Teaching Your Test Framework to Speak LAVA

Tim Orling, Intel Open Source Technology Center

You Already Have Tests

- Wouldn't it be great if:
  - LAVA could auto-discover them?
  - You could run the same commands for a developer as for Continuous Integration?
    - KISS "Keep it simple smarty-pants"
  - You didn't need to rewrite your tests for LAVA?
    - DRY "Don't repeat yourself"
  - You didn't need to write a post-processor to parse your results?

Life Cycle of a Test Case

output
debug
results

- Once the test run has begun, LAVA is just observing

1. Start of Test Case
2. End of Test Case
3. Test Case Result

```
+ lava-test-case ping-test --shell ping -W1 -c1 www.example.com
<LAVA_SIGNAL_STARTTC ping-test>                                          (1)
PING www.example.com (93.184.216.34): 56 data bytes.
--- www.example.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
<LAVA_SIGNAL_ENDTC ping-test>                                            (2)
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=ping-test RESULT=fail>                (3)
```

```
Received signal: <STARTTC> ping-test
Received signal: <ENDTC> ping-test
Received signal: <TESTCASE> TEST_CASE_ID=ping-test RESULT=fail
```

```
case: ping-test
case_id: 12345678
definition: 1_smoke-tests
result: fail
```

Emit Signals LAVA Needs to See

output
debug
results

- Once the test run has begun, LAVA is just observing

❶ Start of Test Case

```
setup() {
    print (’<LAVA_SIGNAL_STARTTC $testname>’)
}
```

❷ End of Test Case

```
teardown() {
    print (‘<LAVA_SIGNAL_ENDTC $testname>’)
```

❸ Test Case Result

```
    print (‘<LAVA_SIGNAL_TESTCASE \
            TEST_CASE_ID=$testname \
            RESULT=$result’>)
}
```

```
+ my-test-runner my-test-case
<LAVA_SIGNAL_STARTTC my-test-case>          ❶
assert 1 == 1
PASS
<LAVA_SIGNAL_ENDTC my-test-case>            ❷
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=my-test-case RESULT=pass>  ❸
```

```
Received signal: <STARTTC> my-test-case
Received signal: <ENDTC> my-test-case
Received signal: <TESTCASE> TEST_CASE_ID=my-test-case RESULT=pass
```

```
case: ping-test
case_id: 12345678
definition: 1_smoke-tests
result: pass
```

Measurements and Units

output
debug
results

- LAVA can also support results which have:

❶ Measurements (a numeric value)

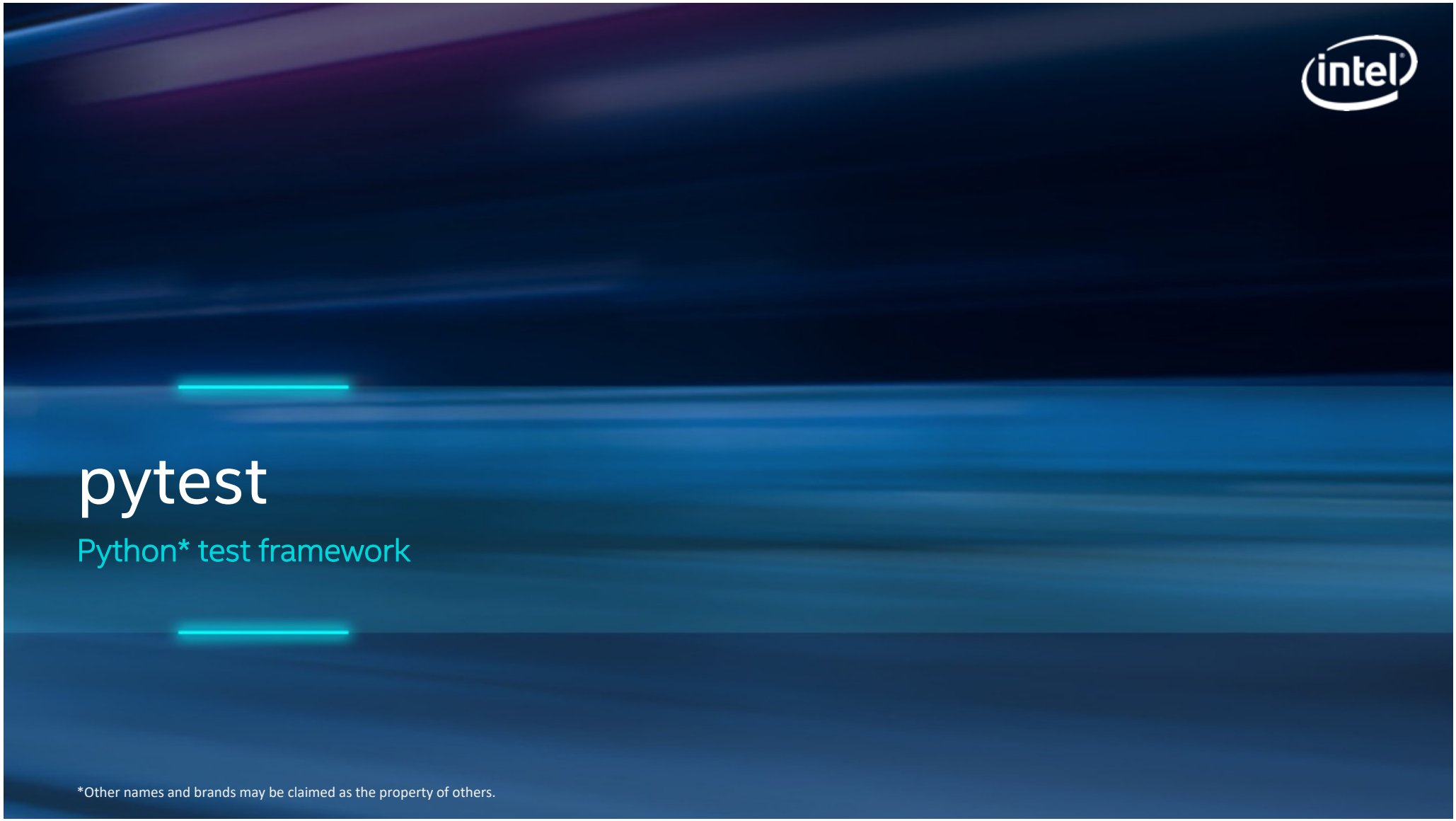
```
+ my-test-runner pi-test
<LAVA_SIGNAL_STARTTC pi-test>
pi = 3.14159
PASS
<LAVA_SIGNAL_ENDTC pi-test>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=pi-test RESULT=pass MEASUREMENT=3.14159>
```

❶

❷ Units (optional)

```
+ my-test-runner speed-test
<LAVA_SIGNAL_STARTTC speed-test>
speed = 60 kph
PASS
<LAVA_SIGNAL_ENDTC speed-test>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=speed-test RESULT=pass MEASUREMENT=60 UNITS=kph>
```

❶ ❷


intel
pytest
Python* test framework
*Other names and brands may be claimed as the property of others.

conftest.py

- At the root of the directory where you call ‘pytest’

```
# conftest.py
#
# Copyright (C) 2018 Intel Corporation
# SPDX-License-Identifier: MIT

def lava_result_convert(pytest_outcome):
    """ Convert the pytest outcome to the string expected by LAVA.
    """
    if pytest_outcome is 'passed':
        return 'pass'
    elif pytest_outcome is 'skipped':
        return 'pass'
    elif pytest_outcome is 'xfailed':
        return 'pass'
    else:
        return 'fail'
```

conftest.py (cont'd)

- Allows you to override pytest defaults

```
def pytest_report_teststatus(report):
    """ Insert strings that LAVA expects to capture test results.
    """
    # Get pytest test name and remove the 'test_' prefix
    test_name = report.location[2][5:]

    if report.when is 'setup':
        print('\n')
        print('<LAVA_SIGNAL_STARTTC ' + test_name + '>')
    elif report.when is 'call':
        test_result = lava_result_convert(report.outcome)

        print('\n')
        print('<LAVA_SIGNAL_ENDTC ' + test_name + '>')
        print('<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=' + test_name +
              ' RESULT=' + test_result + '>')
```

A Simple LAVA Test Definition with pytest


```
metadata:
    format: Lava-Test Test Definition 1.0
    name: python3-six-tests
    description: "Run the python3-six \
                  pytests"
    os:
        - oe
    scope:
        - functional
    devices:
        - x86

run:
    steps:
        - lava-test-case python3-six-tests \
          --shell pytest
```

```
metadata:
    format: Lava-Test Test Definition 1.0
    name: python3-six-tests-lava
    description: "Run the python3-six \
                  pytests, with LAVA \
                  signals enabled."
    os:
        - oe
    scope:
        - functional
    devices:
        - x86

run:
    steps:
        - pytest
```

pytest Output (before)

output
debug
results

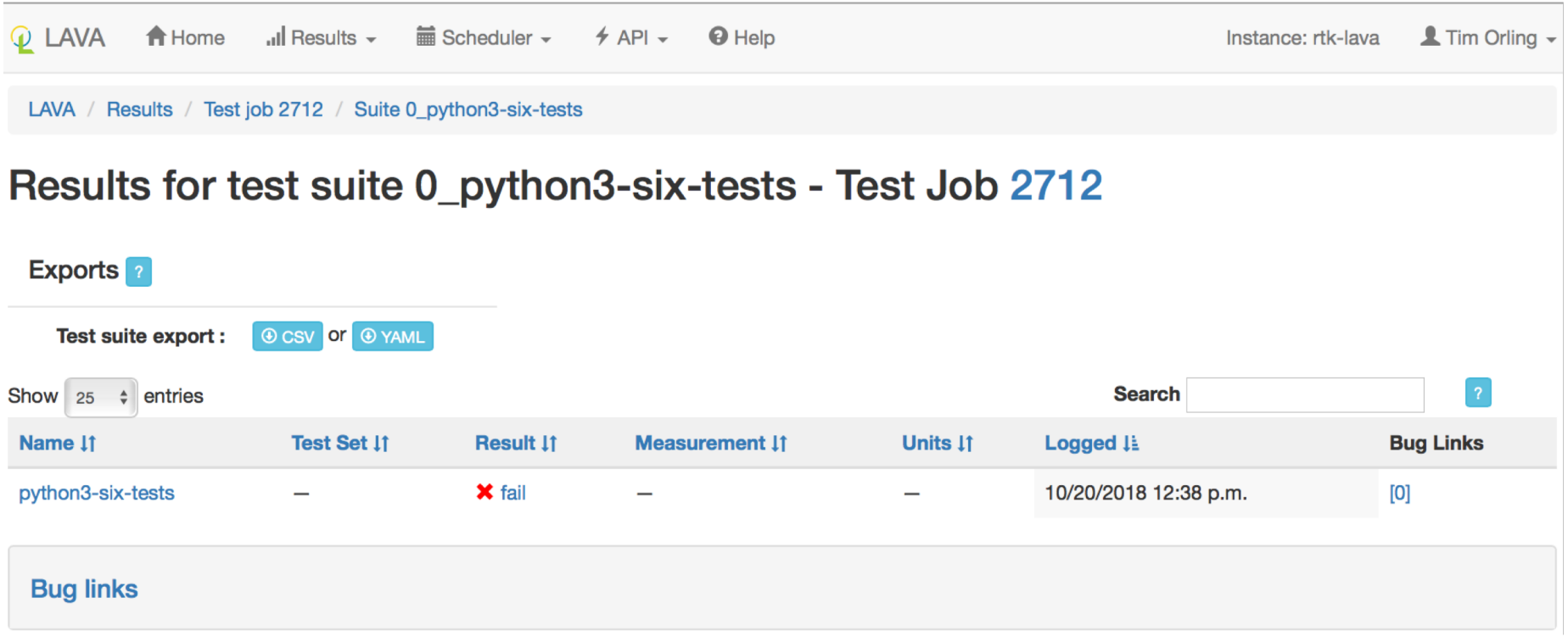
```
<LAVA_SIGNAL_STARTTC python3-six-tests>
Received signal: <STARTTC> python3-six-tests
============================= test session starts ==============================
platform linux -- Python 3.5.6, pytest-3.4.2, py-1.6.0, pluggy-0.6.0
rootdir: /lava-2712/0/tests/0_python3-six-tests, inifile: setup.cfg
collected 195 items
test_six.py .............................s.................sssssss [ 30%]
sssssss.................................................................. [ 67%]
.....................................................s.....F....          [100%]
=================================== FAILURES ===================================
_____________________________ test_add_metaclass_nested _____________________________
def test_add_metaclass_nested():
[…]
test_six.py:895: AssertionError
============== 1 failed, 177 passed, 17 skipped in 3.07 seconds ==============
<LAVA_SIGNAL_ENDTC python3-six-tests>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=python3-six-tests RESULT=fail>
Received signal: <ENDTC> python3-six-tests
Received signal: <TESTCASE> TEST_CASE_ID=python3-six-tests RESULT=fail
case: python3-six-tests
case_id: 47840
definition: 0_python3-six-tests
result: fail
```

pytest Output (after)

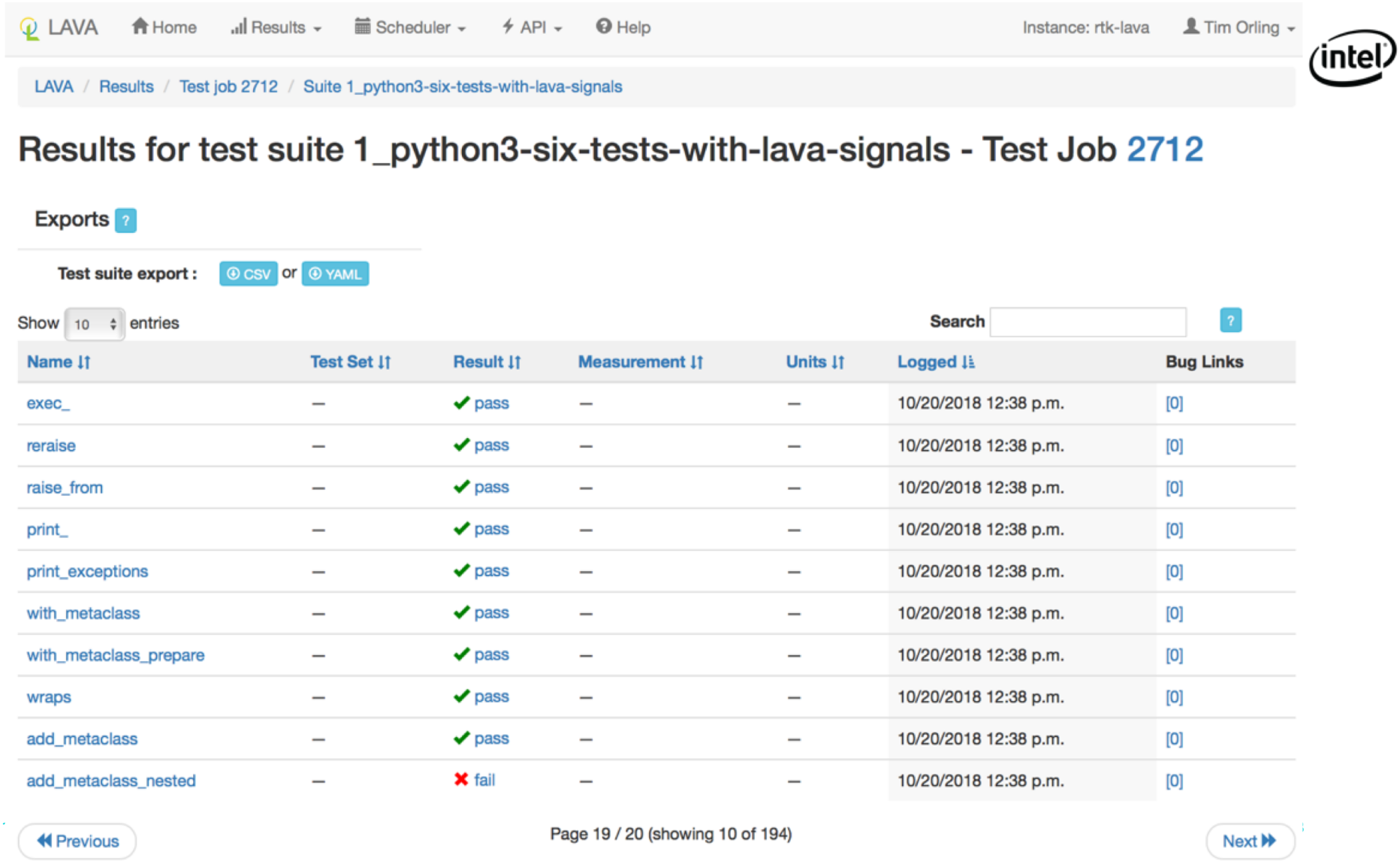
output
debug
results

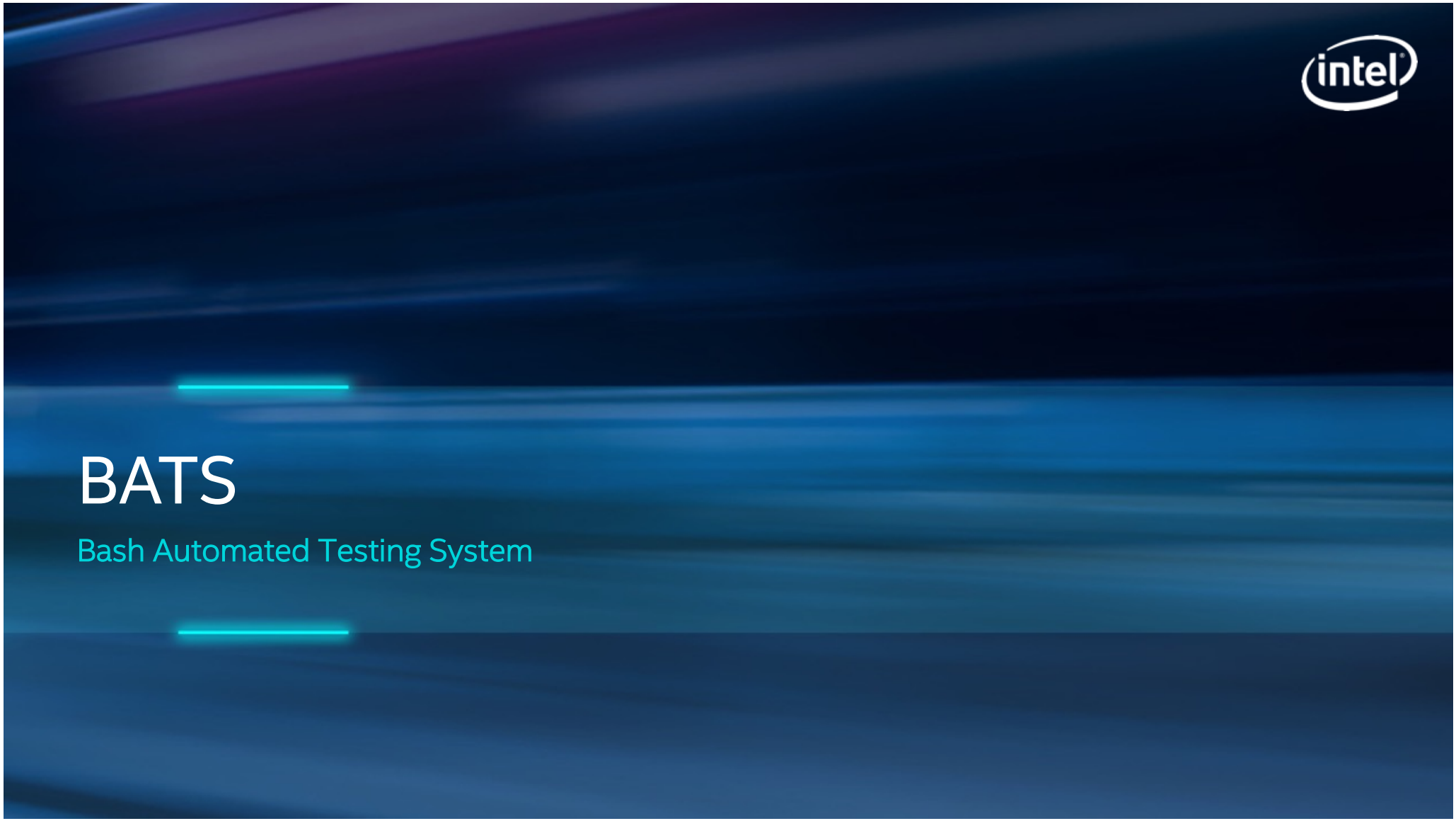
```
[…]
<LAVA_SIGNAL_STARTTC add_metaclass>
Received signal: <STARTTC> add_metaclass
.
<LAVA_SIGNAL_ENDTC add_metaclass>
Received signal: <ENDTC> add_metaclass
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=add_metaclass RESULT=pass>
Received signal: <TESTCASE> TEST_CASE_ID=add_metaclass RESULT=pass
case: add_metaclass
case_id: 48265
definition: 1_python3-six-tests-with-lava-signals
result: pass
<LAVA_SIGNAL_STARTTC add_metaclass_nested>
Received signal: <STARTTC> add_metaclass_nested
F
<LAVA_SIGNAL_ENDTC add_metaclass_nested>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=add_metaclass_nested RESULT=fail>
Received signal: <ENDTC> add_metaclass_nested
Received signal: <TESTCASE> TEST_CASE_ID=add_metaclass_nested RESULT=fail
case: add_metaclass_nested
case_id: 48266
definition: 1_python3-six-tests-with-lava-signals
result: fail
[…]
```

pytest Results (before)



pytest Results (after)




intel
BATS
Bash Automated Testing System

libexec/bats-core/bats

```
diff --git a/libexec/bats-core/bats b/libexec/bats-core/bats
index d3e2f47..7a4976c 100755
--- a/libexec/bats-core/bats
+++ b/libexec/bats-core/bats
@@ -20,7 +20,7 @@ usage() {
   while IFS= read -r line; do
     printf '%s\n' "$line"
   done <<END_OF_HELP_TEXT
-Usage: $cmd [-cr] [-f <regex>] [-p | -t] <test>...
+Usage: $cmd [-cr] [-f <regex>] [-l | -p | -t] <test>...
        $cmd [-h | -v]

   <test> is the path to a Bats test file, or the path to a directory
```

libexec/bats-core/bats (cont'd)

```
@@ -29,6 +29,7 @@ Usage: $cmd [-cr] [-f <regex>] [-p | -t] <test>...
   -c, --count      Count the number of test cases without running any tests
   -f, --filter     Filter test cases by names matching the regular expression
   -h, --help       Display this help message
+  -l, --lava       Emit signals for LAVA integration
   -p, --pretty     Show results in pretty format (default for terminals)
   -r, --recursive  Include tests in subdirectories
   -t, --tap        Show results in TAP format
@@ -80,8 +81,9 @@ done
 set -- "${arguments[@]}"
 arguments=()

-unset flags pretty recursive
+unset flags lava pretty recursive
 flags=()
+lava=
 pretty=
 recursive=
 if [[ -z "${CI:-}" && -t 0 && -t 1 ]] && command -v tput >/dev/null; then
```

libexec/bats-core/bats (cont'd)

```
@@ -109,6 +111,9 @@ while [[ "$#" -ne 0 ]]; do
    -r|--recursive)
      recursive=1
      ;;
+   -l|--lava)
+     lava=1
+     ;;
    -t|--tap)
      pretty=
      ;;
@@ -159,6 +164,8 @@ fi
 set -o pipefail execfail
 if [[ -z "$pretty" ]]; then
    exec "$command" "${flags[@]}" "${filenames[@]}"
+elif [[ "$lava" -eq 1 ]]; then
+  exec "$command" -x "${flags[@]}" "${filenames[@]}" | bats-format-lava-stream
 else
    exec "$command" -x "${flags[@]}" "${filenames[@]}" | bats-format-tap-stream
 fi
```

libexec/bats-core/bats-format-lava-stream

```
--- bats-format-tap-stream   2018-10-20 10:15:06.130047652 -0700
+++ bats-format-lava-stream  2018-10-20 10:12:46.886044973 -0700
@@ -9,6 +9,9 @@
   index=0
   failures=0
   skipped=0
+  result=
+  measurement=
+  units=
   name=
   count_column_width=$(( ${#count} * 2 + 2 ))
 else
@@ -27,6 +30,7 @@

 begin() {
   go_to_column 0
+  setup
   buffer_with_truncation $(( $count_column_left - 1 )) '   %s' "$name"
   clear_to_end_of_line
   go_to_column $count_column_left
```

libexec/bats-core/bats-format-lava-stream


```
@@ -37,6 +41,7 @@
 pass() {
   go_to_column 0
   buffer ' ✓ %s' "$name"
+  result="pass"
   advance
 }

@@ -47,6 +52,7 @@
   fi
   go_to_column 0
   buffer ' - %s (skipped%s)' "$name" "$reason"
+  result="pass" # lava only has pass/fail
   advance
 }
```

libexec/bats-core/bats-format-lava-stream

```
@@ -54,6 +60,7 @@
    go_to_column 0
    set_color 1 bold
    buffer ' ✗ %s' "$name"
+   result="fail"
    advance
  }

@@ -63,6 +70,11 @@
    clear_color
  }

+setup() {
+   escaped_name="${name// /_}"
+   buffer '\n<LAVA_SIGNAL_STARTTC %s>\n' "$escaped_name"
+}
+
  summary() {
    buffer '\n%d test' "$count"
    if [[ "$count" -ne 1 ]]; then
```

libexec/bats-core/bats-format-lava-stream


```
@@ -81,6 +93,20 @@
   buffer '\n'
 }

+teardown() {
+  escaped_name="${name// /_}"
+  buffer '\n<LAVA_SIGNAL_ENDTC %s>\n' "$escaped_name"
+  # It is valid to have a measurement without units, but not a units without a measuremnt
+  if [[ "$measurement" != "" ]] && [[ "$units" != "" ]]; then
+    buffer '\n<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=% MEASUREMENT=%s UNITS=%s>\n' "$escaped_name"
```

A Simple LAVA Test Definition with BATS

```
metadata:
    format: Lava-Test Test Definition 1.0
    name: bats-core-test
    description: "Run the bats-core tests"
    os:
        - debian
    scope:
        - functional
    devices:
        - qemu

run:
    steps:
        - lava-test-case bats-core-tap \
          --shell bin/bats -t test
```

```
metadata:
    format: Lava-Test Test Definition 1.0
    name: bats-core-test-lava
    description: "Run the bats-core tests, \
                  with LAVA signals enabled."
    os:
        - debian
    scope:
        - functional
    devices:
        - qemu

run:
    steps:
        - bin/bats -l test
```

BATS Output (before)

output
debug
results

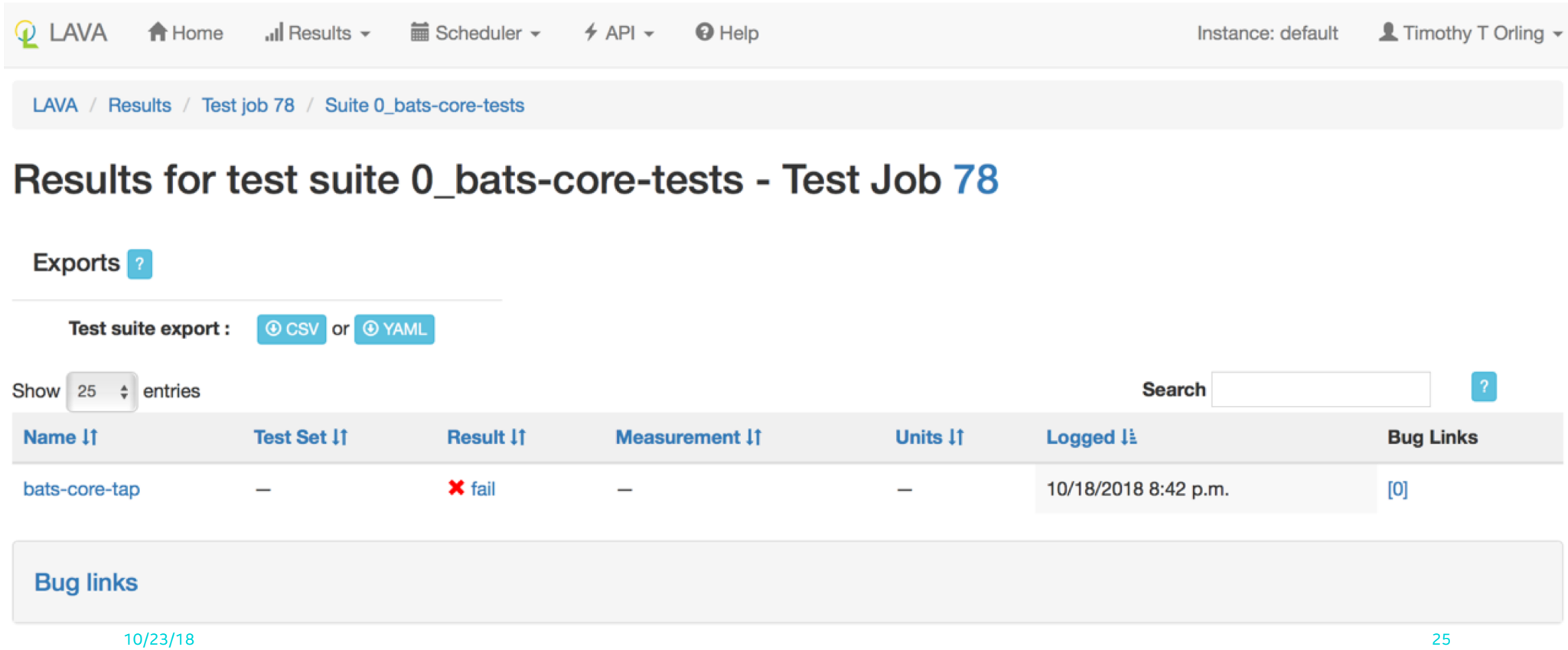
```
<LAVA_SIGNAL_STARTTC bats-core-tap>
Received signal: <STARTTC> bats-core-tap
1..70
ok 1 no arguments prints message and usage instructions
[…]
not ok 56 #113: set BATS_ROOT when /bin is a symlink to /usr/bin
# (from function `setup' in test file test/root.bats, line 21)
# `\"$BATS_ROOT/install.sh\" \"opt/bats-core\"' failed
# /tmp/bats.869.src: line 21: /lava-78/0/tests/0_bats-core-tests/install.sh: No such file or directory
not ok 57 set BATS_ROOT with extreme symlink resolution
# (from function `setup' in test file test/root.bats, line 21)
# `\"$BATS_ROOT/install.sh\" \"opt/bats-core\"' failed
# /tmp/bats.869.src: line 21: /lava-78/0/tests/0_bats-core-tests/install.sh: No such file or directory
[…]
ok 70 --filter can handle regular expressions that start with ^
<LAVA_SIGNAL_ENDTC bats-core-tap>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=bats-core-tap RESULT=fail>
Received signal: <ENDTC> bats-core-tap
Received signal: <TESTCASE> TEST_CASE_ID=bats-core-tap RESULT=fail
case: bats-core-tap
case_id: 642
definition: 0_bats-core-tests
result: fail
```

BATS Output (after)

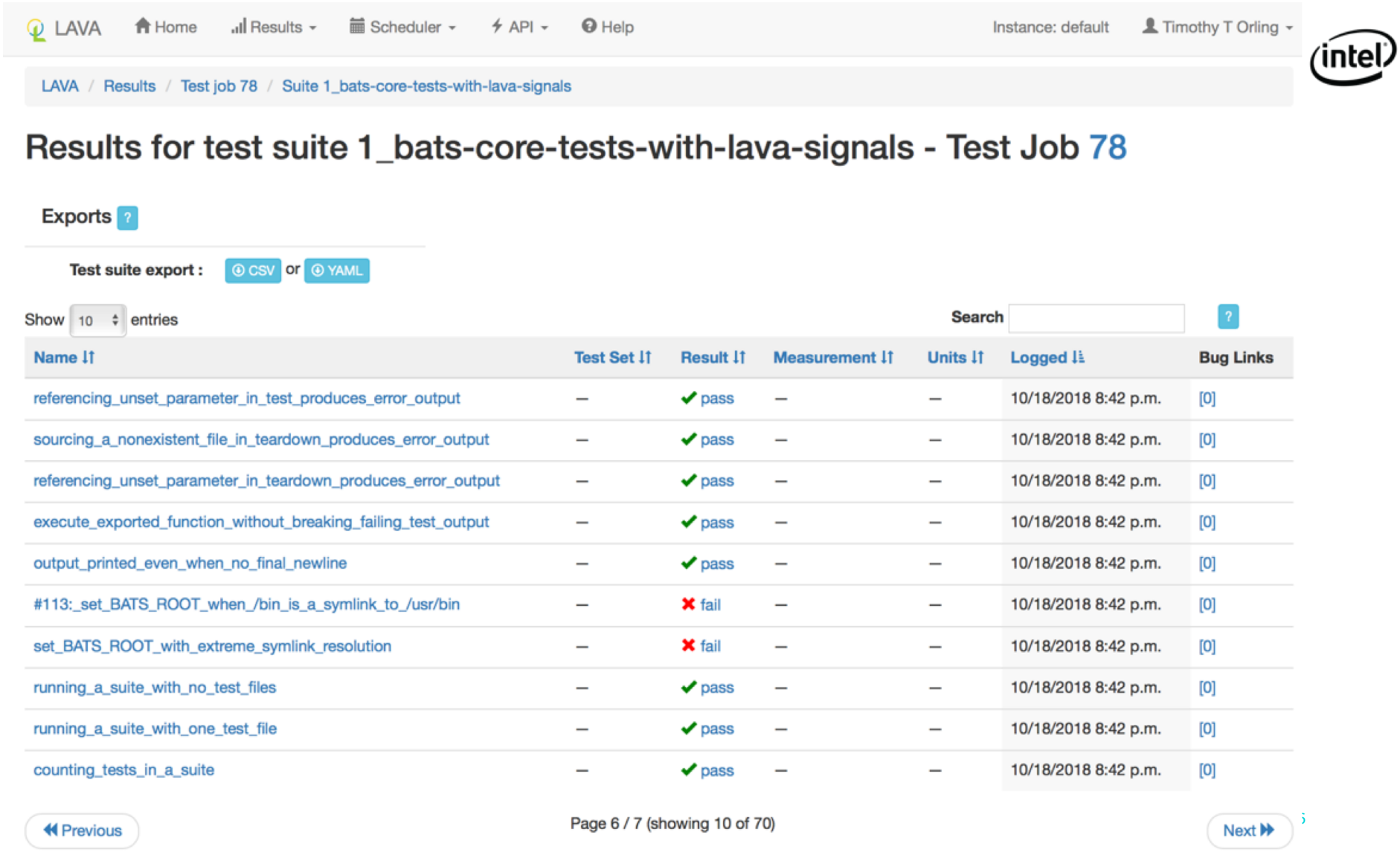
output
debug
results

```
[…]
<LAVA_SIGNAL_STARTTC output_printed_even_when_no_final_newline>
Received signal: <STARTTC> output_printed_even_when_no_final_newline
output printed even when no final newline
✓ output printed even when no final newline
<LAVA_SIGNAL_ENDTC output_printed_even_when_no_final_newline>
Received signal: <ENDTC> output_printed_even_when_no_final_newline
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=output_printed_even_when_no_final_newline RESULT=pass>
Received signal: <TESTCASE> TEST_CASE_ID=output_printed_even_when_no_final_newline RESULT=pass
case: output_printed_even_when_no_final_newline
case_id: 698
definition: 1_bats-core-tests-with-lava-signals
result: pass
<LAVA_SIGNAL_STARTTC #113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin>
Received signal: <STARTTC> #113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin
 #113: set BATS_ROOT when /bin is a symlink to /usr/bin
✗ #113: set BATS_ROOT when /bin is a symlink to /usr/bin
<LAVA_SIGNAL_ENDTC #113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin>
Received signal: <ENDTC> #113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=#113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin RESULT=fail>
Received signal: <TESTCASE> TEST_CASE_ID=#113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin RESULT=fail
case: #113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin
case_id: 699
definition: 1_bats-core-tests-with-lava-signals
result: fail
[…]
```

BATS Results (before)



BATS Results (after)



LAVA Home Results Scheduler API Help Instance: default Timothy T Orling

LAVA / Results / Test job 78 / Suite 1_bats-core-tests-with-lava-signals

Results for test suite 1_bats-core-tests-with-lava-signals - Test Job 78

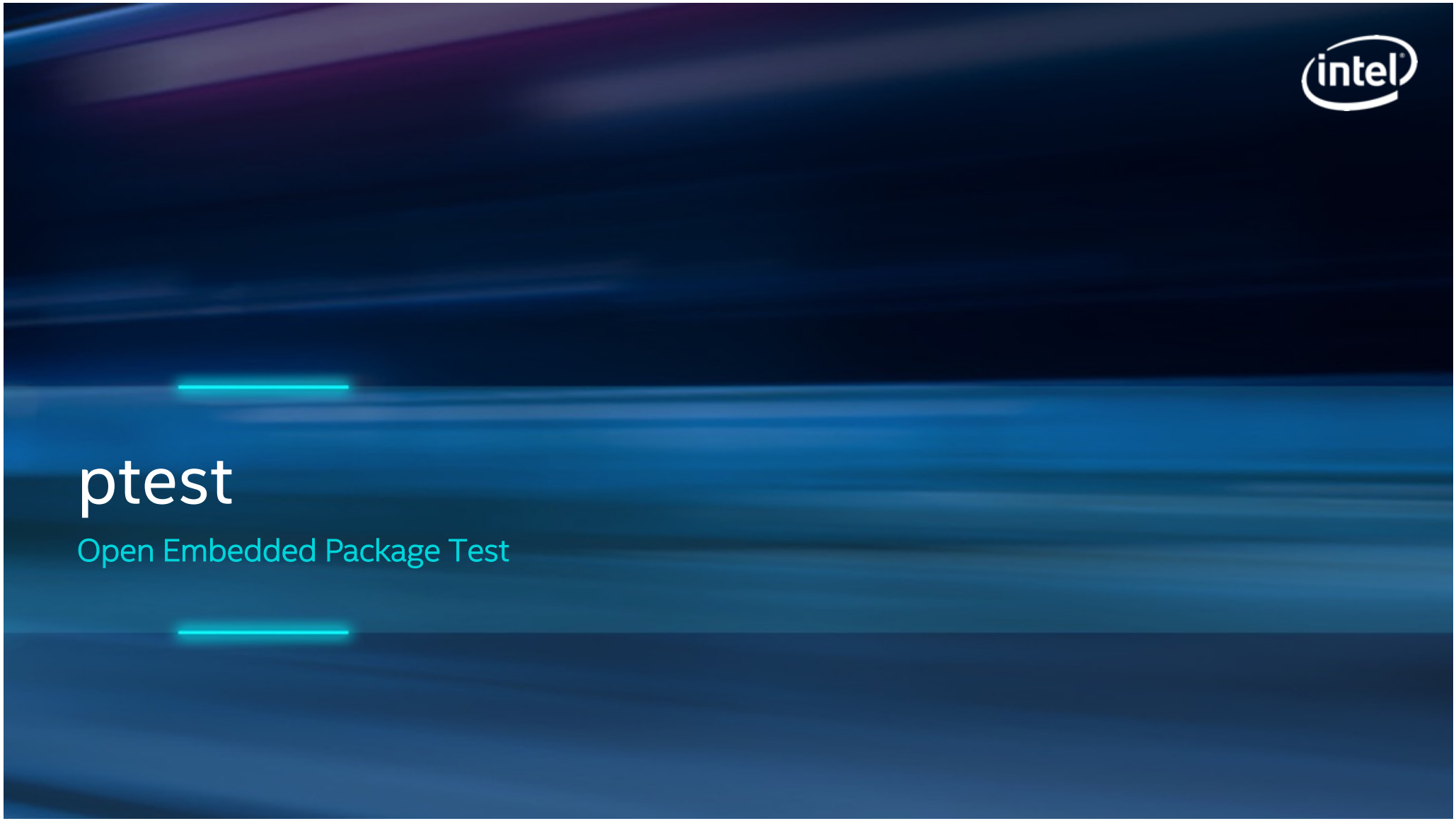
Exports

Test suite export : CSV or YAML

Show 10 entries Search

Name	Test Set	Result	Measurement	Units	Logged	Bug Links
referencing_unset_parameter_in_test_produces_error_output	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
sourcing_a_nonexistent_file_in_teardown_produces_error_output	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
referencing_unset_parameter_in_teardown_produces_error_output	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
execute_exported_function_without_breaking_failing_test_output	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
output_printed_even_when_no_final_newline	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
#113:_set_BATS_ROOT_when_/bin_is_a_symlink_to_/usr/bin	–	fail	–	–	10/18/2018 8:42 p.m.	[0]
set_BATS_ROOT_with_extreme_symlink_resolution	–	fail	–	–	10/18/2018 8:42 p.m.	[0]
running_a_suite_with_no_test_files	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
running_a_suite_with_one_test_file	–	pass	–	–	10/18/2018 8:42 p.m.	[0]
counting_tests_in_a_suite	–	pass	–	–	10/18/2018 8:42 p.m.	[0]

Previous Page 6 / 7 (showing 10 of 70) Next


intel
ptest
Open Embedded Package Test

ptest-runner2/flags.h

```
diff --git a/flags.h b/flags.h
new file mode 100644
index 0000000..8595138
--- /dev/null
+++ b/flags.h
@@ -0,0 +1,11 @@
+/* Copyright (c) 2018 Intel Corporation */
+/* SPDX-License-Identifier: GPL-2.0 */
+
+/* Flag bit definitions */
+
+#ifndef __FLAGS_H__
+#define __FLAGS_H__
+
+#define LAVA_SIGNAL_ENABLE     (0x0001)
+
+#endif                         /* __FLAGS_H__ */
```

ptest-runner2/main.c

```
diff --git a/main.c b/main.c
index 83600b7..3a2158f 100644
--- a/main.c
+++ b/main.c
@@ -36,6 +36,7 @@
 #endif

 #include "utils.h"
+#include "flags.h"

 #define DEFAULT_DIRECTORY "/usr/lib"
 #define DEFAULT_TIMEOUT 300
@@ -44,7 +45,7 @@ static inline void
 print_usage(FILE *stream, char *progname)
 {
        fprintf(stream, "Usage: %s [-d directory] [-e exclude] [-l list] [-t timeout]"
-                       " [-x xml-filename] [-h] [ptest1 ptest2 ...]\n", progname);
+                       " [-x xml-filename] [-L] [-h] [ptest1 ptest2 ...]\n", progname);
 }

 int
```

ptest-runner2/main.c (cont'd)

```
@@ -70,8 +71,9 @@ main(int argc, char *argv[])
        opts.timeout = DEFAULT_TIMEOUT;
        opts.ptests = NULL;
        opts.xml_filename = NULL;
+       opts.flags = 0;

-       while ((opt = getopt(argc, argv, "d:e:lt:x:h")) != -1) {
+       while ((opt = getopt(argc, argv, "d:e:lt:x:Lh")) != -1) {
                switch (opt) {
                        case 'd':
                                free(opts.directory);
@@ -118,6 +120,11 @@ main(int argc, char *argv[])
                                opts.xml_filename = strdup(optarg);
                                CHECK_ALLOCATION(opts.xml_filename, 1, 1);
                        break;
+                       case 'L':
+                               // set LAVA signal mode
+                               opts.flags |= LAVA_SIGNAL_ENABLE;
+                               fprintf(stdout, "LAVA_SIGNAL_ENABLE == %d\n", opts.flags);
+                       break;
                        default:
                                print_usage(stdout, argv[0]);
                                exit(1);
```

ptest-runner2/utils.c

```
diff --git a/utils.c b/utils.c
index ed2eff7..0fd1da6 100644
--- a/utils.c
+++ b/utils.c
@@ -39,6 +39,7 @@

 #include "ptest_list.h"
 #include "utils.h"
+#include "flags.h"

 #define GET_STIME_BUF_SIZE 1024
 #define WAIT_CHILD_POLL_TIMEOUT_MS 200
@@ -358,6 +359,7 @@ run_ptests(struct ptest_list *head, const struct ptest_options opts,
                fprintf(fp, "START: %s\n", progname);
                PTEST_LIST_ITERATE_START(head, p);
                        char *ptest_dir = strdup(p->run_ptest);
+                       char *ptest = strdup(p->ptest);
                        if (ptest_dir == NULL) {
                                rc = -1;
                                break;
```

ptest-runner2/utils.c (cont'd)

```
@@ -376,6 +378,11 @@ run_ptests(struct ptest_list *head, const struct ptest_options opts,
                                int fds[2]; fds[0] = pipefd_stdout[0]; fds[1] = pipefd_stderr[0];
                                FILE *fps[2]; fps[0] = fp; fps[1] = fp_stderr;

+                               char result[5]; // pass\0, fail\0, skip\0
+
+                               if (opts.flags & LAVA_SIGNAL_ENABLE) {
+                                       fprintf(stdout, "<LAVA_SIGNAL_STARTTC %s>\n", ptest);
+                               }
                                fprintf(fp, "%s\n", get_stime(stime, GET_STIME_BUF_SIZE));
                                fprintf(fp, "BEGIN: %s\n", ptest_dir);
```

ptest-runner2/utils.c (cont'd)

```
@@ -389,6 +396,14 @@ run_ptests(struct ptest_list *head, const struct ptest_options opts,

                                fprintf(fp, "END: %s\n", ptest_dir);
                                fprintf(fp, "%s\n", get_stime(stime, GET_STIME_BUF_SIZE));
+                               if (opts.flags & LAVA_SIGNAL_ENABLE) {
+                                       if (status)
+                                               sprintf(result, "fail");
+                                       else
+                                               sprintf(result, "pass");
+                                       fprintf(stdout, "<LAVA_SIGNAL_ENDTC %s>\n", ptest);
+                                       fprintf(stdout, "<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=%s \
+                                               RESULT=%s>\n", ptest, result);
+                               }
                        }
                PTEST_LIST_ITERATE_END;
                fprintf(fp, "STOP: %s\n", progname);
```

ptest-runner2/utils.h

```
diff --git a/utils.h b/utils.h
index ee85163..06d4c10 100644
--- a/utils.h
+++ b/utils.h
@@ -37,9 +37,9 @@ struct ptest_options {
         int timeout;
         char **ptests;
         char *xml_filename;
+        unsigned int flags;
 };

-
 extern void check_allocation1(void *, size_t, char *, int, int);
 extern struct ptest_list *get_available_ptests(const char *);
 extern int print_ptests(struct ptest_list *, FILE *);
```

A Simple LAVA Test Definition with ptest


```
metadata:
    format: Lava-Test Test Definition 1.0
    name: ptest-tests
    description: "Run enabled Package Test (ptest) \
                  tests"
    os:
        - oe
    scope:
        - functional
    devices:
        - qemu

run:
    steps:
        - lava-test-case ptest-tests --shell ptest-runner \
          libcapture-tiny-perl libcgi-perl \
          libcrypt-openssl-rsa-perl libdbd-sqlite-perl \
          libdbi-perl libdigest-hmac-perl
```

```
metadata:
    format: Lava-Test Test Definition 1.0
    name: ptest-tests-with-lava-signals
    description: "Run enabled Package Test (ptest) \
                  tests with lava signals enabled.."
    os:
        - oe
    scope:
        - functional
    devices:
        - qemu

run:
    steps:
        - ptest-runner -L libcapture-tiny-perl \
          libcgi-perl libcrypt-openssl-rsa-perl \
          libdbd-sqlite-perl libdbi-perl \
          libdigest-hmac-perl
```

ptest Output (before)

output
debug
results

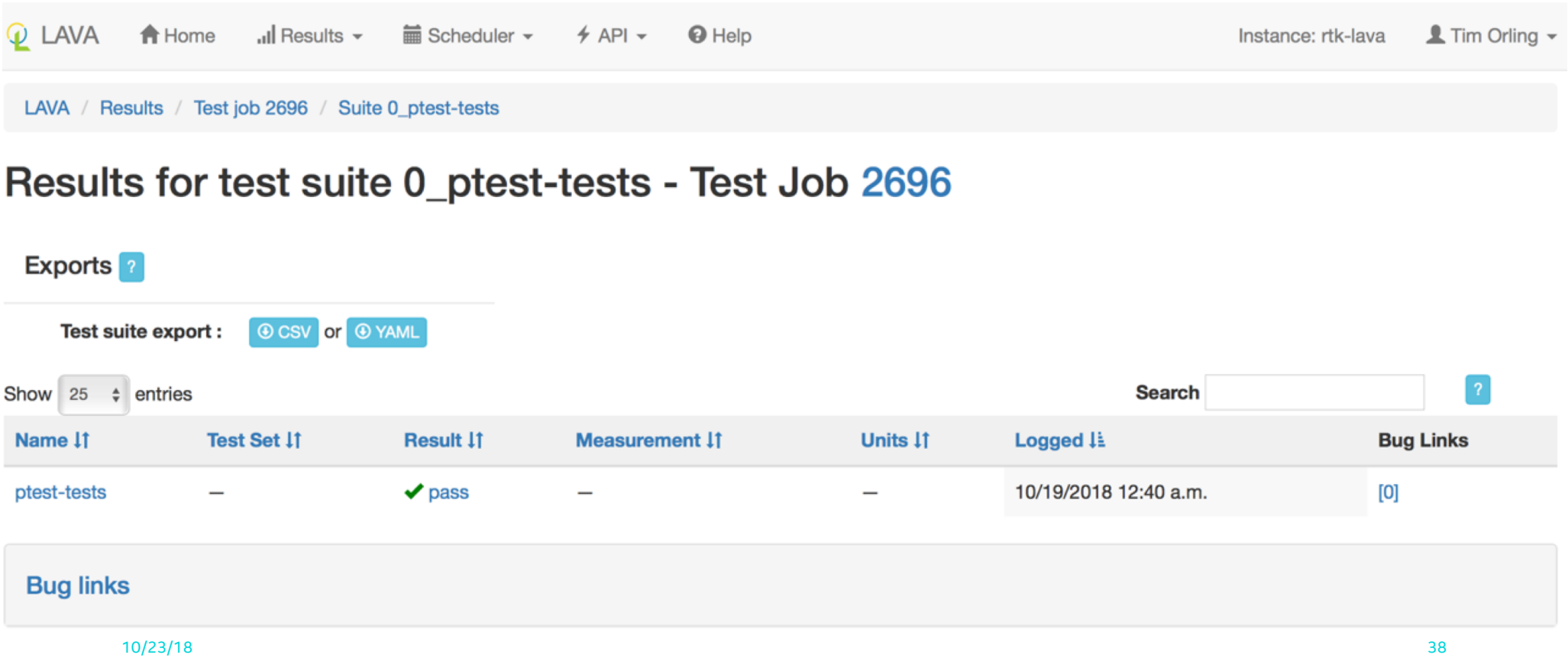
```
<LAVA_SIGNAL_STARTTC ptest-tests>
Received signal: <STARTTC> ptest-tests
START: ptest-runner
2018-10-18T23:16
BEGIN: /usr/lib/libcapture-tiny-perl/ptest
ok 1 - STDERR is tied
ok 2 - capture|perl|stderr|multiline - got STDOUT
ok 3 - capture|perl|stderr|multiline - got STDERR
ok 4 - capture|perl|stderr|short - got STDOUT
[…]
ok 13
PASS: t/rfc2202
END: /usr/lib/libdigest-hmac-perl/ptest
2018-10-18T23:19
STOP: ptest-runner
<LAVA_SIGNAL_ENDTC ptest-tests>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=ptest-tests RESULT=pass>
Received signal: <ENDTC> ptest-tests
Received signal: <TESTCASE> TEST_CASE_ID=ptest-tests RESULT=pass
case: ptest-tests
case_id: 47268
definition: 0_ptest-tests
result: pass
```

ptest Output (after)

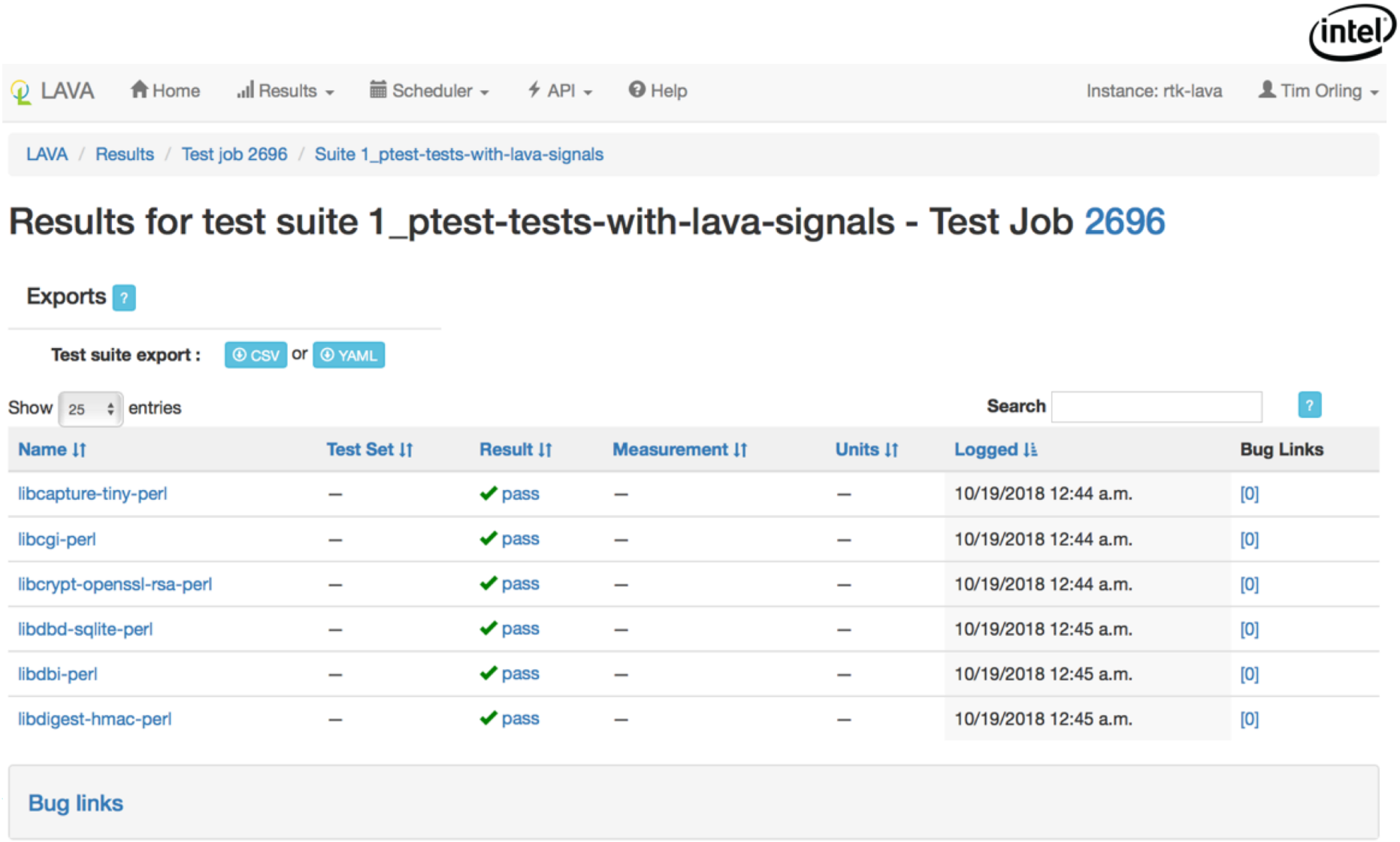
output
debug
results

```
[…]
<LAVA_SIGNAL_STARTTC output_printed_even_when_no_final_newline>
Received signal: <STARTTC> libcapture-tiny-perl
LAVA_SIGNAL_STARTTC libcapture-tiny-perl>
2018-10-18T23:19
BEGIN: /usr/lib/libcapture-tiny-perl/ptest
ok 1 - STDERR is tied
ok 2 - capture|sys|empty|short - got STDOUT
[…]
ok 19 - no file descriptors leaked
PASS: t/18-custom-capture
END: /usr/lib/libcapture-tiny-perl/ptest
2018-10-18T23:20
<LAVA_SIGNAL_ENDTC libcapture-tiny-perl>
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=libcapture-tiny-perl RESULT=pass>
Received signal: <ENDTC> libcapture-tiny-perl
<LAVA_SIGNAL_TESTCASE TEST_CASE_ID=libcapture-tiny-perl RESULT=pass>
Received signal: <TESTCASE> TEST_CASE_ID=libcapture-tiny-perl RESULT=pass
case: libcapture-tiny-perl
case_id: 47270
definition: 1_ptest-tests-with-lava-signals
result: pass
[…]
case: libcapture-tiny-perl
case_id: 47270
definition: 1_ptest-tests-with-lava-signals
result: pass
[…]
```

ptest Results (before)

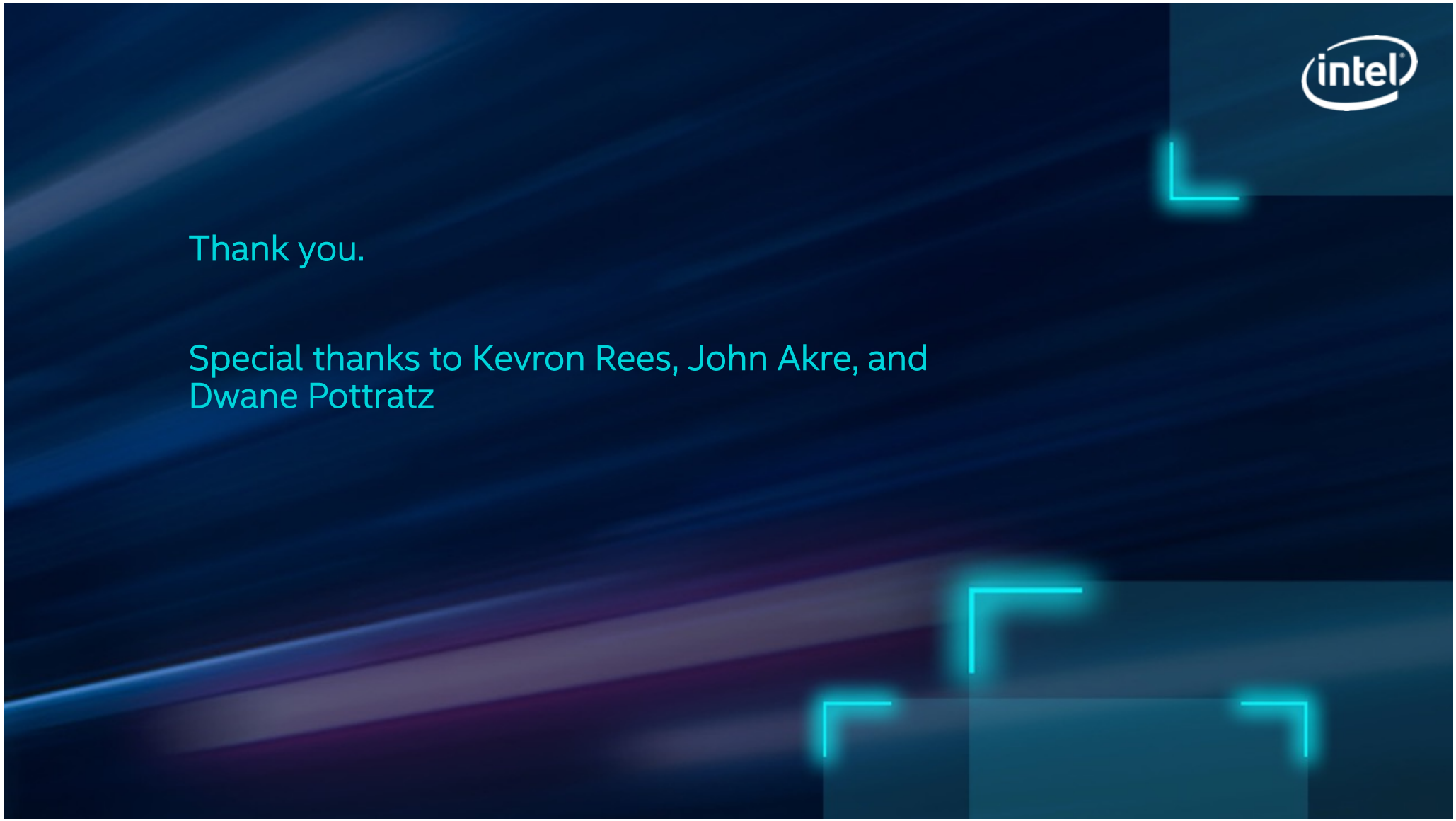


ptest Results (after)



Summary and Future Work

- Simple concept dramatically improves the LAVA dashboard experience and simplifies test definitions
- Much easier to find the cause of test failures
- **pytest**
  - Works well, although output gets a bit cluttered
  - A proper pytest plugin is desirable and in the works
  - The measurement_decorator in the Appendix is not implemented yet in the plugin environment
- **BATS**
  - Works surprisingly well
  - Would be nice to have TAP output (currently –l and –t options are exclusive)
  - Measurements and units functionality is not yet tested
  - Not a commonly used testing framework?
- **ptest**
  - Really each “test case” for ptest-runner is a “test suite”
  - Break out individual actual test case results?
  - Some failures might be hidden if the underlying run-ptest script does not properly expose them (e.g. set –x)
- Enable LAVA signals in even more test runners


intel
Thank you.
Special thanks to Kevron Rees, John Akre, and Dwane Pottratz

Disclaimer

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Appendix: pytest

With Measurements and Units

conftest.py (with Measurements)

- At the root of the directory where you call ‘pytest’

```
# conftest.py
#
# Copyright (C) 2018 Intel Corporation
# SPDX-License-Identifier: MIT

import measurement_decorator

def lava_result_convert(pytest_outcome):
    """ Convert the pytest outcome to the string expected by LAVA.
    """
    if pytest_outcome is 'passed':
        return 'pass'
    elif pytest_outcome is 'skipped':
        return 'pass'
    elif pytest_outcome is 'xfailed':
        return 'pass'
    else:
        return 'fail'
```

conftest.py (cont'd)

- Allows you to override pytest defaults

```
def pytest_report_teststatus(report):
    """ Insert strings that LAVA expects to capture test results.
    """
    # Get pytest test name and remove the 'test_' prefix
    test_name = report.location[2][5:]

    if report.when is 'setup':
        print('\n')
        print('<LAVA_SIGNAL_STARTTC ' + test_name + '>')
    elif report.when is 'call':
        test_result = lava_result_convert(report.outcome)

        print('\n')
        print('<LAVA_SIGNAL_ENDTC ' + test_name + '>')

        measurement_text = ""
```

conftest.py (cont'd)

```
for measured_test in measurement_decorator.Measurement.tests.keys():
        if measured_test.__name__ == report.location[2]:
            val = measurement_decorator.Measurement.tests[measured_test]
            meas = val.measurement
            units = val.units
            measurement_text = "MEASUREMENT={} UNITS={}".format(meas, units)

    print('\n')
    print('<LAVA_SIGNAL_ENDTC ' + test_name + '>')
    print('<LAVA_SIGNAL_TESTCASE TEST_CASE_ID={} RESULT={} {}>'.format(
           test_name, test_result, measurement_text))
```

measurement_decorator.py

- At the root of the directory where you call ‘pytest’

```
# measurement_decorator.py
#
# Copyright (C) 2018 Intel Corporation
# SPDX-License-Identifier: MIT


class Measurement(object):

    tests = {}

    def __init__(self, f):
        self.f = f
        self.measurement = None
        self.units = None

        Measurement.tests[f] = self
```

measurement_decorator.py (cont'd)

```
def export_measurement(the_test):

    measurement = Measurement(the_test)

    def wrapper:
        measurement.measurement, measurement.units = the_test()

    return wrapper
```

test_measurement.py

```
# test_measurement.py
#
# Copyright (C) 2018 Intel Corporation
# SPDX-License-Identifier: MIT

""" Example usage of measurement_decorator"""
@measurement_decorator.export_measurement
def test_boiling():

    boiling_point_of_water = 100

    return boiling_point_of_water, "°C"
```

pytest with Measurements Thoughts

- The test cases themselves have to be modified (adding the measurement_decorator and returning <measurement>,<units>).
- This somewhat breaks the original intent (keep the tests unmodified for LAVA).


intel