Megatest/Logpro Training

Using the Megatest Regression & Automation Engine and the Logpro log file analysis tools to do robust QA and automation.

Matt Welland, 2013

Megatest Information

- Main development site:
  - http://chiselapp.com/users/kiatoa/megatest
- Mirror
  - http://www.kiatoa.com/fossils/megatest
- Source Forge Page
  - http://sourceforge.com/projects/megatest

Overview

- Background on Megatest
- Getting started
    - Run management
    - Creating a Megatest area
    - Creating tests/tasks
    - Getting information about runs and tests
    - Write Logpro files
- Advanced Megatest topics
- Future Megatest development

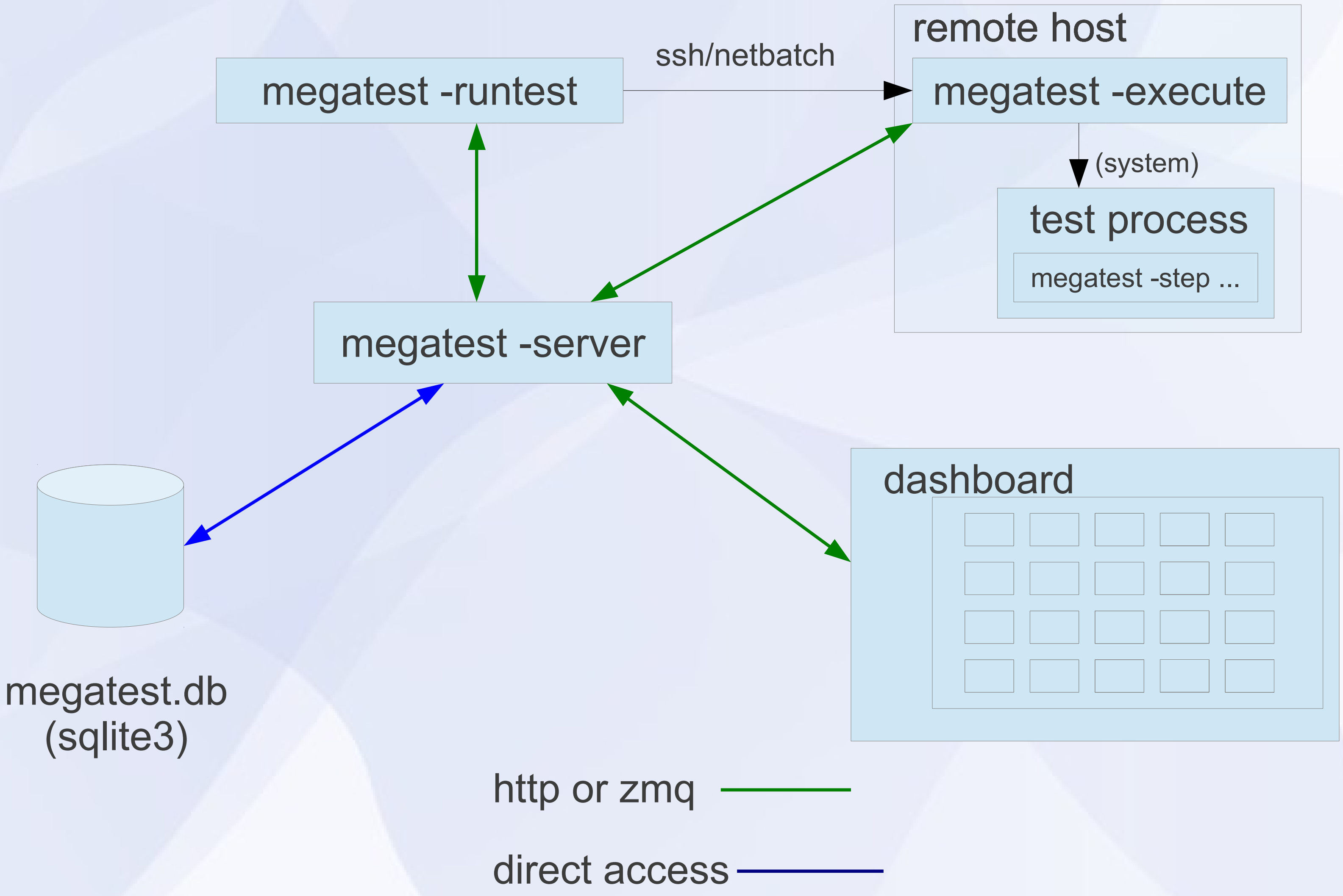
What Can Megatest Do?

- Run tests with
  - one or many steps
  - dependencies on other tests
  - on different hosts
- Report, record and roll up
  - PASS, FAIL, WARN, CHECK, SKIPP
  - Test generated data details

Megatest Architecture

- config files
  - megatest.config
  - runconfigs.config
  - tests/<testname>/testconfig
- SQL database
  - megatest.db
- Tools
  - megatest (command line), dashboard (gui), and logpro (log file analysis via rules)

How it Works


remote host
ssh/netbatch
megatest -runtest
megatest -execute
(system)
test process
megatest -step ...
megatest -server
dashboard
megatest.db
(sqlite3)
http or zmq
direct access

Terminology

target	one or more “keys” separated by “/”, used to organize runs hierarchically; examples include host, platform, stage (e.g. development, final QA, alpha, beta) and so forth. E.g target = x86/centos/dev where the keys are ARCHITECTURE, OS, and RELEASE
run name	a unique name (within a single target grouping) for a run, a common idiom is to use week and day numbers: date +%V.%u
a “run”	a group of tests run under a single target and run name
iterated test	a single test run multiple times with variables iterated over a range of values
state	the state of a test; NOT_STARTED, RUNNING, COMPLETED etc.
status	the current status of this test given its state; PASS, FAIL, n/a

Megatest Design Philosophy

Repeatable	this test result can be recreated in the future
Encapsulated	the area where the test was run is self-contained and all inputs and outputs to the test can be found in the test run area.
Traceable	environment variables, host OS and other possibly influential variables are captured and kept recorded.
Relocatable	the test area can be checked out and the tests run anywhere
Immutable	once this test is run it cannot be easily overwritten or accidentally modified.
Deployable	anyone on the team, at any site, at any time can run the tests
Self-checking	strive for directed or self-checking test as opposed to delta based tests

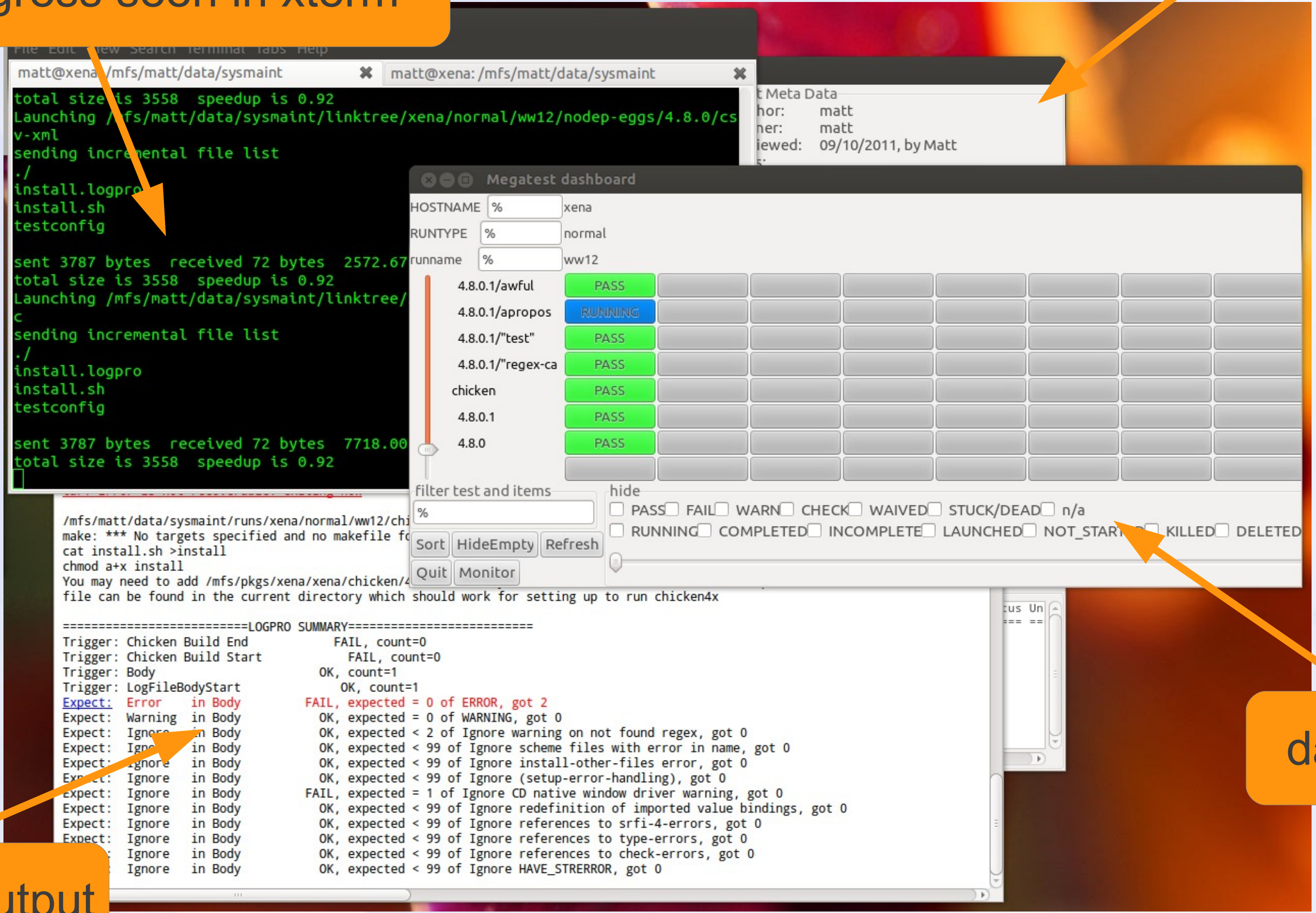
Wisdom is knowing when it is ok to bend or break the rules.

Megatest strives to make it straightforward to do things right but still possible to get the job done when the rules must be bent or broken.

Quick Look

run progress seen in xterm

test control panel (in background)



File Edit View Search Terminal Tabs Help
matt@xena: /mfs/matt/data/sysmaint
matt@xena: /mfs/matt/data/sysmaint
Megatest dashboard
Sort
HideEmpty
Refresh
Quit
Monitor

```
total size is 3558  speedup is 0.92
Launching /mfs/matt/data/sysmaint/linktree/xena/normal/ww12/nodep-eggs/4.8.0/cs
v-xml
sending incremental file list
./
install.logpro
install.sh
testconfig

sent 3787 bytes  received 72 bytes  2572.67
total size is 3558  speedup is 0.92
Launching /mfs/matt/data/sysmaint/linktree/
c
sending incremental file list
./
install.logpro
install.sh
testconfig

sent 3787 bytes  received 72 bytes  7718.00
total size is 3558  speedup is 0.92
```

t Meta Data
hor: matt
ner: matt
iewed: 09/10/2011, by Matt

HOSTNAME % xena
RUNTYPE % normal
runname % ww12

4.8.0.1/awful	PASS
4.8.0.1/apropos	RUNNING
4.8.0.1/"test"	PASS
4.8.0.1/"regex-ca	PASS
chicken	PASS
4.8.0.1	PASS
4.8.0	PASS

filter test and items %

hide
□ PASS □ FAIL □ WARN □ CHECK □ WAIVED □ STUCK/DEAD □ n/a
□ RUNNING □ COMPLETED □ INCOMPLETE □ LAUNCHED □ NOT_STARTED □ KILLED □ DELETED

```
/mfs/matt/data/sysmaint/runs/xena/normal/ww12/chi
make: *** No targets specified and no makefile fo
cat install.sh >install
chmod a+x install
You may need to add /mfs/pkgs/xena/xena/chicken/4
file can be found in the current directory which should work for setting up to run chicken4x

=========================LOGPRO SUMMARY=========================
Trigger: Chicken Build End           FAIL, count=0
Trigger: Chicken Build Start           FAIL, count=0
Trigger: Body                        OK, count=1
Trigger: LogFileBodyStart              OK, count=1
Expect:  Error    in Body          FAIL, expected = 0 of ERROR, got 2
Expect:  Warning  in Body            OK, expected = 0 of WARNING, got 0
Expect:  Ignore   in Body            OK, expected < 2 of Ignore warning on not found regex, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore scheme files with error in name, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore install-other-files error, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore (setup-error-handling), got 0
Expect:  Ignore   in Body          FAIL, expected = 1 of Ignore CD native window driver warning, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore redefinition of imported value bindings, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore references to srfi-4-errors, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore references to type-errors, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore references to check-errors, got 0
Expect:  Ignore   in Body            OK, expected < 99 of Ignore HAVE_STRERROR, got 0
```

dashboard

logpro output

Run Management

- Launching runs
  - command line; “megatest -runtests”
  - test control panel; push “run” then “execute”
- Removing runs
  - “megatest -remove-runs”
- Rolling up runs
  - “megatest -rollup”

Task/Test Management

- Killing jobs
  - In the gui set status to “KILLREQ” and the job will be killed.
  - Command line example:

```
megatest -set-state-status KILLREQ,FAIL -target ubuntu/nfs/none \
           :runname % -testpatt %/% :state RUNNING
```

- Changing state and status of tests
  - Use -set-state-status, see the killing jobs example above.
- Add “-rerun FAIL” to your launch command line to force the re-run of failed jobs

Getting information

- -list-runs pattern
  - lists runs with runname matching pattern.
- -extract-ods
  - creates an open-document spreadsheet
- Miscellaneous queries

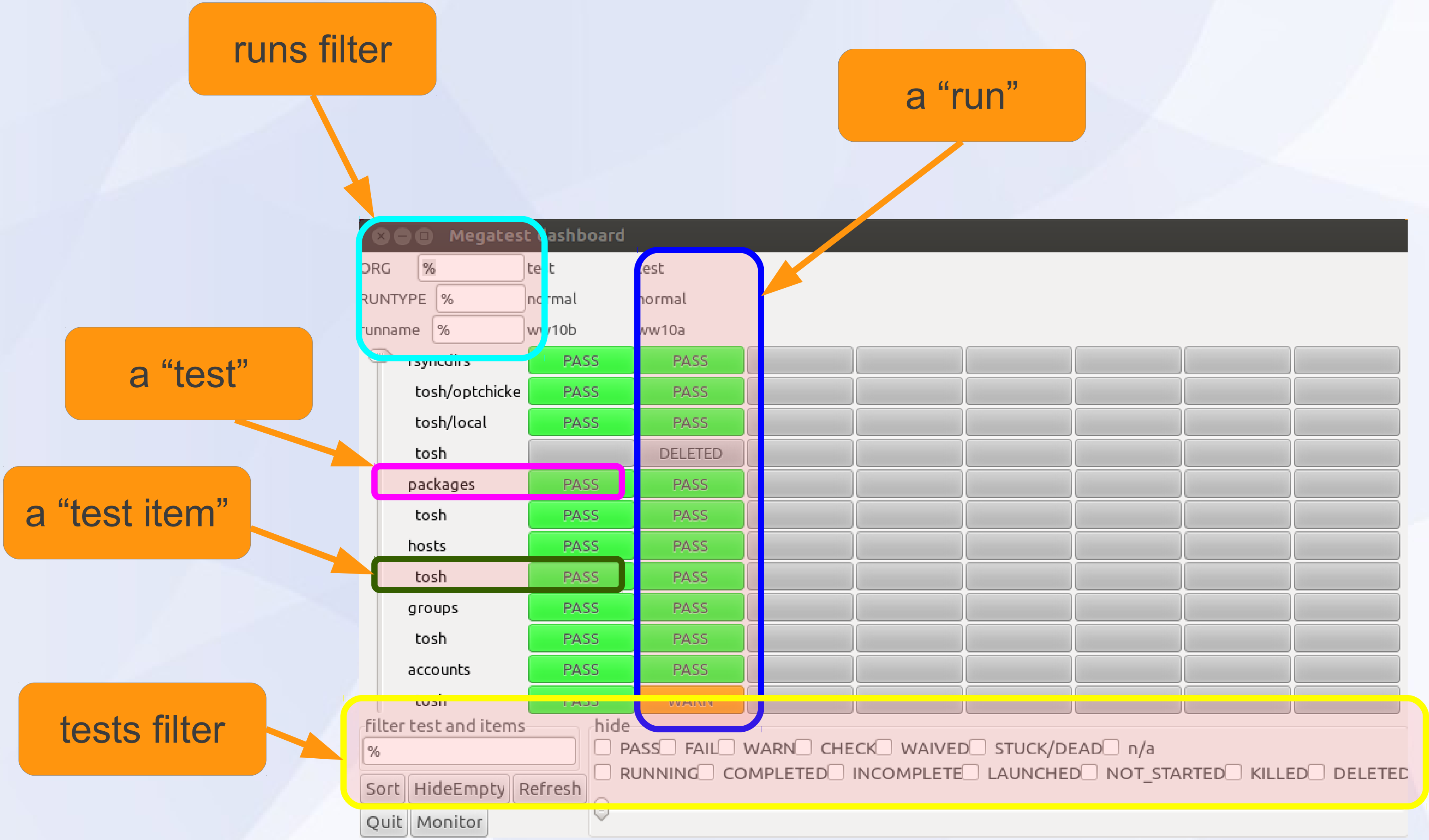
  -list-disks

  -list-targets

  -list-db-targets

  -find-files, -find-paths

dashboard



Megatest Dashboard

ORG	%	test	test
RUNTYPE	%	normal	normal
runname	%	ww10b	ww10a
rsyncdirs		PASS	PASS
tosh/optchicke		PASS	PASS
tosh/local		PASS	PASS
tosh			DELETED
packages		PASS	PASS
tosh		PASS	PASS
hosts		PASS	PASS
tosh		PASS	PASS
groups		PASS	PASS
tosh		PASS	PASS
accounts		PASS	PASS
tosh		PASS	WARN

filter test and items

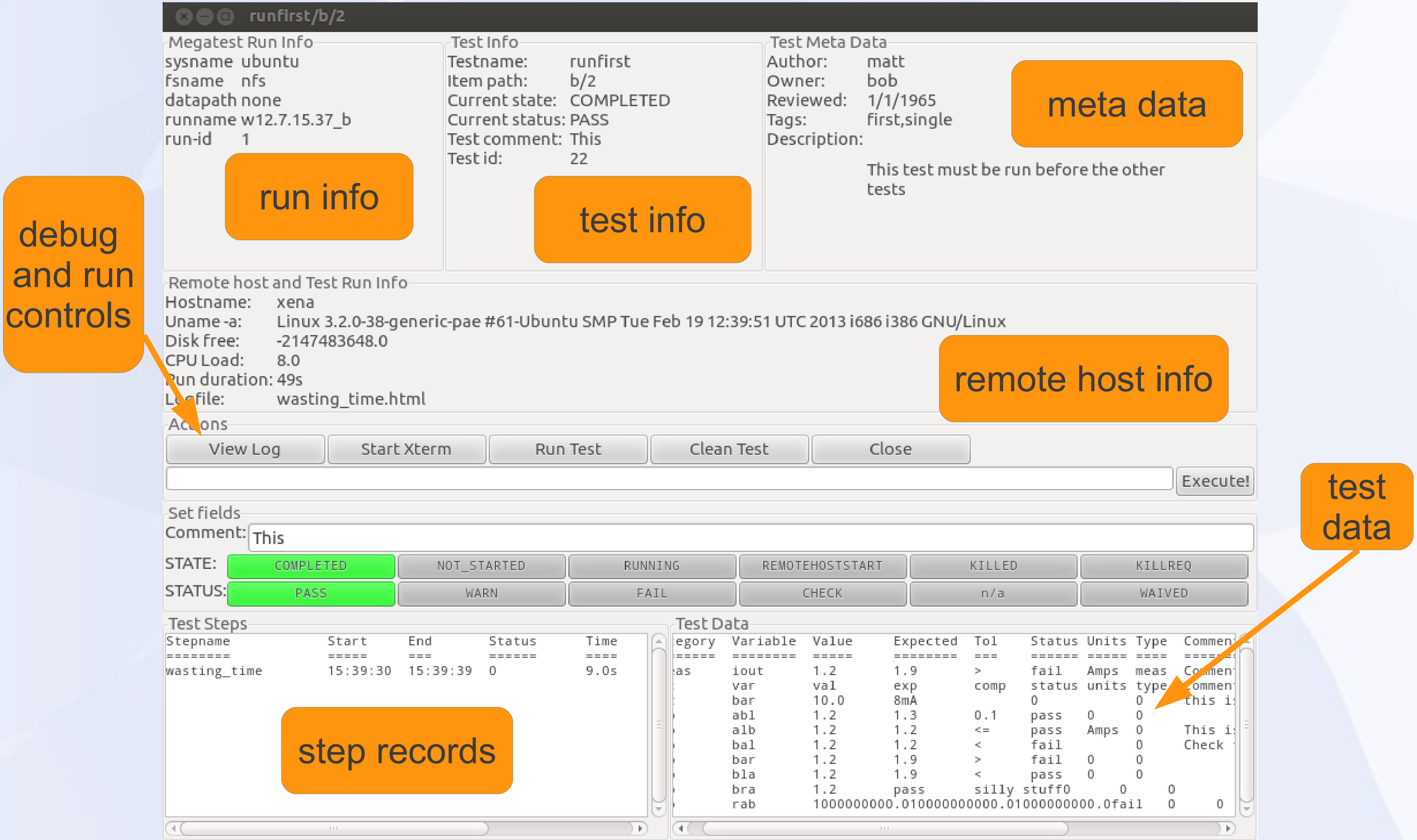
%

hide

□ PASS □ FAIL □ WARN □ CHECK □ WAIVED □ STUCK/DEAD □ n/a

□ RUNNING □ COMPLETED □ INCOMPLETE □ LAUNCHED □ NOT_STARTED □ KILLED □ DELETED

test control panel



runfirst/b/2

Megatest Run Info

sysname ubuntu
fsname nfs
datapath none
runname w12.7.15.37_b
run-id 1

Test Info

Testname: runfirst
Item path: b/2
Current state: COMPLETED
Current status: PASS
Test comment: This
Test id: 22

Test Meta Data

Author: matt
Owner: bob
Reviewed: 1/1/1965
Tags: first,single
Description:

This test must be run before the other tests

Remote host and Test Run Info

Hostname: xena
Uname -a: Linux 3.2.0-38-generic-pae #61-Ubuntu SMP Tue Feb 19 12:39:51 UTC 2013 i686 i386 GNU/Linux
Disk free: -2147483648.0
CPU Load: 8.0
Run duration: 49s
Logfile: wasting_time.html

Actions

Set fields

Comment: This

STATE: COMPLETED | NOT_STARTED | RUNNING | REMOTEHOSTSTART | KILLED | KILLREQ

STATUS: PASS | WARN | FAIL | CHECK | n/a | WAIVED

Test Steps

```
Stepname                Start     End       Status      Time
========                =====     ===       ======      ====
wasting_time            15:39:30  15:39:39  0           9.0s
```

Test Data

```
egory  Variable  Value      Expected  Tol        Status Units Type  Commen
=====  ========  =====      ========  ===        ====== ===== ====  ======
as     iout      1.2        1.9       >          fail   Amps  meas  Commen
       var       val        exp       comp       status units type  Commen
       bar       10.0       8mA                  0            0     this i
       abl       1.2        1.3       0.1        pass   0     0
       alb       1.2        1.2       <=         pass   Amps  0     This i
       bal       1.2        1.2       <          fail         0     Check
       bar       1.2        1.9       >          fail   0     0
       bla       1.2        1.9       <          pass   0     0
       bra       1.2        pass      silly stuff0       0     0
       rab       1000000000.010000000000.01000000000.0fail  0     0
```

meta data

run info

test info

debug and run controls

remote host info

test data

step records

Config File Syntax

The config file syntax was designed to be:

- simple and forgiving to syntax mistakes
- easy to understand and trace where values originated
- expressive enough for complex needs.

	Example	description of the example
Sections	[setup]	Variables defined on subsequent lines will be in the “setup” section
Variables	ABC 1	Variable “ABC” will have the value “1”
[ ] directives	[include a.txt]	include file “a.txt”, see manual for all directives
#{ } text substitutions	#{shell ls $PWD}	replace the #{ … } with the output of the ls $PWD command. Note that newlines are replaced with spaces.

Setup Megatest Area (configs)

- Config files
  - megatest.config
    - Target
      - One or more “keys”
      - Choose carefully! They cannot be changed after your megatest.db is created
    - links area
    - runs disk (can add more over time)
  - runconfigs.config
    - can be empty initially

Example Config Files

megatest.config

```
[fields]
SYSTEM TEXT
RELEASE TEXT

[setup]
# Adjust max_concurrent_jobs to limit how much you load your machines
max_concurrent_jobs 50

# This is your link path, best to set it and then not change it
linktree #{shell realpath #{getenv PWD}/../simplelinks}

# Job tools control how your jobs are launched
[jobtools]
useshell yes
launcher nbfind

# You can override environment variables for all your tests here
[env-override]
EXAMPLE_VAR example value

# As you run more tests you may need to add additional disks
# the names are arbitrary but must be unique
[disks]
disk0 #{shell realpath #{getenv PWD}/../simpleruns}
```

runconfigs.config

```
[default]
ALLTESTS see this variable

# Your variables here are grouped by targets [SYSTEM/RELEASE]
[SYSTEM_val/RELEASE_val]
ANOTHERVAR only defined if target is SYSTEM_val/RELEASE_val
```

Example testconfig

testconfig

```
# Add additional steps here. Format is "stepname script"
[ezsteps]
step1 step1.sh
step2 step2.sh

# Test requirements are specified here
[requirements]
waiton setup
priority 0

# Iteration for your tests are controlled by the items section
[items]
COMPONENT parser datastore transport analyzer

# test_meta is a section for storing additional data
# on your test
[test_meta]
author matt
owner  matt
description An example test
tags tagone,tagtwo
reviewed never
```

Setup Megatest Area (tests)

- Tests
  - tests/<yourfirsttest>/testconfig
- Can use the helper “wizards”

```
megatest -gen-megatest-area
megatest -gen-megatest-test
```

Setup for Run “Flavors”

- runconfigs.config

  ```
  [default]
  VARS here are inherited by all runs

  [some/target]
  VARS here inherited in some/target runs
  ```

- NB// the last specified definition overrides prior definitions.

Setup Tests/Tasks

- A test or task is a set of scripts and data designed to do something or test something.
- Create in tests directory
- Test name limitations
  - No spaces or special characters
  - [a-zA-Z0-9_] and “-” are ok.

The testconfig file [setup]

- [setup]
  runscript scriptname.sh
  - The script must exist in the testconfig directory and be executable
  - Output from the script is NOT captured by Megatest directly
  - The script can be an executable or written in any scripting language

The testconfig file [ezsteps]

- [ezsteps]
  step1name step1script.sh
  - The script “step1script.sch” will be executed and its output redirected to the file step1name.log.
  - If a logpro file step1name.logpro exists it will be used to process the logfile step1name.log and generate the PASS/FAIL/WARN status.

The testconfig file [items]

```
[items]
VAR1 value11 value12 value13 …
VAR2 value21 value22 value23 ...
```

  - This will iterate this test with all possible combinations of VAR1 and VAR2 values.

- Results:
  - value11/value21, value11/value22, value11/value23, value12/value21, value12/value22, value12/value23 ...

The testconfig file [itemstable]

```
[itemstable]
VAR1    value11    value12  …
VAR2    value21    value22  …
```

  - This will iterate over the test with only aligned value combinations.

- Result:
  - value11/value21, value12/value22 …

NOTE: You can combine items and itemstable but they work independently and the result may not be what you expect.

The testconfig file [requirements]

[requirements]

waiton <testname … >

- this test will not be launched until the listed tests are COMPLETED and PASS, WAIVE or SKIP.

jobgroup <groupname>

- this test will be added to the named job group and the relevant max concurrent jobs will apply

toplevel <testname>

- this test will proceed once all it waiton tests are completed with any status.

The testconfig file[test_meta]

- author matt
- owner bob
- description The description can run to multiple lines but subsequent lines must be indented with spaces.
- tags first,single
- reviewed 09/10/2011, by Matt

Megatest Calls in Tests

- -step stepname
  - mark the start or end of a step
- -test-status

  set the state and status of a test
- -setlog logfname

  set the path/filename to the final log relative to the test directory.
- -set-toplog logfname

  set the log for a series of iterated tests

Other Megatest calls

- -summarize-items
  - for an itemized test create a summary html (usually called automatically)
- -m comment
  - insert a comment for this test, can be used with any of the above calls but only one comment is stored per test
- -test-files or -test-paths
  - Use the database to search for files or paths in the test run area

Example Megatest in-test calls

- -step

```
$MT_MEGATEST -step step1 :state start :status
running -setlog step1.html
```

- -test-status

  (Mark a test as completed and trigger a rollup to the parent test of overall status)

```
$MT_MEGATEST -test-status :state COMPLETED :status
AUTO
```

- -test-path

```
export EZFAILPATH2=`$MT_MEGATEST -test-paths -target
  $MT_TARGET :runname $MT_RUNNAME -testpatt
  runfirst/a%`
```

Logpro

- Logpro syntax

Logpro uses scheme calls directly and the full power of scheme is available. However 99% of logpro rule files will not need anything other than the base logpro rules.

Rule	Example	Purpose
expect:error	(expect:error in “Logf” = 0 “Err desc” #/err1/i)	Flags errors matching the pattern err1
expect:ignore	(expect:ignore in “Logf” < 10 “Err desc” #/err2/i)	Ignore errors matching the pattern err2
expect:warning	(expect:warning in “Logf” = 0 “Desc” #/warn1/i)	Lines matching pattern warn1 flagged as warning
expect:required	(expect:required in “Logf” = 1 “Desc” #/reqrd/i)	Line matching pattern reqrd must exit in log file
expect:waive	(expect:waive in “Logf” = 0 “Err desc” #/err3/i)	Waive error matching pattern err3
expect:value	(expect:value in “Logf” 10 1 “Err desc” #/(\d+)/i)	The number matched must be 10 +/- 1
trigger	(trigger “start” #/Start logfile/)	Set trigger “**start**” on line with “Start logfile” string.
section	(section “Logf” “start” “end”)	Section **Logf** starts at trigger **start**, ends at **end**
hook:add	(hook:add “err1” “err1.pl #{msg}”)	On err1 call the err1.pl script with msg as param

Advance Logpro Usage

- Data collection
  - Capturing with logpro
  - Rolling up with Megatest

Direct Access to Megatest Functions

- -repl
- -load

Future Megatest Development

Advanced Topics