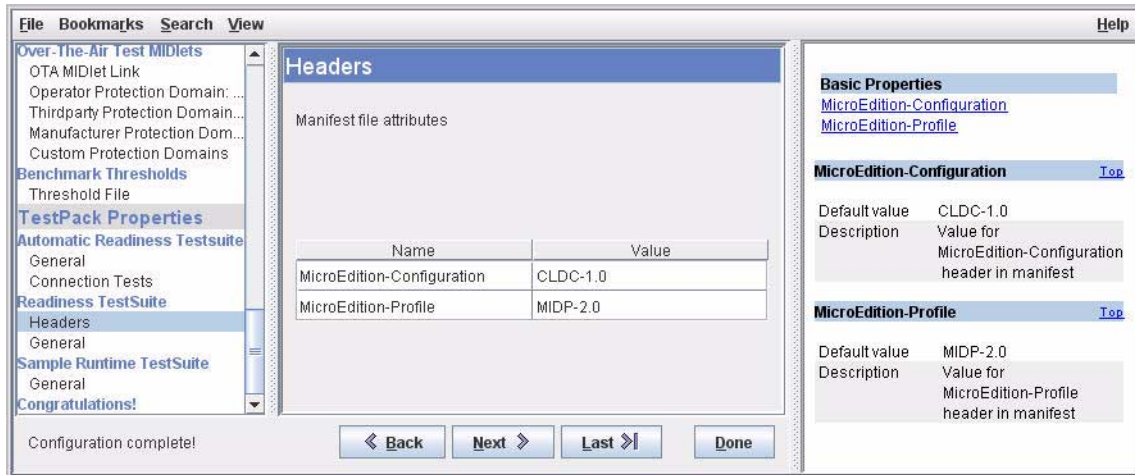
The Relay is a Sun Java System Application Server component that intermediates between harnesses and test devices. The Relay's main task is to respond to HTTP requests from devices. Devices obtain test bundles from the Relay and return test results to the Relay. Harnesses send test bundles to the Relay and receive results from the Relay. A Relay can be shared or dedicated to one user. A shared Relay can simplify administration by being installed in the "DMZ" between an organization's inner and outer firewalls. All Relay-device communication can then be conducted through a small number of ports in the firewall.

Work Directory, Configuration, and Template

To run a test, you must have created a work directory into which the harness stores test results. You can have as many work directories as you wish, for example, creating one work directory per device or per device version. A work directory has a configuration, which specifies how to run tests, and how each test pack is customized for a test device. For example, a configuration can specify that only tests

of a particular severity are to be run, and that a hypothetical test pack property called `X_Origin = 0.0`. You create and edit configurations with the harness's configuration editor. [FIGURE 1-5](#) shows the configuration editor.

FIGURE 1-5 Configuration Editor



Configurations are based on templates. A template is a shareable resource and can therefore only be created or changed with the administrator harness, for example, with `Configure > New Template`. The template editor looks much like the configuration editor shown in [FIGURE 1-5](#). Many configurations can be derived from the same template. If you change a value in a template, the change flows to the configurations automatically.

If you work with families of related devices that share many characteristics, you can simplify template management by creating hierarchical families of templates whose branches correspond to test device branches. You can set common values high in the hierarchy and synchronize (propagate) changes in those values to the lower-level templates, those that correspond to actual devices. Use the administrator harness `Configure > Templates Manager` command.

Installation

This chapter covers these topics:

- [Installation Options](#)
- [Individual Installation](#)
- [Group Installation](#)
- [Post-installation Reconfiguration](#)

Installation Options

The test framework can be installed for individuals or groups.

- An individual installation is created on one computer. It can be used by one person to run tests and to develop tests.
- A group installation is created on multiple computers, enabling multiple users to run tests without replicating the Central Installation and the Relay. A group installation is not suitable for development because tests under development must be isolated (in individual installations).

Individual Installation

Choose this installation if you plan to run tests by yourself on one computer, or if you plan to write tests.

Prerequisites

The test framework runs on the Solaris 10 operating system for the SPARC® processor, or Windows XP with Service Pack 2.

Before installing the test framework, obtain and install the following software:

- Java Platform Standard Edition Developer's Kit version 1.6.0_03 (also known as JDK™ version 6.0 Update 3) or greater. You can download the JDK software from <http://java.sun.com/javase/downloads/index.jsp>.
- Apache Ant version 1.7.1 or greater. Download this version from <http://archive.apache.org/dist/ant/binaries/>. The file is `apache-ant-1.7.1-bin.zip`.
- Sun Java System Application Server Platform Edition (PE) version 8.2. Download this version from <http://java.sun.com/j2ee/1.4/download.html>. Other application servers can work but they have not been tested. An application server must implement Java Servlet API 2.3 and JavaServer Pages™ 1.2 specifications (JSR 53) or any higher versions of the specifications. This guide gives configuration instructions for the Sun Java System Application Server.
- NetBeans integrated development environment version 6.5.1 or later.

- Windows users: At <http://netbeans.org/downloads/index.html>, select the Windows 2000/XP/Vista Platform and download either the “Java” bundle or the “All” bundle.

When you install, you can omit unneeded components, such as Java Web and EE or C/C++, but you must *not* omit Java SE or Java ME.

If you already have the IDE installed with the Java SE component, you can install the Mobility plugin when you install the test framework plugin (see “[Install NetBeans Platform Plugin](#)” on page 16). The Mobility plugin is a subset of the Java ME component.

- Solaris operating system users: At <http://netbeans.org/downloads/index.html>, select the OS Independent Zip Platform and download the “Java” bundle or the “All” bundle.

When you install, you can omit unneeded components, such as Java Web and EE or C/C++, but you must *not* omit Java SE or Java ME.

- A version of the Wireless Toolkit is required for preverification of tests and is recommended for testing new tests.
 - Windows users: If you did not download the “Java” or “All” NetBeans software bundle, download and install the Wireless Toolkit for CLDC version 2.5.2 from <http://java.sun.com/products/sjwtoolkit/download.html>.

The “Java” and “All” bundles include the Wireless Toolkit. It is installed as `NBInstallDir\NetBeans 6.5.1\mobility8\WTK2.5.2\`.

- Solaris operating system users: Download and install version 2.1_01 of the J2ME Wireless Toolkit from http://java.sun.com/products/sjwtoolkit/download-2_1.html.

Version 2.1_01 of the Wireless Toolkit cannot be installed with JDK version 1.6.0_03, which is recommended for the test framework. When you install the WTK, specify a JDK version in the 1.4 or 1.5 series.

This document refers to the Wireless Toolkit installation directory as *WTKInstallDir*.

Solaris operating system users must also download and install on a Windows platform version 2.5.2 of the Sun Java Wireless Toolkit for CLDC from <http://java.sun.com/products/sjwtoolkit/download.html>. This guide refers to the Windows installation directory as *winWTKInstallDir*.

Copy *winWTKInstallDir/lib/jsr082.jar* to *WTKInstallDir/lib/*.

Solaris platform developers can use JAR files created on the Windows platform because the files are platform independent.

For testing tests, you can use additional or different emulators or real devices. The harness can automatically launch a compatible emulator and run tests on it. Solaris operating system users must install the Wireless Toolkit emulator on a remote Linux host. The test framework's NetBeans software online help has instructions.

Verify Prerequisites

You can verify that the required software versions are installed with these commands:

- `java -version`
- `ant -version`
- Start the Sun Java System Application Server, then load <http://appServerHost:appServerPort/appContext> (for example, <http://localhost:8080/JdtsServer>) into a web browser.

To start the Application Server in a Windows environment, choose Start > Programs > Sun Microsystems > Application Server PE > Start Default Server or execute this command:

```
AppServerInstallDir\bin\asadmin start-domain domain1
```

To start the Application Server in a Solaris operating system, execute this command:

```
AppServerInstallDir/bin/asadmin start-domain domain1
```

Prepare Configurer

An Ant script known as the configurer prepares the downloaded test framework files for use. After you have installed the test framework, you can re-run the configurer to change properties of your installation, as described in [“Post-installation Reconfiguration”](#) on page 30.

You specify configuration options in a file called `jdtfconfigurer.properties`, which is in the unzipped test framework's top directory. Open `jdtfconfigurer.properties` in a text editor. If you need a fresh copy of `jdtfconfigurer.properties` (because someone else has previously edited it), extract it from the test framework zip file.

1. Inspect and change the following lines in `jdtfconfigurer.properties` as necessary (default values shown - Windows users, note forward slashes):

- Directory to hold test framework Central Installation (administrator harness, developer's kit, and so on):

```
central.install.dir=c:/sun/jdtf-ci
```

This directory must be empty or non-existent.

- Path to Java Platform Standard Edition Developer's Kit (note forward slashes in Windows path name):

```
java.exec=c:/program files/java/jdk1.6.0_10/bin/java.exe
```

- Change this IP address to your workstation's true IP address as seen by a test device:

```
relay.server.host.device=777.777.777.777
```

Do not specify localhost or 127.0.0.1 because this address represents a test device's point of view, and the Relay runs on your workstation, not on a test device.

2. If the default values are acceptable, you do not need to edit the following properties.

- a. If the Relay's port for incoming HTTP requests from administrator and tester hosts is not the default 8080, specify the port number.**

```
relay.server.port.console=8080
```

- b. If the Relay's port for incoming HTTP requests from test devices is not the default 8080, specify the port number.**

```
relay.server.port.device=8080
```

- c. If the default path is incorrect or undesirable, specify a directory for the Relay to use for test-related storage.**

```
relay.persistent.storage.root.dir=c:/sun/jdtf-server-storage
```

This path must refer to a file system that the application server can access. In the default value, drive c refers to the drive on the application server host, which is not necessarily the c drive on the host where you will run `jdtdconfigurer`. The Relay creates the directory when it needs it.

- d. If the Relay must be restricted to particular IP ports for communicating with tests, change the default value.**

```
relay.server.port.range=any
```

You can specify individual ports, or ranges, or combinations, for example: 1001,1010-1050,1100-1140. Specify about 100 total ports.

- e. If you have installed an emulator, change the default value if it is not correct.**

```
default.emulator.home=c:/WTK2.5.2
```

- f. If you want the administrator harness to use a different name to distinguish this emulator from others you might also install, change the default value:**

```
default.emulator.alias=JDTF-EMU
```

- 3. Save and close `jdtdconfigurer.properties`.**

Run Configurer

Run the configurer by executing this command in the directory containing `jdtdconfigurer.properties`:

```
ant -f jdtdconfigurer.xml
```

The configurer creates a Central Installation at the location you specified in `central.install.dir`. The Central Installation contains these important files:

- `server/JdtsServer.war`: the Relay component which you deploy to the application server (see [“Deploy Relay” on page 15](#))
- `admin/shared/resources/localRelay/JdtsServer.war` which you can deploy to test a device that communicates by HTTP over a serial cable connected to your workstation
- `admin/run.bat` and `run.sh`: scripts that launch the administrator harness for running tests
- `admin/shared/devkit.zip`: the test developer's kit, which includes samples and API documentation
- `index.html`: descriptions of and links to test framework and related documents.

Edit Application Server Permissions

To install and run tests, the default Sun Java System Application Server permissions must be modified as described in the following steps.

1. Stop the application server.

- a. **Windows: Start > Programs > Sun Microsystems > Application Server PE > Stop Default Server** or execute this command:

```
AppServerInstallDir\bin\asadmin stop-domain domain1
```

- b. **Solaris operating system: execute this command:**

```
AppServerInstallDir/bin/asadmin stop-domain domain1
```

domain1 is the default domain.

2. Make a backup copy of the relevant `server.policy` file.

Forexample: `C:\Sun\AppServer\domains\domain1\config\server.policy`

3. Open the `server.policy` file in a text editor and find the line beginning

```
// Basic set of required permissions
```

4. Below this line, change the following lines (ignore line breaks):

```
permission java.net.SocketPermission "*", "connect";
to

permission java.net.SocketPermission "*",
"accept,connect,listen,resolve";

permission java.io.FilePermission "<<ALL FILES>>", "read,write";
to

permission java.io.FilePermission "<<ALL FILES>>",
"read,write,delete";

permission java.util.PropertyPermission "*", "read";
to

permission java.util.PropertyPermission "*", "read,write";
```

5. Add the following four permissions after:

```
permission java.lang.RuntimePermission "modifyThreadGroup";
Additional Permissions:
permission java.lang.RuntimePermission "createClassLoader";
permission java.lang.RuntimePermission "getClassLoader";
```

```
permission java.util.logging.LoggingPermission "control";  
permission java.net.NetPermission "specifyStreamHandler";
```

6. Save the `server.policy` file.

Deploy Relay

Deploy the Relay to the application server as follows:

- 1. Start the application server as described in “Verify Prerequisites” on page 11.**
- 2. If you have write access to the file system of the host that runs the Application Server, copy `centralInstallDir/server/JdtsServer.war` to `AppServerInstallDir/domains/yourDomain/autodeploy`. The default value of `yourDomain` is `domain1`.**
- 3. If you do not have write access to the Application Server host file system, you must know the following information, then follow the remaining steps in this section:**
 - Administrator port (default 4848)
 - Administrator user name
 - Administrator password
- 4. Login to the Application Server as administrator by the URL <http://appServerHost:appServerAdminPort>.**
- 5. In the left (Common Tasks) column, select Web Applications.**
- 6. In the right (Web Applications) pane, click Deploy.**
- 7. In Deploy Web Module, select Specify a package file to upload, and click Browse.**
- 8. Browse to `centralInstallDir/server/` and select `JdtsServer.war`.**
- 9. In the file browser, click Open.**
- 10. In Deploy Web Module, Click Next.**
- 11. In Deploy Web Module, click OK.**
- 12. Logout of the application server Admin Console.**

Verify the deployment with this URL:

<http://appServerHost:appServerPort/appContext>. For example, <http://localhost:8080/JdtsServer>. The Application Server displays a page that says “Java Device Test Framework Relay *Version* is running”.

Unzip Developer's Kit

You can skip these steps if you do not plan to develop tests.

- 1. Create a directory to hold the developer's kit files.**

This guide refers to the directory holding the unzipped developer's kit as *devKitInstall*.

- 2. Unzip `CentralInstallDir/admin/shared/devkit.zip` into your developer's kit directory.**

Unzipping creates a `tests/` and a `docs/` directory and two `tpim` script files. The plugin online help describes these files and directories.

Install NetBeans Platform Plugin

You can skip these steps if you do not plan to develop tests.

- 1. Start the NetBeans integrated development environment.**

- 2. Choose Tools > Plugins.**

- 3. Solaris operating system users and Window users who did not download the "Java" or "All" NetBeans software bundle:**

- a. Click the Available Plugins tab.**

- b. Check (tick) Mobility.**

- c. Click Install.**

- d. Follow the wizard instructions. On the final wizard page, select Restart IDE Later, then click Finish.**

- 4. Click the Downloaded tab.**

- 5. Click Add Plugins.**

- 6. Navigate to the test framework directory, select `jdtf_nb_plugin.nbm`, and click Open.**

- 7. Follow the wizard instructions.**

- 8. If you want to make the test framework's Javadoc™ tool documentation accessible from the IDE, follow these steps:**

- a. Select Tools > Libraries.**

The Library Manager appears.

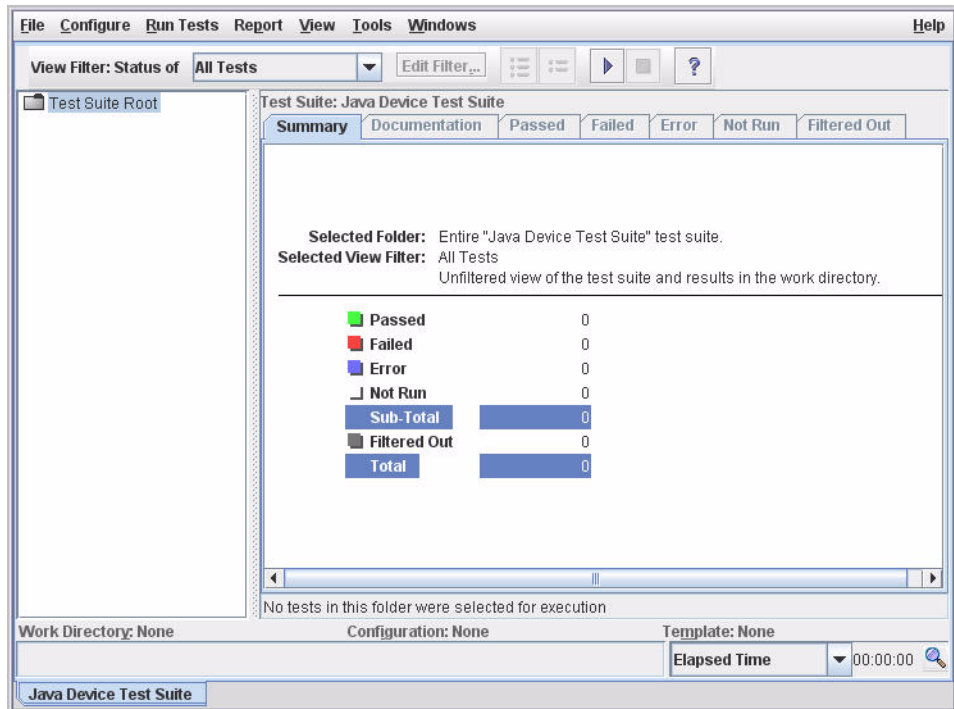
- b. Click **New Library** (lower left corner).
 - c. In the **New Library Dialog**, enter any name for the library, select **Class Libraries** for the **Library Type**, then click **OK**.
 - d. In the **Library Manager**, click the **Classpath** tab.
 - e. Click **Add JAR/Folder**.
 - f. In the file browser, navigate to `CentralInstallDir/admin/shared/resources/agentFiles`.
 - g. Select the following: `Logging-device.jar`, `TestManager-device.jar`, `BenchmarkTestManager.jar`.
 - h. Click **Add JAR/Folder**.
 - i. In the file browser, navigate to `CentralInstallDir/admin/shared/lib`.
 - j. Select `Core.jar`, then click **Add JAR/Folder**.
 - k. In the **Library Manager**, click the **Javadoc** tab.
 - l. Click **Add ZIP/Folder**.
 - m. In the file browser, navigate to `devKitInstall` (see “Unzip Developer’s Kit” on page 16), then to `docs/`.
 - n. Select `test-api/`, `test-dependencyprovider-api/`, and `test-server-api/`, then click **Add ZIP/Folder**.
 - o. In the **Library Manager**, click **OK**.
9. Exit or restart the Netbeans integrated development environment.

Install Basic Test Packs

If you want to experience an interactive exercise that teaches the basics of running tests, or if you want to run tests that can detect some basic device problems, install the basic test packs by following these steps:

1. **Navigate to** `CentralInstallDir/admin/`.
 - a. **Windows:** double click `run.bat`.
 - b. **Solaris operating system:** `$ sh run.sh`
The administrator harness appears, looking similar to [FIGURE 2-1](#).

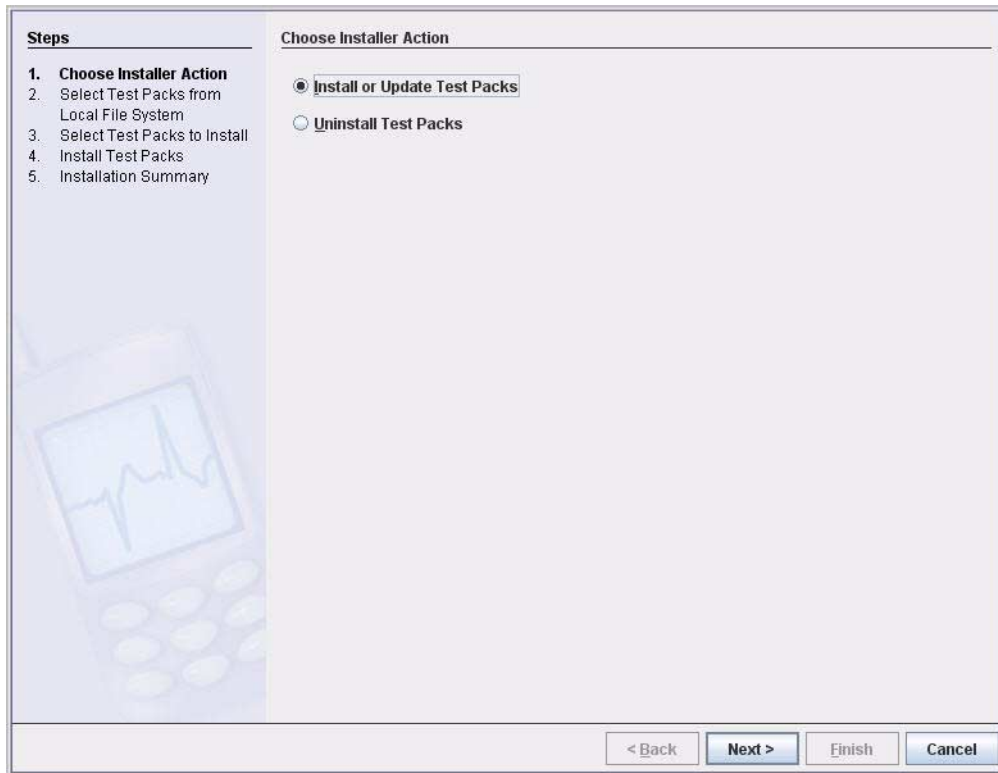
FIGURE 2-1 Administrator Harness After Installation



2. Choose Configure > Test Packs.

The test pack installer appears, looking similar to [FIGURE 2-2](#).

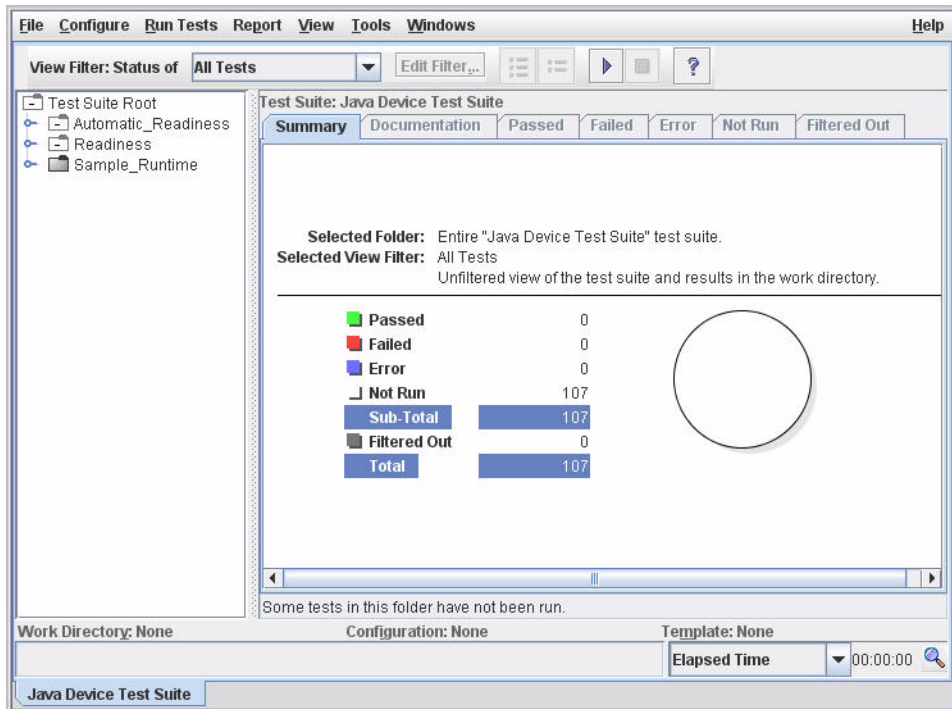
FIGURE 2-2 Test Pack Installer First Screen



3. Select **Install or Update Test Packs**, then click **Next >**.
4. Click **Add**.
5. Navigate to *CentralInstallDir/tp_packages/*, select *tp_packages.zip*, and click **Open**.
6. Click **Next >**.
7. Click **Add All >>**.
8. Click **Next >**.
9. When the test packs have been installed, click **Next >**.
10. Click **Finish**.

The test packs appear in the harness test tree, similar to [FIGURE 2-3](#).

FIGURE 2-3 Test Tree After Basic Test Pack Installation



Group Installation

To install the test framework so the Relay and Central Installation can be shared among a group of testers, read this section.

Central Installation and Relay Installation

For a group installation, you install the Central Installation and Relay once, then install the tester harness one or more times (see ["Tester Installation"](#) on page 27).

System Requirements

TABLE 2-1 shows the hardware and software requirements for Relay and Central Installation hosts. It assumes that each component is installed on a different host. If you install multiple components on one host, the host must meet the aggregate requirements of all of the components.

TABLE 2-1 Relay and Central Installation Host Requirements

Component	Relay Host	Central Installation Host
Software		
Operating System	Solaris 10 operating system for the SPARC® processor, or Windows XP with Service Pack 2	
Java Developer's Kit or Java Runtime Environment	Java Platform Standard Edition Developer's Kit (<i>not</i> Java Runtime Environment) version 1.6.0_03 (also known as JDK version 6.0 Update 3) or greater. You can download the JDK software from http://java.sun.com/javase/downloads/index.jsp . This chapter refers to the directory that holds the JDK software as <i>jdkInstallDir</i> . The officially supported JDK version is 1.6.0_03.	One of the following: <ul style="list-style-type: none">• Java Platform Standard Edition Developer's Kit version 1.6.0_03 (also known as JDK version 6.0 Update 3) or greater. You can download the JDK software from http://java.sun.com/javase/downloads/index.jsp. This chapter refers to the directory that holds the JDK software as <i>jdkInstallDir</i>.• Java Runtime Environment version 1.6.0_03 (also known as JRE™ version 6.0 Update 3) or greater. You can download the JRE software from http://java.sun.com/javase/downloads/index.jsp. This chapter refers to the directory that holds the JRE software as <i>jreInstallDir</i>. The officially supported JDK and JRE versions are 1.6.0_03.
Ant	(not applicable)	Ant version 1.7.1. Download this version from http://archive.apache.org/dist/ant/binaries/ . The file is <code>apache-ant-1.7.1-bin.zip</code> . This chapter refers to the directory that holds the Ant files as <i>antInstallDir</i> . Ant is used to install test packs. Test pack developers also use Ant.

TABLE 2-1 Relay and Central Installation Host Requirements *(Continued)*

Component	Relay Host	Central Installation Host
Sun Java System Application Server	<p>The Relay host requires the Sun Java System Application Server Platform Edition (PE) version 8.2. Download this version from http://java.sun.com/j2ee/1.4/download.html. Other application servers can work but they have not been tested. An application server for the Relay must implement Java Servlet API 2.3 and JavaServer Pages™ 1.2 specifications (JSR 53) or any higher versions of the specifications. This document gives configuration instructions for the Sun Java System Application Server.</p> <p>If you want to install or update using an existing Sun Java System Application Server, the application server must be configured to allow HTTP (not HTTPS) access to its administration port. If administration access is normally by HTTPS, change the configuration to HTTP for the duration of the installation or update, then change it back to HTTPS.</p>	(not applicable)
IP Ports	At least 100 free IP ports for each harness that can run simultaneously	(not applicable)
Hardware		
Disk Space	At least 500 megabytes of free disk space is required to run the Sun Java System Application Server. In addition, during installation, there must be at least 500 megabytes of writable free space in the system's temporary directory, <code>/var/tmp</code> for the Solaris operating system or, by default, <code>%USERPROFILE%\Local Settings\Temp</code> for Windows.	At least 40 megabytes of free disk space is required for the Central Installation. In addition, during installation, there must be at least 500 megabytes of writable free space in the system's temporary directory, <code>/var/tmp</code> for the Solaris operating system or, by default, <code>%USERPROFILE%\Local Settings\Temp</code> for Windows.

TABLE 2-1 Relay and Central Installation Host Requirements (*Continued*)

Component	Relay Host	Central Installation Host
Memory	At least 256 megabytes of RAM is required to run the Sun Java System Application Server. 500–1000 megabytes of memory gives better performance, especially when multiple testers share the Relay.	At least 512 megabytes of RAM is required to run the Java Device Test Framework on Windows platforms. Any host that runs the supported Solaris operating system has enough memory.
Accessibility	<p>Harnesses, test devices, and WAP gateways (if any) must be able to communicate by HTTP with the Relay. If a firewall protects the Relay host, the firewall must permit test devices to connect to Relay host ports that you specify when you install the Relay.</p> <p>An outer firewall that protects the Relay host from the Internet must permit 5-15 simultaneous TCP and/or UDP connections from test devices. Stress tests make the most connections.</p> <p>An inner firewall that protects the Relay host from the internal network (see must permit up to five simultaneous TCP connections from each harness.</p>	The Central Installation must be installed in a shared file system that is accessible to tester harness hosts.
POSIX <code>df</code> command (Solaris operating system only)	In a default Solaris operating system installation, the command is <code>/usr/xpg4/bin/df</code> , and the installer looks for it there. If this directory does not exist or does not contain <code>df</code> , you must prepend the POSIX <code>df</code> command's location to your <code>PATH</code> so the installer finds it before any other version of <code>df</code> . If necessary, you can obtain the POSIX <code>df</code> command from the Solaris installation CD or DVD.	In a default Solaris operating system installation, the command is <code>/usr/xpg4/bin/df</code> , and the installer looks for it there. If this directory does not exist or does not contain <code>df</code> , you must prepend the POSIX <code>df</code> command's location to your <code>PATH</code> so the installer finds it before any other version of <code>df</code> . If necessary, you can obtain the POSIX <code>df</code> command from the Solaris installation CD or DVD.

Recommended Components

The Wireless Toolkit for CLDC version 2.5.2 is recommended for testing. The administrator and tester harnesses can automatically launch the toolkit's emulator and run tests on it. Solaris operating system users must run the toolkit on a Linux host. The test framework NetBeans software plugin online help has instructions. You can download the Wireless toolkit for CLDC from <http://java.sun.com/products/sjwtoolkit/download.html>.

Verify Prerequisites

You can verify that the required software versions are installed with these commands:

- `java -version`
- `ant -version`
- Start the Sun Java System Application Server, then load <http://appServerHost:appServerPort> (for example, <http://localhost:8080>) into a web browser. The browser displays “Your server is up and running”.

To start the Application Server in a Windows environment, choose Start > Programs > Sun Microsystems > Application Server PE > Start Default Server or execute this command:

```
AppServerInstallDir\bin\asadmin start-domain domain1
```

To start the Application Server in a Solaris operating system, execute this command:

```
AppServerInstallDir/bin/asadmin start-domain domain1
```

Prepare Configurer

An Ant script known as the configurer prepares the downloaded test framework files for use. After you have installed the test framework, you can re-run the configurer to change properties of your installation.

You specify configuration options in a file called `jdtfconfigurer.properties`, which is in the test framework’s top directory. Open `jdtfconfigurer.properties` in a text editor. If you need a fresh copy of `jdtfconfigurer.properties` (because someone else has previously edited it), extract it from the test framework zip file.

1. Inspect and change the following lines in `jdtfconfigurer.properties` as necessary (default values shown - Windows users, note forward slashes):

a. Set the following property value to false. This property is for individual installations.

```
do.configure.client.platform.home=true
```

b. Specify the directory to hold test framework Central Installation (administrator harness, documentation, built-in test packs):

```
central.install.dir=c:/sun/jdtf-ci
```

This directory must be empty or non-existent.

- c. Specify the path to the Java Platform Standard Edition Developer's Kit (note forward slashes in Windows path name):

```
java.exec=c:/program files/java/jdk1.6.0_10/bin/java.exe
```

- d. Change this IP address to the Relay host's true IP address as seen by administrator and tester hosts. Do not specify localhost or 127.0.0.1 because the Relay is likely installed on a remote computer.

```
relay.server.host.console=127.0.0.1
```

- e. If the Relay's port for incoming HTTP requests from administrator and tester hosts is not the default 8080, specify the port number.

```
relay.server.port.console=8080
```

- f. Change this IP address to the Relay host's true IP address as seen by a test device. Do not specify localhost or 127.0.0.1 because this address represents a test device's point of view, and the Relay runs on a host computer, not on a test device:

```
relay.server.host.device=777.777.777.777
```

- g. If the Relay's port for incoming HTTP requests from test devices is not the default 8080, specify the port number.

```
relay.server.port.device=8080
```

- h. If the default path is incorrect or undesirable, specify a directory for the Relay to use for test-related storage.

```
relay.persistent.storage.root.dir=c:/sun/jdtf-server-storage
```

This path must refer to a file system that the application server can access. In the default value, drive c refers to the drive on the application server host, which is not necessarily the c drive on the host where you will run jdtfconfigurer. The Relay creates the directory when it needs it.

- i. If the Relay must be restricted to particular IP ports for communicating with tests, change the default value.

```
relay.server.port.range=any
```

You can specify individual ports, or ranges, or combinations, for example: 1001,1010-1050,1100-1140. Specify about 100 total ports.

- If you already have a Relay installed in this application Server, and it uses the default application context name, change the default value:

```
relay.application.context=JdtsServer
```

- j. If you have installed an emulator, change the default value if it is not correct.

```
default.emulator.home=c:/WTK2.5.2
```

- k. If you want the administrator harness to use a different name to distinguish this emulator from others you might also install, change the default value:

```
default.emulator.alias=JDTF-EMU
```

2. **Save and close** `jdtfconfigurer.properties`.

Run Configurer

Run the configurer by executing this command in the directory containing `jdtfconfigurer.properties`:

```
ant -f jdtfconfigurer.xml
```

The configurer creates a Central Installation at the location you specified in `central.install.dir`. The Central Installation directory contains these important files:

- `server/relay.application.context.war`: the Relay component which you deploy to the application server (see [“Deploy Relay” on page 26](#)). The default value for `relay.application.context` is `JdtsServer`.
- `admin/shared/resources/localRelay/relay.application.context.war` which you can deploy to test on a device that communicates by HTTP over a serial cable connected to your workstation. See the *Local Relay User’s Guide* described in `index.html` for a description of the local Relay and how to install it.
- `admin/run.bat` and `run.sh`: scripts that launch the administrator harness for running tests.
- `index.html`: descriptions of and links to Java Device Test Suite documents included in the test framework.

Edit Application Server Permissions

Edit the application server permissions as described in [“Edit Application Server Permissions” on page 14](#).

Deploy Relay

Deploy the Relay as described in [“Deploy Relay” on page 15](#).

Install Basic Test Packs

This step is optional but recommended. If you want to experience an interactive exercise that teaches the basics of running tests, or if you want to run tests that can detect some basic device problems, install the basic test packs by following the steps in “Install Basic Test Packs” on page 17.

Tester Installation

If you have already created a Central Installation and have deployed and started the Relay, you can create tester installations as described in this section.

Prerequisites

Before you create a tester installation, have the following software already installed on the tester’s host computer:

- One of the following operating systems:
 - Solaris 10 operating system for the SPARC® processor
 - Windows XP with Service Pack 2
- A Java Runtime Environment (JRE) software version 6 Update 3 (also known as 1.6.0_03) or greater

Sun recommends using the latest JRE software. JRE version 6 Update 3 is the officially supported version.

You can download the JRE software at

<http://java.sun.com/javase/downloads/index.jsp>

- (Recommended) Sun Java Wireless Toolkit 2.5.2 for CLDC

You can download the Wireless Toolkit at

<http://java.sun.com/products/sjwtoolkit/download.html>

- Solaris operating system hosts: The POSIX version of the `df` command.

In a default Solaris operating system installation, the command is `/usr/xpg4/bin/df`, and the installer looks for it there. If this directory does not exist or does not contain `df`, you must prepend the POSIX `df` command’s location to your `PATH` so the installer finds it before any other version of `df`. If necessary, you can obtain the POSIX `df` command from the Solaris installation CD or DVD.

Use the following guidelines for hardware requirements:

- Disk space - The tester harness installation uses about 20 megabytes of disk space. Reports, especially XML reports, are the major variable consumer of disk space. 100 megabytes per report is good for estimating, assuming you are running all tests. You need less space to run test subsets.
- Memory - Allow about 500 megabytes for the harness, another 100 megabytes for the Sun Java Wireless Toolkit for CLDC.

Verify Prerequisites

You can verify that the required software versions are installed with these commands:

- `java -version`
- `ant -version`
- Start the Sun Java System Application Server, then load <http://appServerHost:appServerPort/appContext> (for example, <http://localhost:8080/JdtsServer>) into a web browser.

To start the Application Server in a Windows environment, choose Start > Programs > Sun Microsystems > Application Server PE > Start Default Server or execute this command:

```
AppServerInstallDir\bin\asadmin start-domain domain1
```

To start the Application Server in a Solaris operating system, execute this command:

```
AppServerInstallDir/bin/asadmin start-domain domain1
```

Prepare Configurer

An Ant script known as the configurer prepares the downloaded test framework files for use. After you have installed the test framework, you can re-run the configurer to change properties of your installation.

You specify configuration options in a file called `jdtfconfigurer.properties`, which is in the test framework's top directory. Open `jdtfconfigurer.properties` in a text editor. If you need a fresh copy of `jdtfconfigurer.properties` (because someone else has previously edited it), extract it from the test framework zip file.

To prepare the configurer for a tester installation, follow these steps:

1. **Inspect and change the following lines in `jdtfconfigurer.properties` as necessary (default values shown - Windows users, note forward slashes in path names):**

- a. Set the following property value to `false`. This property is for a Central Installation.

```
do.install.central=true
```

- b. Set the following property value to `true`.

```
do.install.runner=false
```

- c. Set the following property value to `false`. This property is for a Central Installation.

```
do.configure.central=true
```

- d. Set the following property value to `true`.

```
do.configure.runner=false
```

- e. Set the following property value to `false`. This property is for a Central Installation.

```
do.configure.relay.wars=true
```

- f. If you have not have installed an emulator, set the following property value to `false`.

```
do.configure.default.emulator=true
```

- g. Set the following property value to `false`. This property is for developer installations.

```
do.configure.client.platform.home=true
```

- h. Ask an administrator for the path to the shared directory that holds the Central Installation, then change the following property value if necessary. The default value is for an individual installation.

```
central.install.dir=c:/sun/jdtf-ci
```

- i. Change the following property value if you want the tester harness installed elsewhere.

```
runner.install.dir=c:/sun/jdtf-runner
```

- j. Specify the path to the Java Runtime Environment software. The default value is for an individual installation, which requires a JDK version.

```
java.exec=c:/program files/java/jdk1.6.0_10/bin/java.exe
```

- k. If you have installed an emulator, change the default value if it is not correct.

```
default.emulator.home=c:/WTK2.5.2
```

1. If you want the harness to use a different name to distinguish this emulator from others you might also install, change the default value:

```
default.emulator.alias=JDTF-EMU
```

2. Save and close `jdtfconfigurer.properties`.

Run Configurer

Run the configurer by executing this command in the directory containing `jdtfconfigurer.properties`:

```
ant -f jdtfconfigurer.xml
```

The configurer creates a tester harness at the location you specified in `runner.install.dir`. This directory contains these important files:

- `run.bat` and `run.sh`: scripts that launch the tester harness for running tests
- `index.html`: descriptions of and links to documents related to the test framework.

Post-installation Reconfiguration

You can reconfigure your installation by modifying `jdtfconfigurer.properties` and running the configurer again. Study the comments in `jdtfconfigurer.properties` before reconfiguring. If you reconfigure, set `do.configure.central` to `false`.

Note these limitations:

- You cannot change an installation directory (a property whose name contains `.dir`),
- If you change a Relay property (one whose name begins with `relay.`), you must deploy the Relay again.

Launching the Administrator Harness

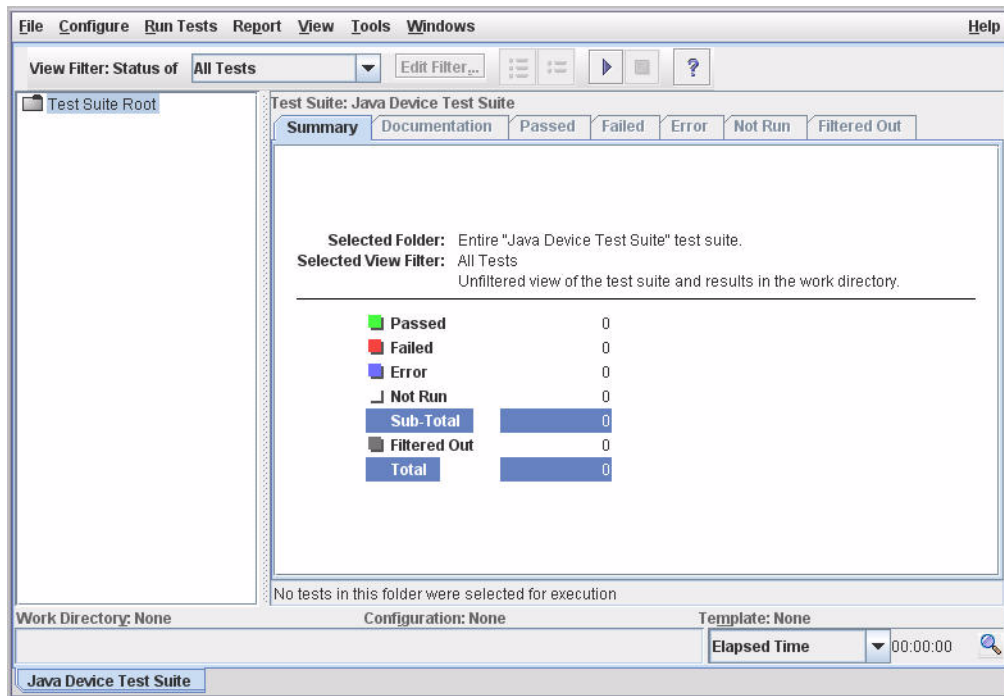
For instructions on launching the tester harness, see the *Java Device Test Suite Tester's Guide* at [\[URL\]](#). If you are working alone, there is no reason to use the tester harness.

To launch the administrator harness, follow one of these steps, depending on your operating system:

- **In the Solaris operating system, enter this command (C shell):**
 - `sh CentralInstallDir/admin/run.sh`
- **In the Windows XP environment, do one of the following:**
 - Navigate to `CentralInstallDir\admin\`, then double-click `run.bat`.
 - Enter this command:
 - > `CentralInstallDir\admin\run.bat`

The administrator harness graphical user interface appears, similar to [FIGURE 3-1](#). Your harness might look different depending on installed tests and its state when you last quit it.

FIGURE 3-1 Administrator Harness Initial Display



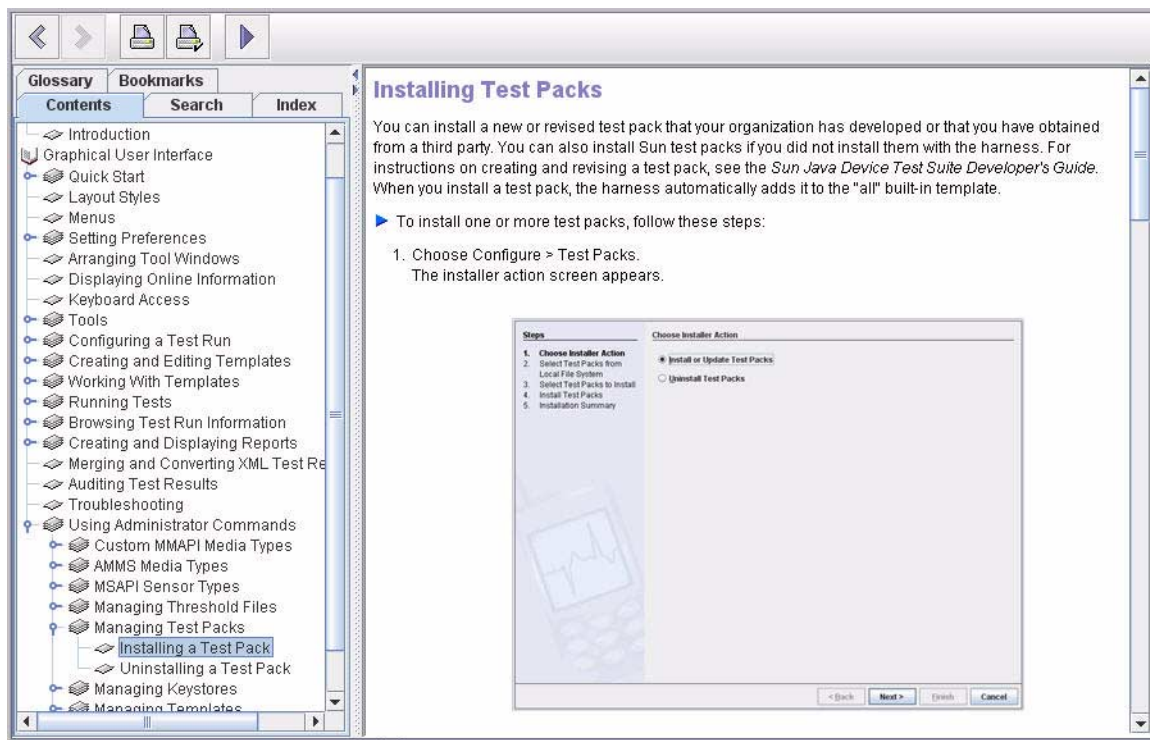
Installing a Test Pack

Follow these steps:

1. **Acquire the test pack, which is a zip file, and save it where the administrator harness can access it.**
2. **Launch the administrator harness as described in “[Launching the Administrator Harness](#)” on page 31.**
3. **In the administrator harness, choose `Configure > Test Packs` and follow the screen instructions.**

If you need help, choose `Help > Online Help`, open the `Using Administrator Commands` topic, then open the `Managing Test Packs` topic, then select `Installing a Test Pack` (see [FIGURE 4-1](#)).

FIGURE 4-1 Installing a Test Pack Help



Connecting Test Devices

For runtime and benchmark tests, there are several ways to connect the test device to the test framework so that tests can flow to the device and test results can flow to the harness. This chapter describes the options and the details of test device-harness communication in the following sections:

- [Test Device Requirements](#)
- [Test Device Connection Options](#)
- [Specifying the Transmission of Bundles and Results](#)

The material in this chapter does not apply to over-the-air provisioning (OTA) tests. Those tests are structured very differently and have no options for communication. In particular, the means of bundle transfer and result disposition described in this chapter has no effect on the execution of OTA tests.

Test Device Requirements

Test devices must meet these requirements:

- **Heap space.** At least 128KB of heap space is required. Some tests require more space.
- **MIDlet suite size:** A test device must be able to download and install MIDlet suites (test bundles) of at least 80Kbytes in size. Most tests are larger but can fit in 128Kbyte bundles. A few tests are as large as 1,300Kbytes.

Test Device Connection Options

In addition to the device requirements, a test device must support at least one of the connection options described in this section. The connection options support the transfer of tests and results:

- Tests must be transferred from the Relay to a test device so they can run on the device.
- Test results must be transferred from the test device to the Relay so they can be incorporated in reports and displayed in the harness. If the device cannot transfer results, they can be displayed on the device.

Test Bundle Transfer

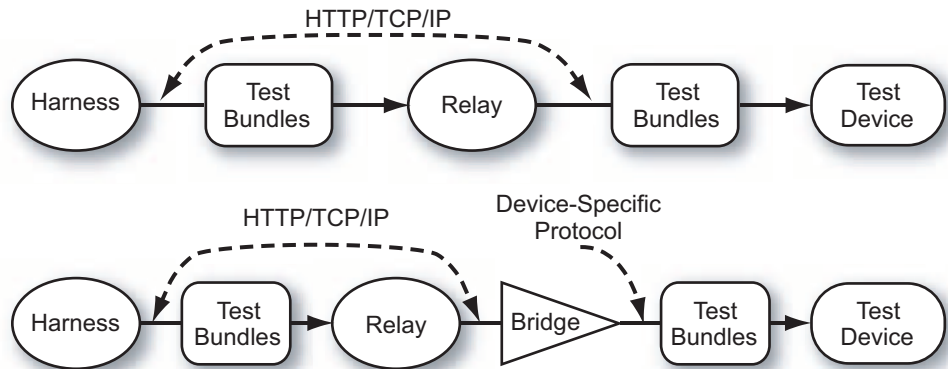
When you direct the harness to run tests, the harness packs groups of tests into MIDP MIDlet suites, which are called test bundles. To run the tests in a bundle, the bundle must be transferred to the test device, installed, and launched. The bundles can be transferred by HTTP or over a local link, such as a serial cable. You specify your choice in the Tests and Bundles section of a template or configuration. The operations for installing and launching a bundle are device dependent.

HTTP Bundle Transfer

When test bundles are transferred by HTTP, the harness repeatedly creates test bundles and sends them to the Relay. You then download a bundle from the Relay, install it and launch it, which causes the tests to run.

The Relay can transfer test bundles over a TCP/IP connection that supports HTTP version 1.0 or 1.1. The device itself does not need to support HTTP over TCP/IP if bridge hardware or software between the test device and the Relay can act as an HTTP client on behalf of the device. A WAP gateway is one example of such a bridge. [FIGURE 5-1](#) illustrates HTTP test bundle downloading with and without a bridge.

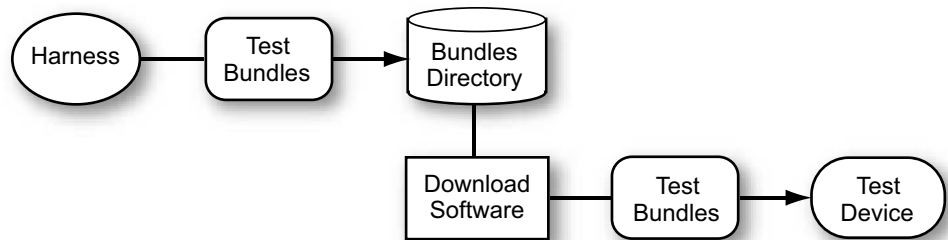
FIGURE 5-1 Test Bundle Transfer Options - HTTP



Local Link Bundle Transfer

The alternative to HTTP bundle transfer is to use a local link, such as serial, infrared, or Bluetooth. This option requires cooperating software on the harness host or another host. In the Configuration Editor, specify a directory into which the harness stores the test bundles. Direct the software to download the test bundles from the directory to the test device. [FIGURE 5-2](#) illustrates test bundle transfer by a local link.

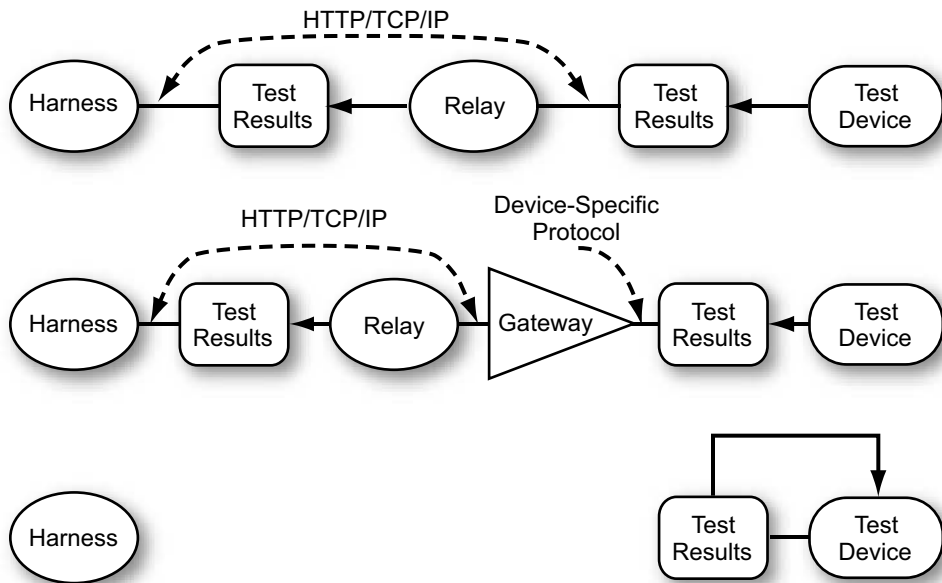
FIGURE 5-2 Test Bundle Transfer - Local Link



Test Result Disposition

For test results to be transferred from a test device to a harness, the test device's Java software or a gateway interposed between the device and the Relay, must support HTTP version 1.0 or 1.1 over TCP/IP. If the test device or gateway does not support HTTP, test results are not sent to the harness. Instead, test results can be displayed on the device. [FIGURE 5-3](#) illustrates the possible test result disposition options:

FIGURE 5-3 Test Result Disposition Options



Specifying the Transmission of Bundles and Results

In the Configuration Editor, specify how tests are downloaded to the test device and how test results are handled. You can choose from the following test bundle transfer and result disposition options:

- **Option A.** Manually download each test bundle over an HTTP connection. Return the test results to the harness using HTTP.

You can select this value if both of the following conditions are true:

- **Bundle Download.** You can direct the test device to download, install, and launch a MIDlet suite (a test bundle) from a web server, possibly with assistance from an intermediating bridge, such as a WAP gateway.
- **Result Disposition.** The test device has working Java methods for HTTP communication that the agent can use to send results to the harness. The results can also pass through a bridge.

For this option, set configuration properties in the Tests and Bundles section as follows:

- Autotest Support: No
 - Bundle Transfer Channel: By HTTP
 - Result Disposition: Yes
- **Option B.** Manually download each test bundle over an HTTP connection. Hold and display the test results on the test device.

This option is for test devices whose Java methods for HTTP communication are absent or do not work correctly. The device must still be able to download MIDlets using HTTP, which is possible if the device's native HTTP implementation is working.

For this option, set configuration properties in the Tests and Bundles section as follows:

- Autotest Support: No
 - Bundle Transfer Channel: By HTTP
 - Result Disposition: No
- **Option C.** Manually download each test bundle from a directory on the harness, using a local link such as a serial line. Return the test results to the harness using HTTP.

Select this value if both of the following conditions are true:

- **Bundle Download.** You can direct the test device, or associated software, to download, install, and launch a MIDlet suite from a harness directory. The MIDlet file transfer can be conducted over a serial line, a Bluetooth connection, an infrared connection, or similar.
- **Result Disposition.** The test device has working Java methods for HTTP communication that the agent can use to send results to the harness.

For this option, set configuration properties in the Tests and Bundles section as follows:

- Autotest Support: No
- Bundle Transfer Channel: By local link

- Result Disposition: Yes
- **Option D.** Manually download each test bundle from a directory on the harness, using a local link such as a serial line. Hold and display the test results on the test device.

This option is for devices whose Java methods for HTTP communication are absent or do not work correctly.

For this option, set configuration properties in the Tests and Bundles section as follows:

- Autotest Support: No
- Bundle Transfer Channel: By local link
- Result Disposition: No
- **Option E.** Have the test device implement the harness's autotest protocol, in which the test device repeatedly and automatically downloads, installs, launches, and removes test bundles until all tests have been run. Test results are returned to the harness using HTTP. This option is only available if the device supports the autotest protocol.

For this option, set configuration properties in the Tests and Bundles section as follows:

- Autotest Support: Yes

If your test device supports more than one of the options consider these factors when making your choice:

- If the test device's HTTP connection is over the air, options C and D might transfer bundles faster. However, the reverse might be true if the HTTP connection is over a local network, such as Ethernet or Wi-Fi.

When bundles are transferred using a local link and the last test run completes, you must press the Stop button so the harness is no longer waiting for more bundles to be transferred to the test device and is no longer in a run test mode.

- One option might be easier to use, and possibly faster, because it requires fewer test device interactions (such as key presses) to download, install, and launch a test bundle. For example, the local link download software might require fewer keystrokes than downloading over the air.
- If option D is used, you must manually stop the test run when test execution completes or if a problem occurs during test execution. Press the Stop button to stop the test run.
- The autotest protocol is easy to use because the device obtains, installs, launches, and removes test bundles. However, it is usually available only on emulators or specially manufactured versions of test devices.

- The options where results are held on the device are the most inconvenient to use. Select one only if none of the “Send” options work with the test device. The options where results are held on the device do not return results to the harness, require you to manually display interactive test instructions, and only support a subset of tests.

Running Tests

The *Java Device Test Suite Tester's Guide's* "Running Tests" chapter is an interactive exercise in running simple tests and creating reports. It is a good way to get experience in the essentials of running tests.

Preparation

To run tests, you need the following:

- A work directory, into which the test results are stored. You can create a work directory with the harness's File > Create Work Directory command. You can open an existing work directory with File > Open > Work Directory, or File > Recent Work Directories. When you launch the harness, it automatically opens the most recently used work directory.
- A template, which specifies configuration information, usually for a test device. You specify a template when you create a work directory. You can create a new template and associate it with the current work directory with Configure > New Template. You can associate a different template with the current work directory with Configure > Load Template.
- A configuration, which also specifies configuration information, usually for a test device. You can create a new configuration for the current work directory with Configure > New Configuration. The configuration is initialized with the values in the work directory's template.

If you are a new user or are testing a new device, you might want to start with the Readiness tests described in the *Java Device Test Suite Tester's Guide's* "Readiness Tests" chapter.

Selecting Tests

There are multiple ways to select the tests to run.

- If you want to run a small number of tests one time, you can select them in the harness test tree, right click, and choose Execute These Tests. You can use Shift-select and Ctrl-select to select multiple tests, and you can select packages and classes.
- If you want to run a group of tests repeatedly, specify them in the configuration. Choose Configure > Edit Template, then specify the tests in the Tests To Run section. You can choose tests in the following ways
 - By selecting in a test tree
 - By including a list of test names in a file
 - Directly by feature (a feature is a collection of tests that exercise a device feature, such as record and playing a video).
 - By keywords associated with tests, for example, Interactive or RadioTuner.
 - By status of the previous result, for example, Failed or Error.
 - By test severity, meaning the seriousness of a failure to the device user experience.
 - By relevance to the test device's capabilities as expressed in other configuration values. For example, if some tests require a hypothetical location capability, and you set the property isLocationSupported to False, those tests do not run because they cannot pass.

If you specify more than one criteria, a test runs if it meets all criteria.

Configuring the Test Run

A configuration or template has properties that affect the way the tests are run.

- Tests and Bundles section: In this section, specify how the device receives test bundles and returns results (see [“Connecting Test Devices” on page 35](#)).
- Timeouts section: The default values are likely acceptable in most devices. You can revisit this section if tests are timing out too soon or too slowly.
- MicroAgent section: The default values are likely acceptable in most devices. Revisit these properties if you have difficulties.

Configuring Test Packs

To configure test packs, you must have both the test pack documentation and the test device documentation.

- Security Permissions section: Specify the intersection of the permissions the test device supports and the permissions the tests you want to run require.
- Test Pack Properties section: For test packs containing tests you want to run, specify values for properties that do not have defaults, and specify device-specific values for other properties when they differ from the defaults. Properties designated Advanced rarely have to be set.

Configuring Test Cases

Some test cases have configurable properties that are separate from the test pack properties. You do not set test case properties in the configuration or template. Instead, in the harness window, you select a test case and right click. If the Configure Test menu item is enabled, choose it to inspect or configure test case properties.

Setting the View Filter

The harness view filter, above the test tree, helps you focus on subsets of tests. Tests colored gray are filtered out (though their results remain in the work directory). For more information, see the online help topic Tools:Test Manager Tool:View Filters. View filter summary:

- All Tests: Shows the status of all tests.
- Last Test run: Shows the status of the most recently run tests.
- Current Template: Shows the status of tests selected by the template.
- Current Configuration: This is the most frequently used setting. It shows the status of tests selected by the configuration. You can use it to see which tests will run and not run when you click the Start button.

Starting the Test Run

The online help's Running Tests topic has more information.

To run tests on the Wireless Toolkit emulator, follow these steps:

1. **Configure the emulator by choosing File > Preferences and then selecting Java Device Test Suite Preferences.**

The online help topic Setting Preferences has more information.

2. **Choose Run Tests and check (tick) the Run on Emulator box.**
3. **Click the Start (blue right-pointing triangle) button.**

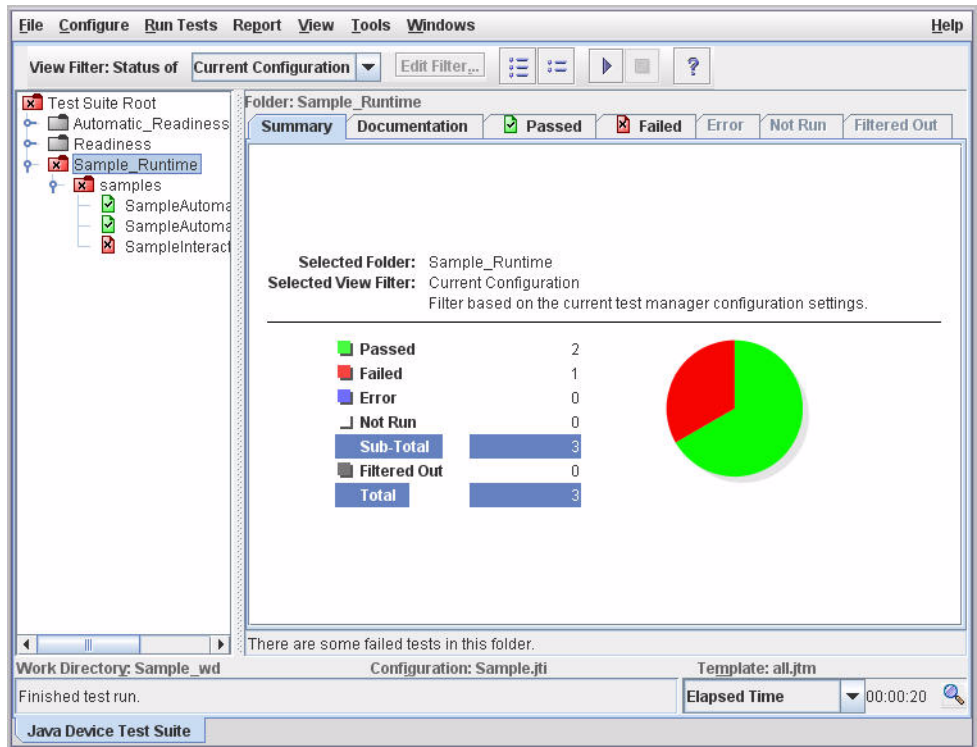
To run tests on a device, ensure that Run Tests > Run on Emulator is not checked (ticked), then click the **Start button**.

Inspecting Results

The harness stores test results in the work directory and colors the icons in the test tree to show each test's status. For a description of the status values and colors, see the online help topic [Browsing Test Run Information:Displaying Folder Information:Status Information](#).

- Set the view filter (see [“Setting the View Filter” on page 45](#)) to focus on the tests of interest.
- To see summary results for a group of tests, select Test Suite Root or a subordinate test pack, package, or class in the test tree. [FIGURE 7-1](#) shows an example.

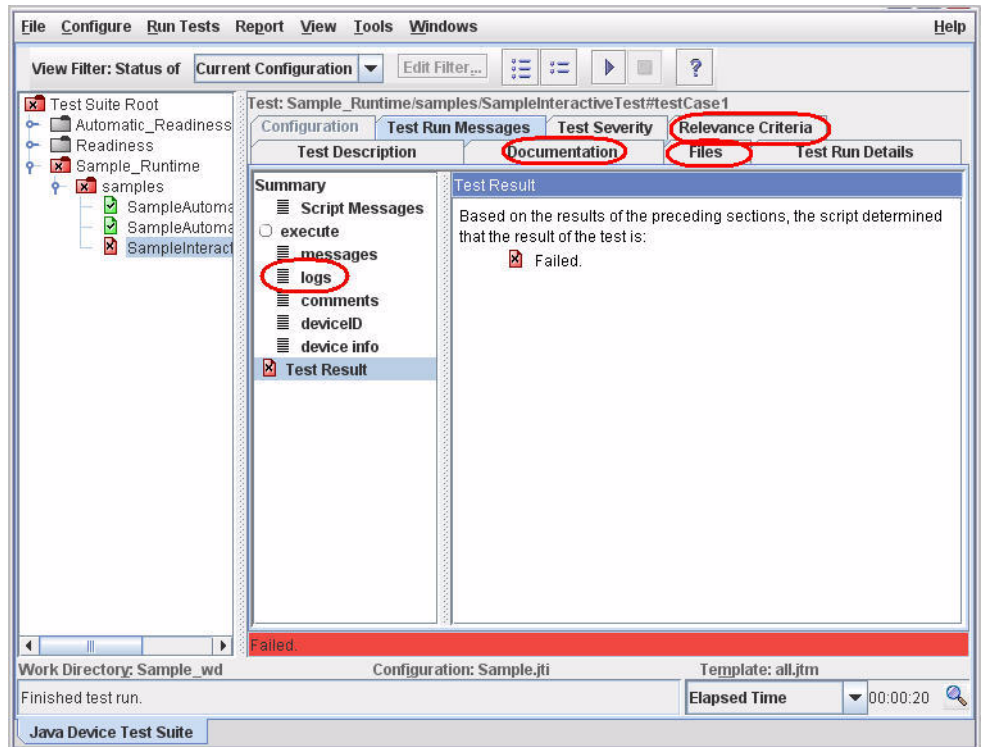
FIGURE 7-1 Package Test Results



- To inspect a test case's results in detail, select the case (lowest level in the test tree).

The harness displays a summary, similar to [FIGURE 7-2](#).

FIGURE 7-2 Failed Test Results



- For a Failed test, click logs to see diagnostic messages emitted by the test as it ran.
- To see why a test has been filtered out (icon colored gray), click Relevance Criteria.
- To see a test's documentation, click Documentation.
- To see a test's source files, click Files, then use the Files pull-down to select a file to view.

Creating Reports

The *Java Device Test Suite Tester's Guide's* "Running Tests" chapter is an interactive tutorial that guides you through running basic tests and creating reports.

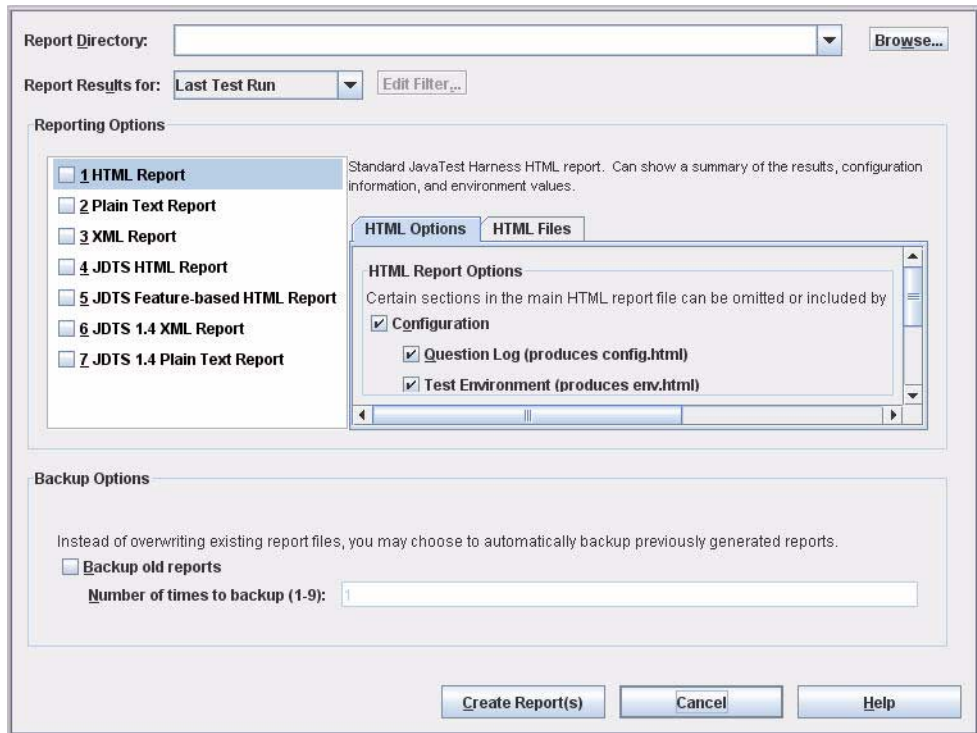
Note – If you execute the exercises in "Running Tests", substitute the `all.jti` template for the `sample.jti` template, which is not included with the test framework.

To create a report from the results in a work directory, launch the administrator harness (see "[Launching the Administrator Harness](#)" on page 31) and follow these steps.

- 1. Choose Report > Create Report.**

The Create Report dialog appears, similar to [FIGURE 8-1](#).

FIGURE 8-1 Create Report Dialog



2. If you want help using this dialog, click **Help**.
3. Specify a **Report Directory** into which to write the reports.
4. Use the **Report Results** drop-down to subset the results you want reported.
The Report Results items are analogous to view filters, see [“Setting the View Filter” on page 45](#)).
5. For guidance on the type of reports to create (checkboxes on left side), click **Help**, then click **Reporting Options**.
6. After checking (ticking) one or more report types, select each type and set options to the right.
7. If you want to keep old reports in the **Report Directory**, check (tick) **Backup old reports**.
8. Click **Create Report(s)**.
9. When the harness asks if you want to view the reports, click **Yes** and then verify that the reports contain the information you want.

Sharing Data

You can exchange results and templates with other test framework users. Sharing can be helpful for documenting device defects and repairs, and for reproducing test conditions.

Sharing Results

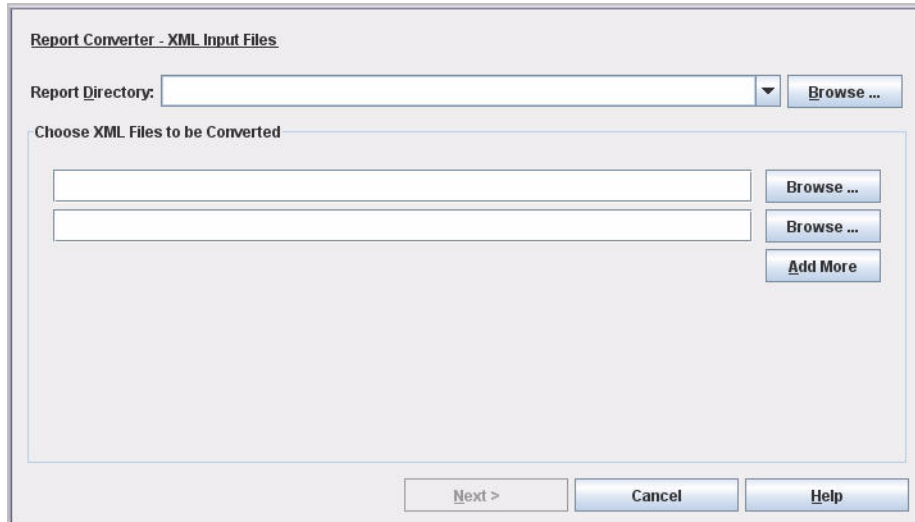
To share test results, create an XML report, as described in [“Creating Reports” on page 51](#). XML files are portable (plain text) and can be emailed and processed by many applications. A test framework user can import an XML report and convert it to a more readable report type, such as HTML.

To import and convert an XML report to another type, follow these steps:

- 1. Launch the administrator harness and choose Tools > Report Converter.**

A wizard similar to [FIGURE 9-1](#) appears. If you need help with the report converter, click Help.

FIGURE 9-1 Report Converter Wizard First Screen



2. In **Report Directory**, specify is the location of the output report.
3. **Specify the XML files you want to convert.**
If you specify more than one XML file, the output report is a merger of the XML files.
4. **Click Next.**
5. **Check (tick) the report types you want to create.**
6. **Click Create Report(s).**

Sharing Templates

A template can specify much of the environment in which a test was run. The alternative of specifying the same information in a configuration is not helpful for sharing because configurations cannot be shared.

Template files have the suffix `.jtm`. You can exchange a template by any means that accepts a text file. Do not edit a template file with anything but the test framework template editor. Changing the text in any other way invalidates the template.

Template portability has one limitation, the path names of local files. These include:

- Exclude lists

- Tests-to-run lists
- Custom severity definitions
- Threshold files for benchmark tests
- Keystore files for security tests.

Many templates do not refer to these files and are therefore self-contained and fully portable. Before you share a template, either remove such references with the template editor or include copies of the referenced files and instructions for creating equivalent references in the template you share.

Writing Tests

Writing tests is a creative activity for which step-by-step instructions are not possible. To begin learning, launch the NetBeans integrated development environment, choose Help > Help Contents, then double-click Java Device Test Framework. Read these topics first:

- The Test Pack Development Cycle
- Starting Your Own Test Pack
- Adding a Test Class File
- Writing the `testsuite.info` File
- Writing Runtime Tests (Automated and Interactive Topics)

Uninstalling

To uninstall the test framework, follow these steps:

- 1. Ensure that there are no files (projects, test packs, templates, and so on) that you want to save in *CentralInstallDir* or in the unzipped test framework directory.**

The next steps delete both directories.

- 2. Use file system commands to delete the following files and directories:**

- *CentralInstallDir*
- Unzipped test framework
- Test framework zip file
- Tester harness install directory (if installed)

- 3. Undeploy the Relay.**

You can find undeploy instructions in the *Java Device Test Suite Administration Guide*.

- 4. Optionally, uninstall the Sun Java System Application Server.**

You can find uninstall instructions in the *Java Device Test Suite Administration Guide*.

Index

A

administrator harness
 and templates, 8
 definition, 7
 in a typical use case, 2
 launching in graphical mode, 31
application server permissions
 for individual installation, 14
automated test, definition, 5

B

benchmark tests, 6

C

Central Installation
 definition, 6
 installing, 20
configuration
 definition, 7
 of a test run, 44
configurer
 executing for group installation, 26
 execution for individual installation, 13
 execution for tester installation, 30
 preparation for individual installation, 12
 preparing for group installation, 24
 preparing for tester installation, 28

D

developer's kit, installing, 16

H

hardware requirements, 21, 27

I

installation
 group, 9, 20
 individual, 9
 reconfiguration, 30
 tester, 27
interactive test, definition, 5

L

local link, bundle transfer using, 37

N

NetBeans plugin
 installing, 16
 learning about, 57

O

over-the-air tests, 6

P

prerequisite software
 group installation, 21
 individual installation, 10
 tester installation, 27

R

reconfiguration, 30
Relay
 definition, 7

- deploying for individual installation, 15
- installing, 20
- role in test framework, 3
- report
 - creating, 51
 - for sharing, 53
- requirements
 - Central Installation host, 21
 - for tester installation, 27
 - hardware for tester installation, 27
 - Relay host, 21
 - test device, 35
- running tests, 43
- runtime tests, 5

S

- sharing
 - templates, 54
 - test results, 53
- software requirements, 21
- status,test, 5

T

- template
 - definition, 8
 - families, 8
 - portability limitations, 54
 - sharing, 54
- templates manager, 8
- test
 - automated, 5
 - interactive, 5
- test bundles
 - setting transfer options, 38
 - transfer by HTTP, 36
 - transfer using local link, 37
 - transferring, 36
- test case
 - configuring, 45
 - definition, 4
- test class, 4
- test device
 - connecting to harness, 35
 - connection options, 36
 - definition, 4
 - requirements, 35
- test framework

- definition, 1
- main components, 2
- typical use case, 2
- test harness, definition, 7
- test pack
 - configuring, 45
 - definition of, 3
 - in test hierarchy, 4
 - installing, 33
- test package, 4
- test packs
 - installing basic, 17
- test results
 - creating a report of, 51
 - disposition, 38
 - inspecting, 47
 - sharing, 53
- test run
 - configuring, 44
 - starting, 46
- test status, 5
- tester harness
 - definition, 7
 - installation, 27
- tests
 - benchmark, 6
 - over-the-air, 6
 - running, 43
 - runtime, 5
 - selecting, 44
 - writing, 57

U

- uninstalling, 59

V

- view filter, 45

W

- work directory
 - and configuration, 43
 - and template, 43
 - creating, 43
 - creating a report from, 51
 - definition, 7
 - opening, 43
- writing tests, 57