

vanadyne Jobs is a job management and automation application characterized by the following outstanding features:

- ▶ Application security that can be applied to a user or a user group from the most general to the smallest unit of a user's interaction with an application.
- ▶ Any desired automation scheme, made possible by including multiple date and time patterns in a single schedule.
- ▶ Powerful capabilities through the use of a sequential set of distinct process steps as a unit of execution.
- ▶ Seamless integration of customer defined modules as another action step in a process sequence.
- ▶ Operating system independent processes.
- ▶ Comprehensive test conditions at each process step, based on the outcomes of previous steps.
- ▶ Easy to use test and debugging features, for making test runs before committing critical processes to automation.
- ▶ Affordable pricing model for companies of any size and budget.

The application runs on all systems for which a Java virtual machine is available.

This document presents a general view to vanadyne Jobs. The specific details are described in the User's Manual. Fig. 1 shows a typical panel of the application.

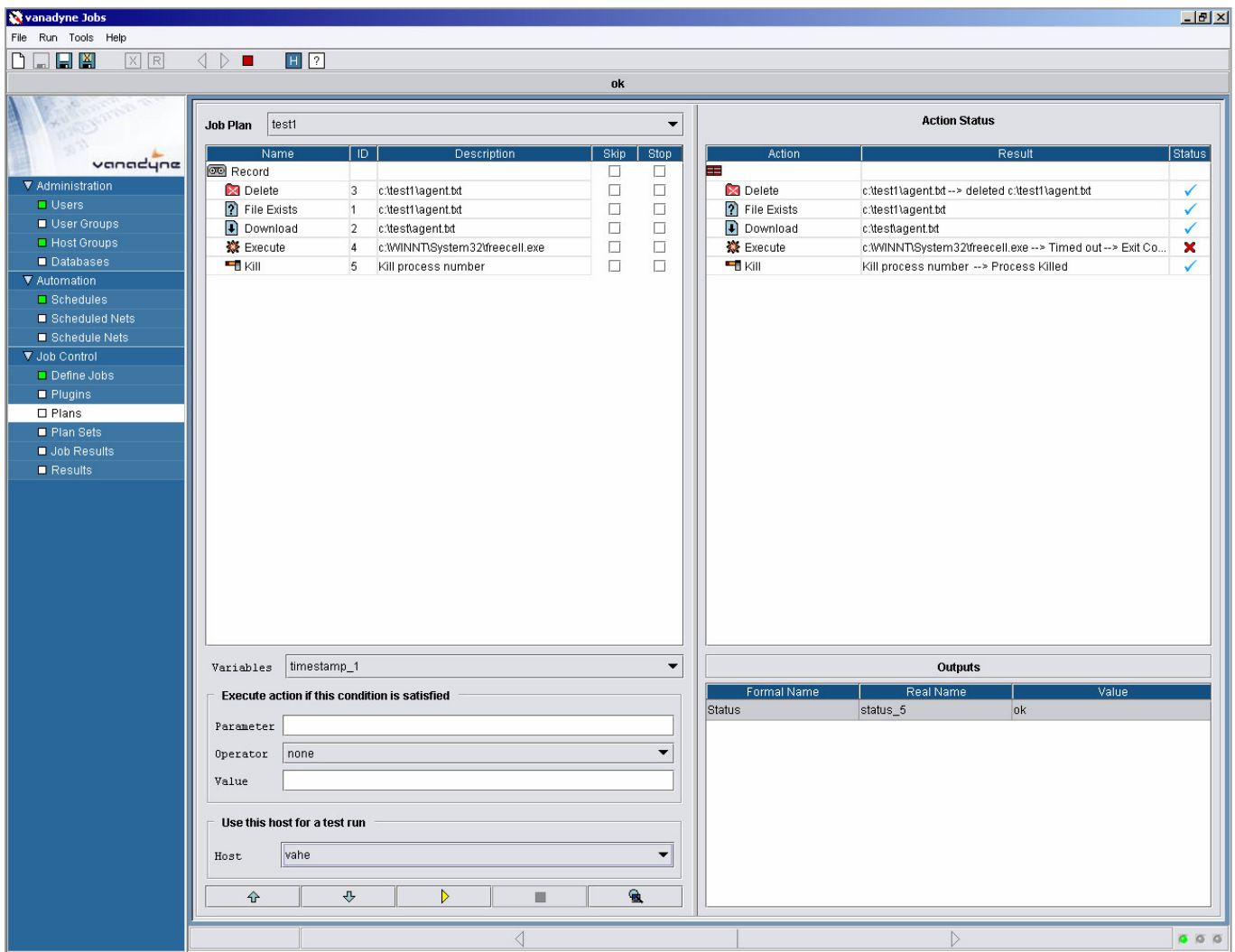


Fig. 1: The Plans Panel

A. APPLICATION SECURITY

The basic entity that can access an application is the 'user'. A user can access applications assigned to him by the 'administrator': the root user.

Each user can change the default password specified by the administrator, the first time he logs in or at any other time.

A user group is a list of users who collectively share the rights assigned to an application. The mechanisms for assigning specific rights are identical for users and user groups.

A user can belong to several 'user groups', in which case he inherits the rights of each group for the specific application.

The rights for an application are divided in three main categories, Object Access Rights, Actions Rights and Host Rights:

1. Object access rights

These are rights for manipulating database objects. They include display, edit, delete, create or any other relevant object manipulation rights defined by the application provider.

Included in this category are 'panel rights', which allow a user to see a specific panel in the application tree. For example the administrator can deny a user the display of the 'Schedules' panel.

2. Action Rights

These are 'dynamic' rights, such as start, stop or schedule a job or a sequence of processes. For example in 'vanadyne Jobs' such actions would be trial run of a Job Plan on a certain host or starting a Plan Set involving several hosts, etc.

3. Host Rights

A host group is a subset of hosts in the active network. A host group may contain a single host. Host groups can be assigned to users or user groups for single panels. Thus for example, if a user wishes to make a trial run of a 'Job Plan', he can do this only on hosts to which he has the access rights to for the 'Plans' panel.

If a user is denied access to a certain host, this host is excluded in any action defined by the application on the corresponding panel.

Fig. 2 illustrates the various options available to set the rights of a user for a certain applications. The name of the application in this case is Jobs. The various panels used by this application are shown in the left panel, and the right panel shows the relevant action rights for the panel called 'Schedules'.

USER DATA

User Name:

Password:

Repeat Password:

User Groups:

User Applications:

Jobs Panel Rights			Schedules Action Rights		
Number	Property	Value	Number	Property	Value
1	Scheduled Nets	false	1	Delete	false
2	User Groups	false	2	Edit	false
3	Users	false	3	Allowed Owners	none
4	Schedules	false			
5	Plan Results	false			
6	Schedule Nets	false			
7	Results	false			
8	Plugins	false			
9	Define Jobs	false			
10	Job Plans	false			
11	Job Actions	false			
12	Host Groups	false			

Fig. 2 Users

B. AUTOMATION

Schedules are the entities that can be assigned to tasks for the purpose of starting them at the dates and times defined in the schedule. Once defined a schedule can be assigned to multiple executable objects. A schedule is a set of date and time patterns, whose logical union forms the final dates at which processes are triggered. By including as many of these patterns in a schedule, almost any desired combination of starting dates can be defined. The schedule depicted in the Fig. 3 consists of four patterns. There are two patterns with a monthly frequency, a weekly pattern and a daily pattern. Any coincidence of dates resulting from the logical union of these patterns will be considered as a single date, unless the starting times of these coincidences differ.

Time Zone = Europe/Berlin GMT+02:00

Period	Start Date	Start Times	Skip	Span	Step	Description
Monthly	8/3/06	14 : 16	1	0	0	On the first working day of the month
Monthly	8/2/06	14 : 16	1	0	0	On Day 2 of each month
Weekly	8/2/06	14 : 16	1	0	0	On Wednesdays
Daily	7/1/06	12 : 23	1	0	0	On Selected Days

Fig. 3 Date Patterns

C. PLANS

Plans consist of a sequence of action steps on a host to accomplish a desired task. An action can be simple or complex. Simple actions include such items as 'check for file existence, set variables, download or upload files, execute a job, kill a process, send mail, append text to files etc. Complex actions are 'plugins' with specific functionality that are tailored to the specific needs of end users; or are general enough to be used by a wide range of applications. For example a 'plugin' could monitor a directory for changes and send notifications when such changes occur.

An action can test the outcomes of previous actions to decide what to do. For example, if a previous action signals that a file exists, the subsequent 'download' action for this file can be skipped, or if a job times out, the next action could decide to kill the process, notify an operator by email or restart it with a different parameter set.

Plans may have a set of predefined variables, which can be used singly or in combinations by any action as a parameter. This facility can be used to launch the same plan with different sets of starting variable values. As mentioned earlier, all outputs of the various steps in a Plan can also be used as variables by the subsequent actions in the Plan; however, these variables are treated as local for the Plan.

The example on the next page should help clarify the operation of a Plan.

This plan executes a typical Baan Job with certain pre and post processing steps, reacts to exceptional conditions and reports the outcome to an ERP system. The plan uses a central repository for scripts, hence reducing risks inherent in starting scripts with different versions in different locations. The individual steps follow and are best understood by examining *Fig. 4*.

Name	ID	Description	Skip	Stop
Record			<input type="checkbox"/>	<input type="checkbox"/>
Delete	1	Clean Directory	<input type="checkbox"/>	<input type="checkbox"/>
File Exists	2	c:\scripts\Baan_1	<input type="checkbox"/>	<input type="checkbox"/>
Download	3	d:\Central\scripts\Baan_1	<input type="checkbox"/>	<input type="checkbox"/>
Execute	4	c:\bin\bw.exe c:\scripts\Baan_1	<input type="checkbox"/>	<input type="checkbox"/>
Append Text	5	Update Logfile	<input type="checkbox"/>	<input type="checkbox"/>
Utilities	6	Get Duration	<input type="checkbox"/>	<input type="checkbox"/>
Kill	7	Kill Script if duration > Timeout	<input type="checkbox"/>	<input type="checkbox"/>
Jump To	8	Run script once more	<input type="checkbox"/>	<input type="checkbox"/>
Utilities	9	Get Duration for Second Run	<input type="checkbox"/>	<input type="checkbox"/>
Mail	10	Notify if rerun timed out	<input type="checkbox"/>	<input type="checkbox"/>
Plugin	11	Update ERP system	<input type="checkbox"/>	<input type="checkbox"/>

Fig. 4 The Plan

Step 1: Cleanup.

Delete files in a scratch directory.

Step 2: Check for File Existence.

Check if the required script exists on the current host.

One can check the file length, modification date etc., to make sure the correct version of the script is available.

Step 3: Download.

If the file does not exist or is an old version, download it from a central repository.

In the bottom of Fig. 1, in the left hand corner, the conditional execution is shown in the panel with the heading, "Execute action if this condition is satisfied".

Here the value from Step 2 is evaluated (for example, download the file if it does not exist on the host).

The condition operators include, "equal, greater than, less than, in range or outside a range", etc. The in range condition can be used, for example, to decide what to do if the exit code of an executable is within a certain range.

Step 4: Execute the Job

The job depicted here is a Baan script with the appropriate parameters and the relevant environmental setting necessary for its proper execution as shown in Fig. 4.1.below.

Step 5: Append Text

Add an entry to the log file.

Step 6: Get Duration

Get the elapsed time since the job started.

You can get the total elapsed time since the start of any action step in your "Job Plan", and make decisions for the subsequent steps.

Step 7: Kill the Script

If the duration in Step 6 is longer than the timeout specified for the executable, kill the running script.

Step 8: Jump to Step 4

Rerun the script with the same or a different set of parameters. You can set the number of reruns to any value. In this case it is only once.

Step 9: Get Duration

Get the duration for the rerun.

Step 10: Notify Operator

Send an email to the operator if the rerun timed out. You can attach files and/or insert the content of a file in the mail text.

Step 11: Update ERP System

Here is an example of a "Plugin" as a complex action step.

Job Properties

Executable Path:

Working Directory:

Standard Output:

Standard Error:

Maximum Duration: Don't Wait

For OS: ▼

Parameters		Environment Settings	
No	Value	Name	Value
1	ttaad5203m000	BSH_JOBOK	Baan_1_OK
		BSH_JOB	Baan_1
		BSH_FRSES	Baan_1_RES

Fig 4.1 Job Parameters

D. PLAN SETS

Plan Sets are the executable units, which can be scheduled to run at dates specified in a “Schedule” as defined in paragraph B (“Automation”). Each Plan Set consists of a sequence of plans that are assigned a specific host to run on. The starting variables of each Plan in a Plan Set are assigned new values in a Plan Set, to define the runtime parameters for its use. For example if a plan uses a ‘Job Name’ as a variable, the same plan can be used in different plan sets to start different jobs.

Plan sets are organized in ‘Packages’. A package is a container for a set of Plan Sets, which represents an organizational unit. Plan sets in a package cannot have duplicate names; however, the same name can be used for Plan sets in different packages. This makes versioning easy and avoids naming conflicts. Packages are in turn contained in Projects; as a further facility to organize Plan sets in large organizations.

Schedules are assigned to Plan Sets, which will then start automatically at the dates specified in the schedule. These schedules can be activated or de-activated as needed. *Fig 5* shows a Project with a single package, which contains three Plan Sets. The Plan Set called ‘plan3’ is not scheduled and consists of a single Plan, whereas ‘plan1’ has two Plans running on two different hosts and is activated by a schedule, starting it every 3 minutes; ‘plan2’ is also assigned a schedule but is not activated.

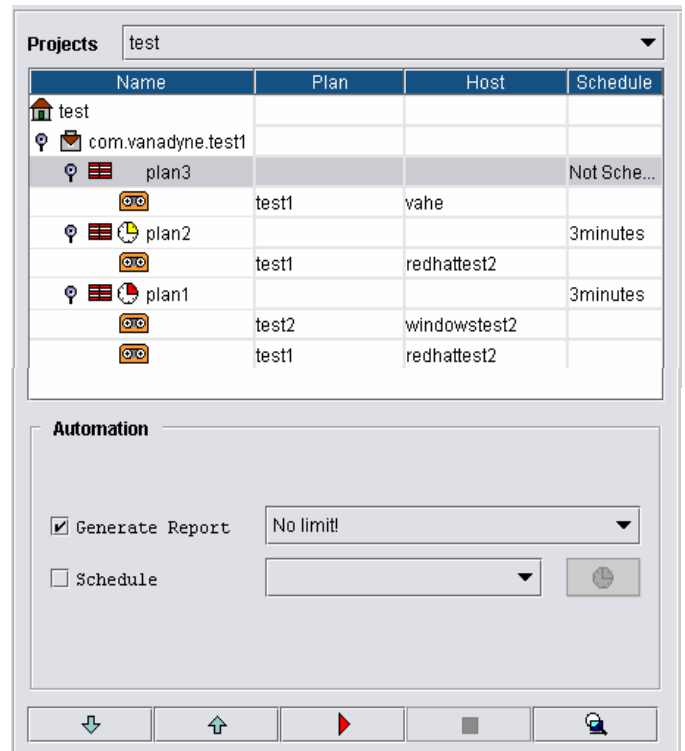


Fig. 5 Action Sets

A Plan set can be started interactively by a user from the GUI, to test its proper operation before automating it with a schedule. The outcomes of each action step are then shown in a separate panel with the values of the outputs listed in a table at the bottom of the panel. This makes it simple to analyse the flow of the complete process, and to take corrective measures as necessary. *Fig. 6* shows the results of a typical test run, started in the GUI.

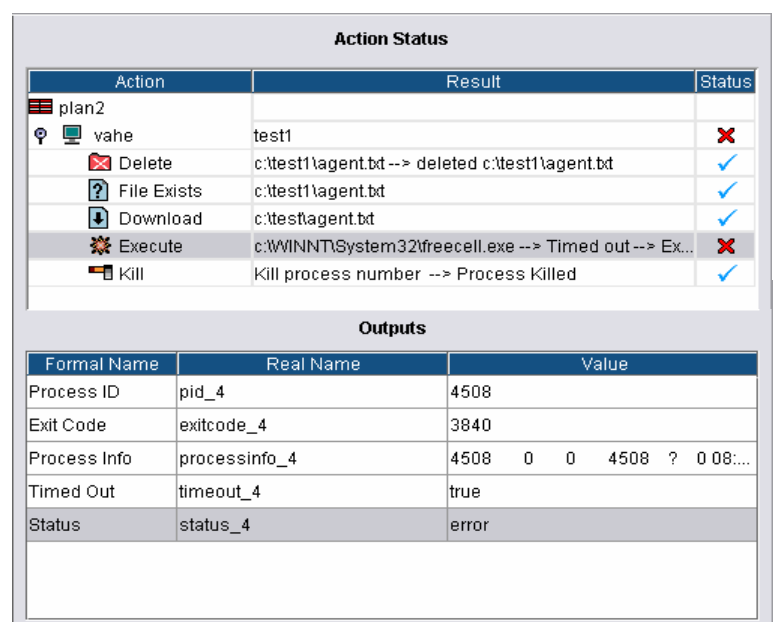


Fig. 6 Test Run

E. RESULTS

Information about executed Plan Sets is recorded and stored for later analysis and further processing if necessary. Each record of such information includes the full details of each action step with the values of the outputs, status and the starting dates and duration of the action. Important comments are also included to facilitate analysis of the information.

In addition, a process that is marked active can be stopped if so desired.

Planset Reports					
com.vanadyne.test1.plan3					
No.	Plan Name	Status	Start Date	End Date	
1	Automated	✗	7/25/06 7:12 AM	7/25/06 7:12 AM	
2	Automated	✗	7/25/06 7:09 AM	7/25/06 7:09 AM	
3	Automated	✗	7/25/06 7:06 AM	7/25/06 7:06 AM	
4	Automated	✗	7/25/06 7:03 AM	7/25/06 7:03 AM	
5	Automated	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
6	test1	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
7	Delete	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
8	File Exists	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
9	Download	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
10	Execute	✓	7/25/06 7:00 AM	7/25/06 7:00 AM	
11	Automated	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
12	test1	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
13	Delete	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
14	File Exists	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
15	Download	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
16	Execute	✓	7/25/06 6:57 AM	7/25/06 6:57 AM	
17	Automated	✗	7/25/06 6:54 AM	7/25/06 6:54 AM	
18	Automated	✗	7/25/06 6:51 AM	7/25/06 6:51 AM	
19	Automated	✗	7/25/06 6:48 AM	7/25/06 6:48 AM	

Fig 7. Results

F. APPLICATION MONITORING

With this functionality the administrator of vanadyne Jobs will have an overall view of the state of the application over a long period of time at a glance. It is a great help for trouble shooting and planning for resources, etc.

1. Calendar:

In this panel you can select a time interval, for example the last two days. This interval is then used to show all jobs with their related status, including "active" and "scheduled jobs".

INTERVAL		STATISTICS								
November 2006		STATUS	COUNT	BARGRAPH						
S	M	T	W	T	F	S	✗ Total	500	100 %	<div style="width: 100%;"></div>
5	6	7	8	9	10	11	✓ Success	400	80 %	<div style="width: 80%;"></div>
12	13	14	15	16	17	18	✗ Error	50	10 %	<div style="width: 10%;"></div>
19	20	21	22	23	24	25	⚠ Warn	50	10 %	<div style="width: 10%;"></div>
26	27	28	29	30			🟢 Active	10		
From: 11/19/06 12:00 AM							🟡 Scheduled	50		
To: 11/20/06 11:59 PM										

Fig 8. Calendar and Statistics Panel

2. Statistics:

This panel shows both numerical and bar graph statistics of the jobs in the interval you selected in the Calendar. You can see in the bar graph the relative sizes of jobs that were ok, failed or produced warnings, as well as the number of active and scheduled jobs.

3. Graphs:

This panel displays several graphs related to the long-time data concerning the application. From the dropdown menu, you can select which graph you would like to see. In the screenshot the number of jobs/day is displayed. With the blue slider you can analyze the details for each day. You can zoom in and out on any desired time interval. The time interval for zooming is decided by moving the yellow and blue sliders.

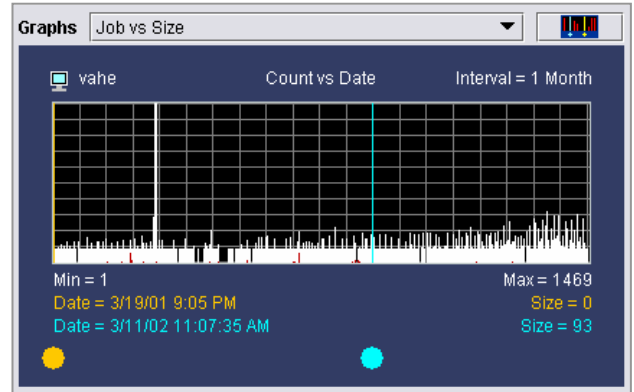


Fig 9. The Graphs Panel

4. Detail:

In this panel the detailed records of the job executions are shown for the interval you selected in the Calendar. Each record shows such items as, start and end times, the user who started the job, the status and the reasons for errors if any.

No.	Job Name	Status	Start Date	End Date	Host	User	Message
1	com.vanadyne.test.testKill	✓ ok	11/6/06 11:20 AM	11/6/06 11:20 AM	vahe	administrator	
2	com.vanadyne.test.testKill	✓ ok	11/6/06 11:21 AM	11/6/06 11:21 AM	vahe	administrator	
3	com.vanadyne.test.testKill	✓ ok	11/6/06 11:24 AM	11/6/06 11:26 AM	vahe	administrator	
4	com.vanadyne.test.testKill	✓ ok	11/6/06 11:26 AM	11/6/06 11:31 AM	vahe	administrator	
5	com.vanadyne.test.testKill	⚠ warn	11/6/06 11:35 AM	11/6/06 11:36 AM	vahe	administrator	Some actions failed
6	com.vanadyne.test.testkill2	⚠ warn	11/6/06 11:41 AM	11/6/06 11:41 AM	vahe	administrator	Some actions failed
7	com.vanadyne.test.testkill2	✗ error	11/6/06 11:49 AM	11/6/06 11:49 AM	vahe	administrator	Some actions failed
8	com.vanadyne.test.testKill	✓ ok	11/6/06 11:57 AM	11/6/06 11:57 AM	vahe	administrator	
9	com.vanadyne.test.testKill	⚠ warn	11/6/06 12:08 PM	11/6/06 12:08 PM	vahe	administrator	Some actions did not c...
10	com.vanadyne.test.testkill2	✗ error	11/6/06 12:09 PM	11/6/06 12:09 PM	vahe	administrator	Some actions failed
11	com.vanadyne.test.ProcessOr...	✗ error	11/6/06 12:09 PM	11/6/06 12:09 PM	vahe	administrator	Some items timed out
12	com.vanadyne.test.fileexists	✓ ok	11/6/06 12:13 PM	11/6/06 12:13 PM	vahe	administrator	
13	com.vanadyne.test.fileexists	✓ ok	11/6/06 12:18 PM	11/6/06 12:18 PM	vahe	none	
14	com.vanadyne.test.testkill2	✗ error	11/6/06 12:20 PM	11/6/06 12:20 PM	vahe	none	Some actions failed
15	com.vanadyne.test.testKill	⚠ warn	11/6/06 12:20 PM	11/6/06 12:20 PM	vahe	none	Some actions did not c...
16	com.vanadyne.test.fileexists	✓ ok	11/6/06 12:21 PM	11/6/06 12:21 PM	vahe	none	
17	com.vanadyne.test.fileexists	✓ ok	11/6/06 12:24 PM	11/6/06 12:24 PM	vahe	none	
18	com.vanadyne.test.testkill2	✗ error	11/6/06 12:25 PM	11/6/06 12:25 PM	vahe	none	Some actions failed
19	com.vanadyne.test.testKill	⚠ warn	11/6/06 12:25 PM	11/6/06 12:25 PM	vahe	none	Some actions did not c...
20	com.vanadyne.test.fileexists	✓ ok	11/6/06 6:27 PM	11/6/06 6:27 PM	vahe	none	
21	com.vanadyne.test.fileexists	✓ ok	11/6/06 6:30 PM	11/6/06 6:30 PM	vahe	none	
22	com.vanadyne.test.testkill2	✗ error	11/6/06 6:30 PM	11/6/06 6:30 PM	vahe	none	Some actions failed
23	com.vanadyne.test.testKill	⚠ warn	11/6/06 6:30 PM	11/6/06 6:30 PM	vahe	none	Some actions did not c...
24	com.vanadyne.test.fileexists	✓ ok	11/6/06 6:33 PM	11/6/06 6:33 PM	vahe	none	
25	com.vanadyne.test.testkill2	✗ error	11/6/06 6:35 PM	11/6/06 6:35 PM	vahe	none	Some actions failed
26	com.vanadyne.test.testKill	⚠ warn	11/6/06 6:35 PM	11/6/06 6:35 PM	vahe	none	Some actions did not c...

Fig. 10. The Detail Panel