**How to add test cases on JTA**

Copyright (C) 2015 FNST LIMITED

Copyright (C) 2015 FUJITSU TEN LIMITED

**Abstract**

This document is used to demonstrate how to add a regular test case to JTA. The newly added test case, as an example for this document, is used to test “touch” command. That is to say, “touch <file>” will be executed on the target machine. If “<file>” is created successfully, the test passes; otherwise it fails.

## Important note

This document is a bit obsolete and needs to be updated.

Important details:

* JTA was renamed to Fuego
* Jenkins is configured (by default) to start web interface at <http://IPADDR:8080/fuego>

Keeping given changes in mind this document is still useful.

1. To make explanation easier, we make some assumptions here:
   1. The machine with JTA installed on it will be called “JTA machine” below. The IP address of JTA machine is 192.168.30.71.
   2. The machine, on which the test is supposed to be tested, will be called “target machine” below. The IP address of target machine is 192.168.30.64.
2. Login to JTA machine as “root” user.
3. Use the following command to check whether “Jenkins” service is working.

|  |
| --- |
| # /etc/init.d/jenkins status |

If message, like “Jenkins Continuous Integration Server is not running”, is showed, please use the following command to start “Jenkins” service.

|  |
| --- |
| # /etc/init.d/jenkins start |

1. The following table lists the files that should be added or fixed in order to add a test case for “touch” command.

|  |  |
| --- | --- |
| file | usage |
| (optional)  /home/jenkins/overlays/testplans | used for selecting “spec” for test cases, so that some variables in test\_specs will be set to satisfy the requirement of the test. |
| (optional)  /home/jenkins/overlays/test\_specs | used for defining some variables for test. These variables are organized as “spec”. In different “spec”, variables will be defined differently |
| /home/jenkins/tests/Functional.touch/touch-script.sh | test start point that will be used to setup the test environment, execute the test and grab test result from target machine |
| /home/jenkins/tests/Functional.touch/touch-device.sh | test program that will be executed on the target machine to test “touch” command |
| /home/jenkins/overlays/boards/porter.board | configuration of target machine, touch-script.sh needs this to setup test environment |
| /home/jenkins/scripts/tools.sh | defining variables used to cross-build programs for target machine |

“/home/jenkins/overlays/testplans” and “/home/jenkins/overlays/test\_specs” are optional, only used when some special variables are needed for certain tests.

More detailed information will be demonstrated in the next several steps.

1. Add “test plan” (optional)

Add “testplan\_touch.json” under “/home/jenkins/overlays/testplans”, and write it as the following example.

|  |
| --- |
| # cd /home/jenkins/overlays/testplans  # cat testplan\_touch.json  {  "testPlanName": "testplan\_touch",    name of test plan  name of test  name of test spec  "tests": [  {  "testName": "Functional.touch",  "spec": "touch-exp1"  }  ]  } |

1. Add “test spec” (optional)

Add “Functional.touch.spec” under “/home/jenkins/overlays/test\_specs”, and write it as the following example.

|  |
| --- |
| # cd /home/jenkins/overlays/test\_specs  # cat Functional.touch.spec  {  name of test  name of test spec  variables for the spec  "testName": "Functional.touch",  "specs":  [  {  "name":"touch-exp1",  "FILENAME":"touch.file"  }  ]  } |

1. Relationship between “test plan” and “test spec”

|  |  |  |
| --- | --- | --- |
| test plan (testplan\_touch.json) |  | test spec (Functional.touch.spec) |
| #cat testplan\_touch.json  {  "testPlanName": "testplan\_touch",  "tests": [  {  "testName": "Functional.touch",  "spec": "touch-exp1"  }  ]  } |  | #cat Functional.touch.spec  {  "testName": "Functional.touch",  "specs":  [  {  "name":"touch-exp1",  "FILENAME":"touch.file"  }  ]  } |

1. Add test script

Create folder “Functional.touch” under “/home/jenkins/tests”, and under the folder add two files, “touch-script.sh” and “touch-device.sh”.

Follow the example below to write “touch-script.sh”.

|  |
| --- |
| # cd /home/jenkins/tests  # mkdir Functional.touch  test name  test start point  function used to build test program  function used to deploy test program to the target machine  function used to execute test program on the target machine  confirm variables are defined  function used to handle the log of executing test program to decide the result of the test  script that will call above functions to do the test  # cat touch-script.sh  #!/bin/bash  function test\_build {  echo "test compiling (should be here)"  }  function test\_deploy {  put $TEST\_HOME/touch-device.sh $JTA\_HOME/jta.$TESTDIR/  }  function test\_run {  assert\_define FUNCTIONAL\_TOUCH\_FILENAME  report "cd $JTA\_HOME/jta.$TESTDIR; ./touch-device.sh $FUNCTIONAL\_TOUCH\_FILENAME"  }  function test\_processing {  log\_compare "$TESTDIR" "1" "PASS$" "p"  log\_compare "$TESTDIR" "0" "FAIL$" "n"  }  . $JTA\_ENGINE\_PATH/scripts/functional.sh |

Follow the example below to write “touch-device.sh”. Be careful, “touch-device.sh” should gain the executable permission in order to be run on target machine.

|  |
| --- |
| # cat touch-device.sh  test program  file that will be “touch”ed  clean environment to avoid the file already exists  “touch” the file  if the file exists, output “PASS”, otherwise, output “FAIL”  clean enviroment  #!/bin/bash  echo "Touch Founction Test!"  file=/tmp/$1  rm –f $file  touch $file  if [ -f $file ];then  echo "PASS"  else  echo "FAIL"  fi  rm –f $file |

1. Fix configuration of target machine

Follow the example below to fix porter’s related configuration, “porter.board” under “/home/jenkins/overlays/boards”.

|  |
| --- |
| # cd /home/jenkins/overlays/boards  # cat qemu-arm.board  inherit "base-board"  include "base-params"  IP address or hostname of target machine  user name for ssh login  password for ssh login, not needed for AGL yet  IPADDR="192.168.30.64"  LOGIN="root"  JTA\_HOME="/home/a"  #PASSWORD="root"  PLATFORM="porter"  TRANSPORT="ssh"  ARCHITECTURE="arm"  #SATA\_DEV="/dev/sda1"  #SATA\_MP="/mnt/sata"  #USB\_DEV="/dev/sdb1"  #USB\_MP="/mnt/usb"  #MMC\_DEV="/dev/mmcblk0p2"  #MMC\_MP="/mnt/mmc" |

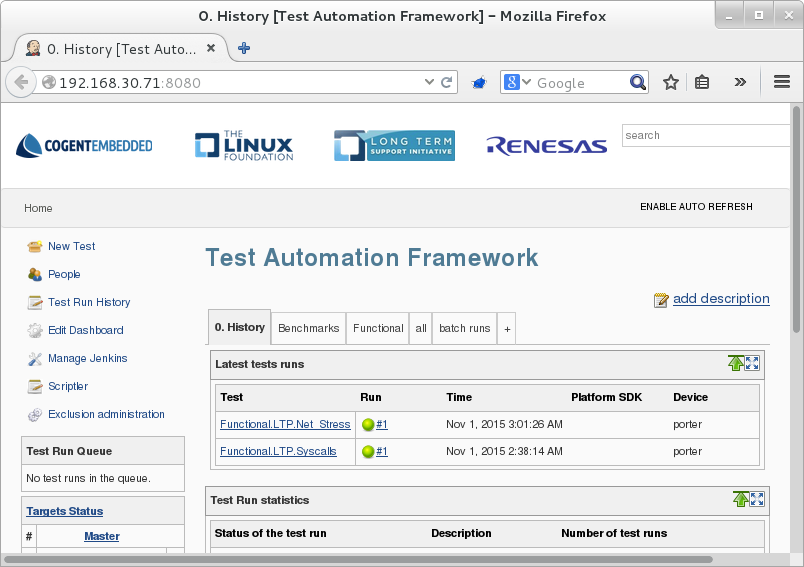
If you want to execute the test on other target machine, fix the related “\*.board” file. You can also refer to “jta-guide.pdf” for more detailed information.

1. Fix variable definition used for corss-building

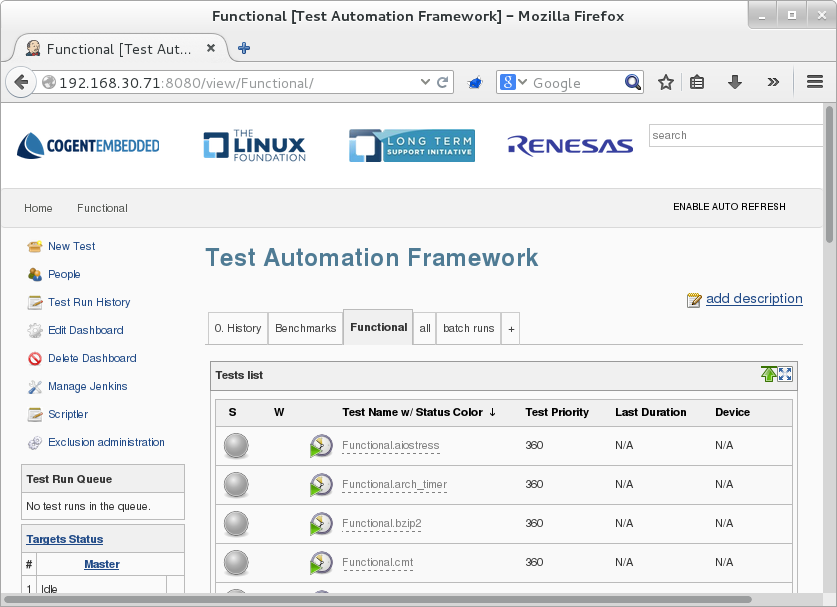
Fix “tools.sh” under “/home/jenkins/scripts”. Variables, like SDKROOT, PREFIX, HOST, and “source” are used to setup cross-build environment.

|  |
| --- |
| # cd /home/jenkins/scripts  # cat tools.sh  ……  selected by “PLATFORM” variable in “\*.board”. Check step 9  elif [ "${PLATFORM}" = "porter" ];  then  ORIG\_PATH=$PATH  PREFIX=arm-poky-linux-gnueabi  source /opt/poky-agl/1.0.0/environment-setup-cortexa15hf-vfp-neon-poky-linux-gnueabi  SDKROOT=/opt/poky-agl/1.0.0/sysroots/cortexa15hf-vfp-neon-poky-linux-gnueabi/  HOST=arm-poky-linux-gnueabi  unset PYTHONHOME  env -u PYTHONHOME  ...... |

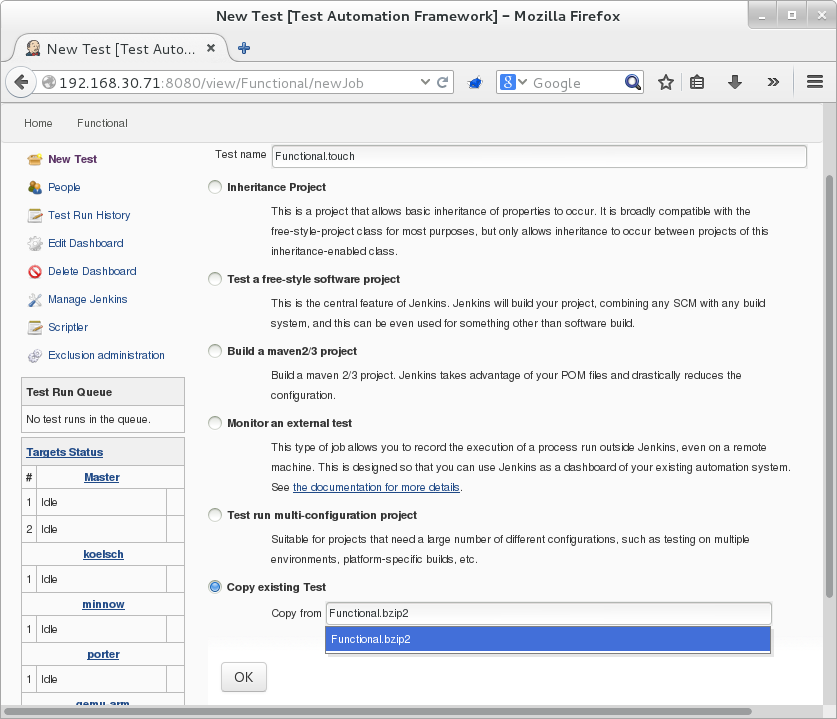
1. Logon to JTA web interface. The URL should be “192.168.30.71:8080” here:



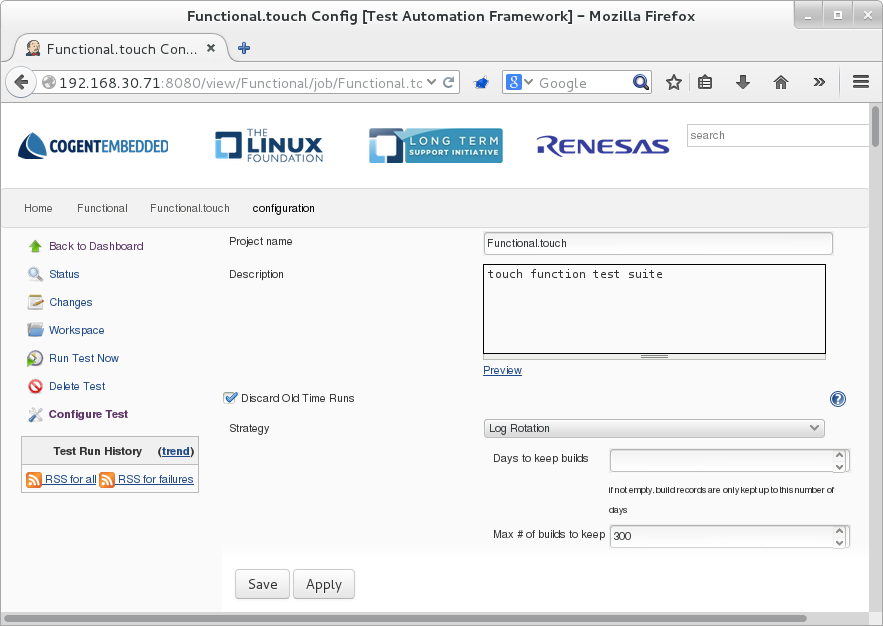
1. Click “Functional” tag, then click “New Test” to create a new test case



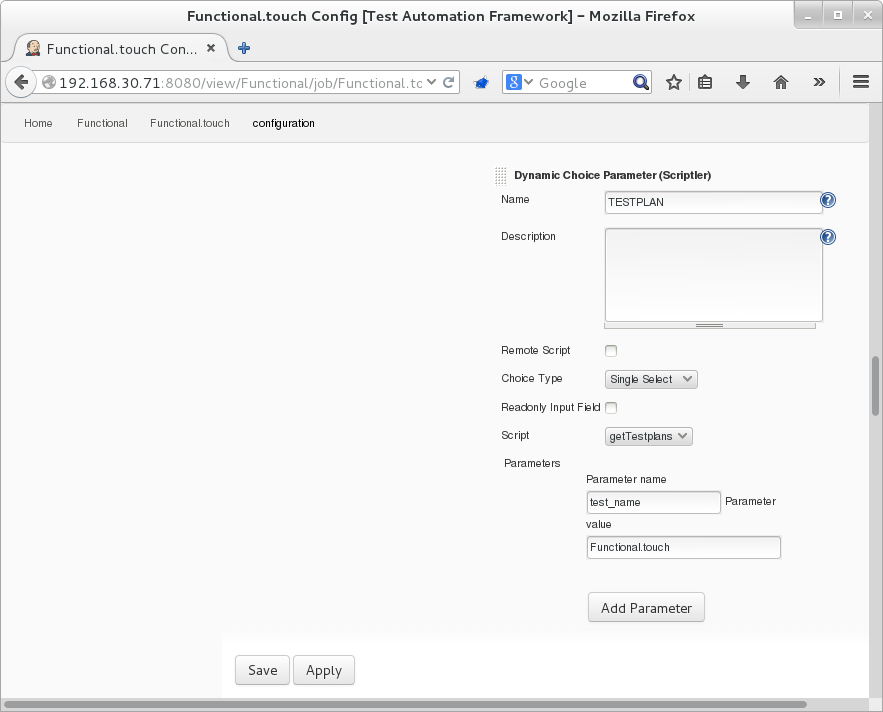
1. Input “Functional.touch” for “Test name”. Then check “Copy existing Test”, input “Functional.bzip2”. After all, click “OK”.



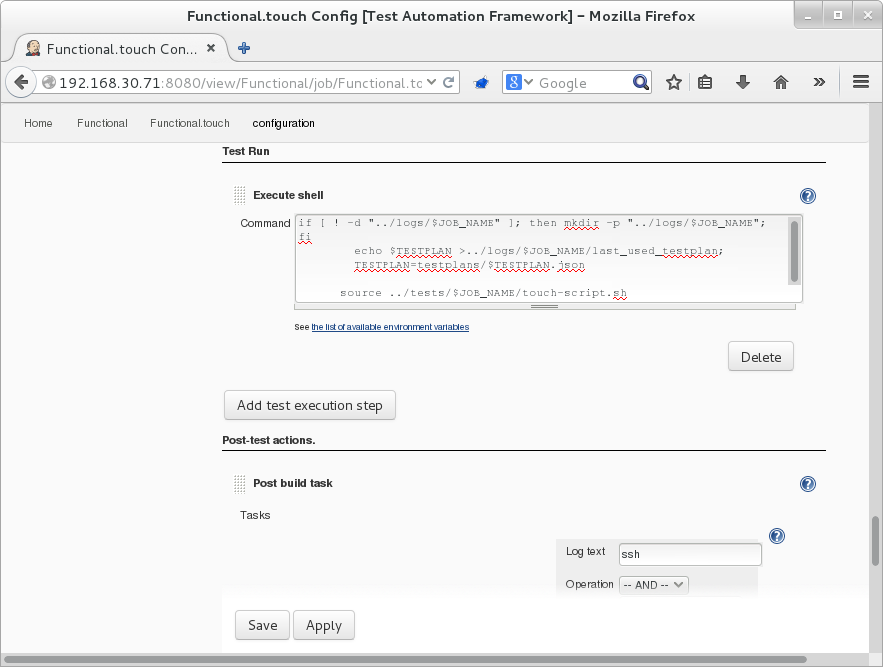
1. Fix configurations related to the test
2. test description:



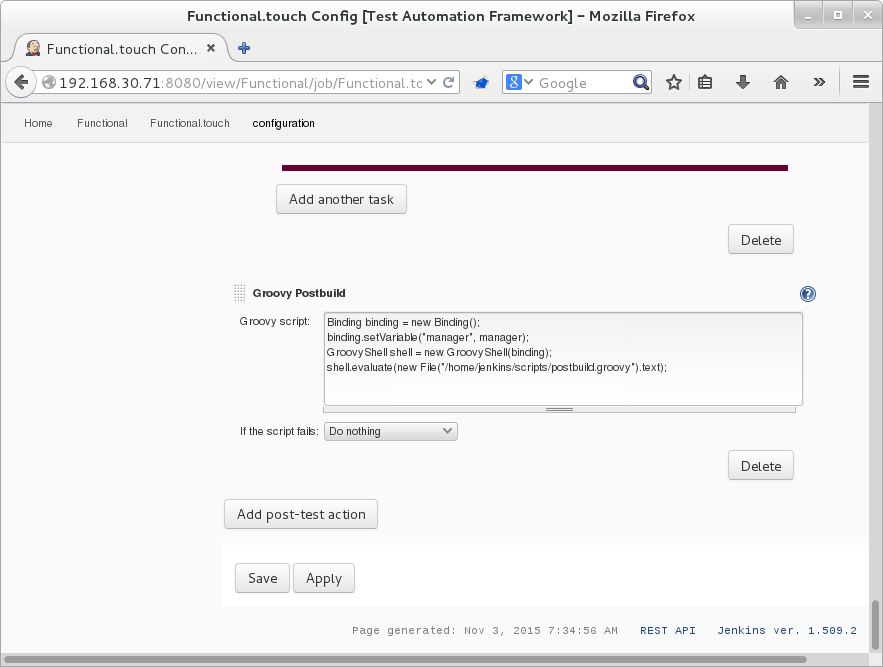
1. test name:



1. test start point, it should be “touch-script.sh” here:



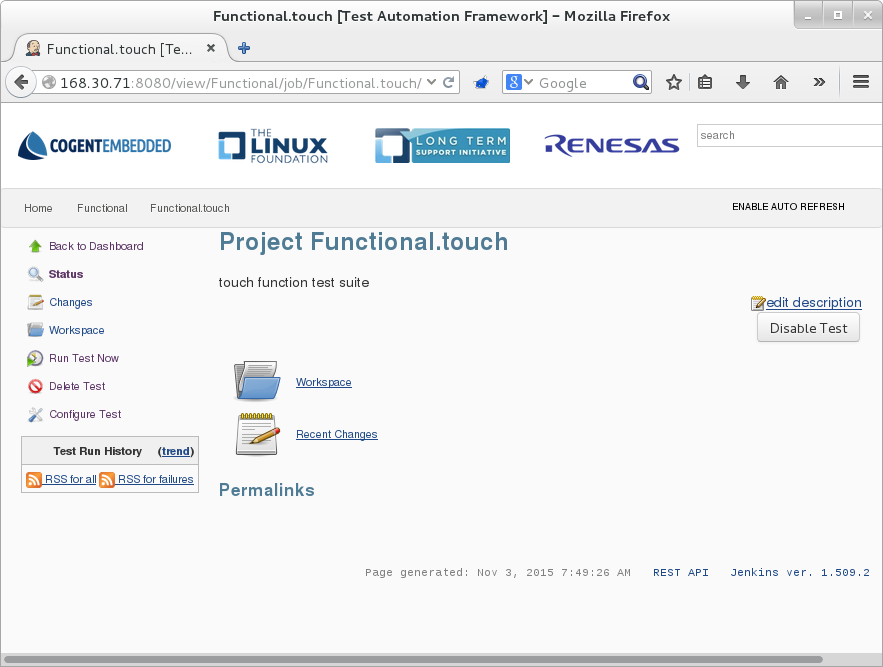
1. click “Apply”, then the new test case is created:

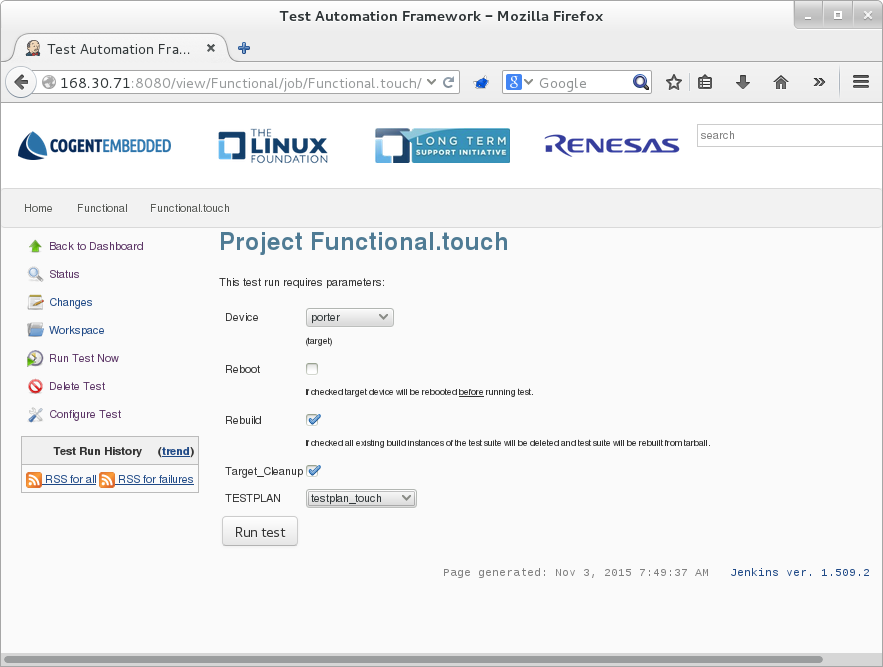


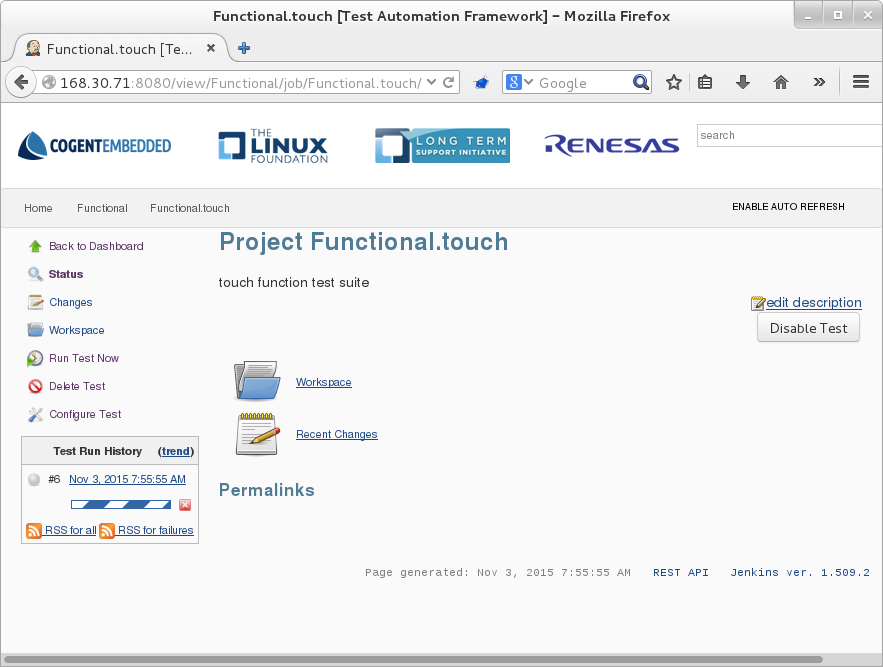
1. Clike “Run Test Now” on the left side.

Choose “porter” for “Device”, check “Rebuild” and choose “testpaln\_touch” for TESTPLAN.

Then click “Run test” to start the test. The test progress will be showed in “Test Run History”.

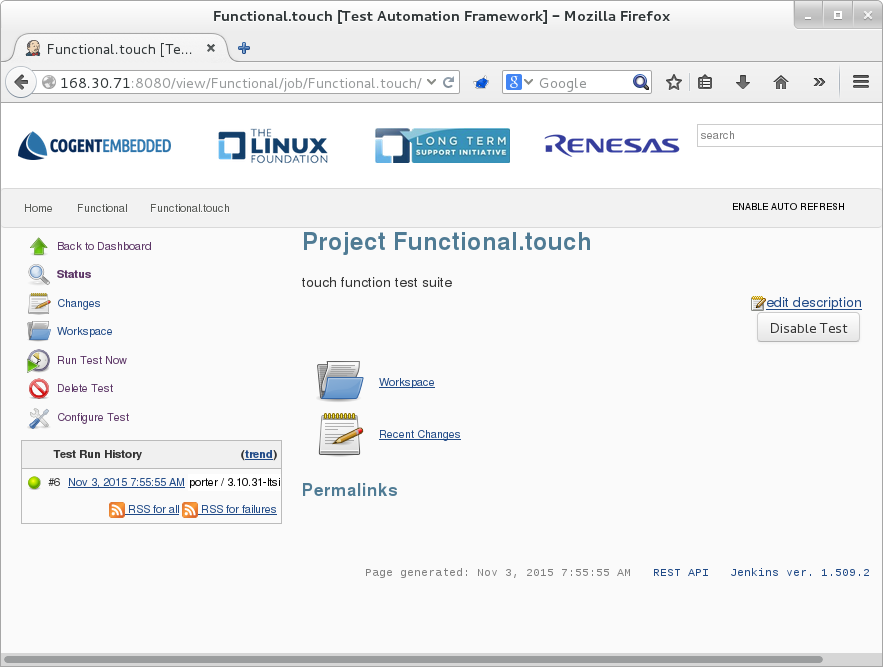






1. If the test succeeded, a line with a green icon in front of it will be showed; otherwise, a red icon will be showed.

Click this line to get more information about this test.



1. Click “Console Output” on the left side, log of the test will be showed.

