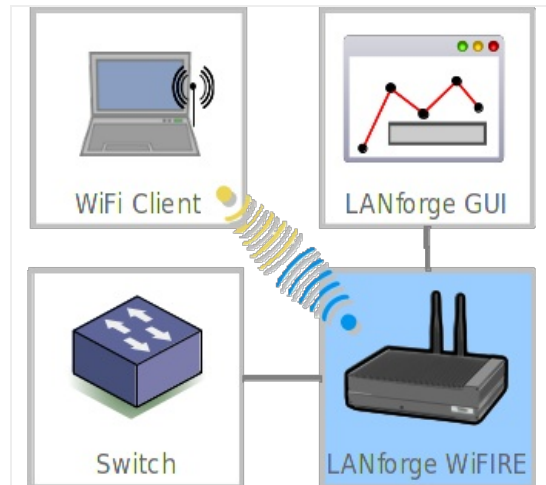


CI/CD Automated Testing of Access Points

Goal: Setup a framework to automatically find new builds, install them on target AP, run regression tests, and report values. Generate graphs of key data over all runs for historical view. Candela support is available to help customize scripts and fully implement this for your particular environment.

In this test scenario, a LANforge CI/CD package (ct523c, RF chamber, Test Controller and other bits and pieces) consumes AP firmware build images, runs automated tests, and reports the values. A test-orchestrator machine queries the build system to find new builds. Work-items are created when new images are found. A test-bed controller looks for images that match the hardware in the test bed. The test-bed controller updates the AP with new firmware, reboots and configures it. The test-bed controller then automates control of the LANforge system to run a series of tests and then gathers the results. When the tests are finished, they are uploaded back to the test-orchestrator. The test-orchestrator will regenerate historical graphs when new results are found, and upload the finished reports to a web server.



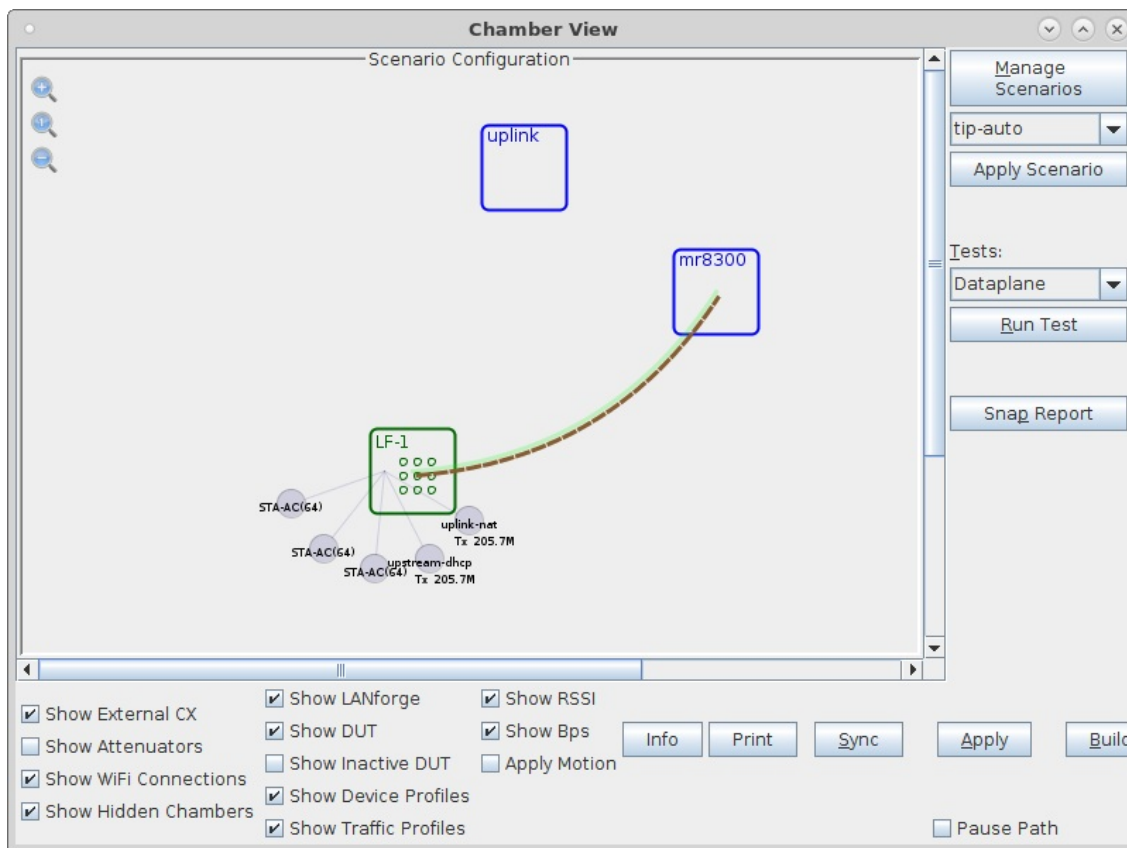
1. Configure Systems for automated testing. In this example, there are three main systems: Test-Orchestrator (TO), Test-Bed Controller (TB), and LANforge system (LF). The AP under test is referred to as DUT. Many of the details are left out of this document, your favorite search engine or network administrator should be able to help show you the details.
 - A. The automation uses ssh and scp to do its work. So, first you have to make sure that the TB can log into the TO with ssh without requiring password. This example uses the user 'lanforge' on the TO and LF machines.
 - B. The testbed machine user must be able to run'sudo' without needing a password.
 - C. In order to read console logs, the TO should act as the serial console server for the LANforge and AP DUT. In order to have the lanforge user on the TB system access the serial ports, you must change the permissions:

```
sudo chmod a+rw /dev/ttyUSB*
```

This command must be run after the serial consoles are connected, or at least re-run once all of the /dev/ttyUSB* devices are found.
 - D. Install additional software packages needed by the automation scripts.

```
sudo pip3 install pexpect-serial
```
2. Configure LANforge to be used as an automated testbed.

A. Clean up any old config and create a chamber view scenario and DUT.



B. In this example, the AP DUT is called 'mr8300'.

The screenshot shows the 'Create/Modify DUT' dialog box. The 'Name' field is set to 'mr8300'. The 'Image file' is 'NONE'. The 'SW Info' field is '046ab4f' and the 'HW Info' field is 'Linksys-MR8300'. The 'Model Number' is 'Linksys-MR8300'. The 'Serial port' is empty. The 'LAN' field is empty. The 'SSID-1' is 'OpenWrt-2', 'SSID-2' is 'OpenWrt-5lo', and 'SSID-3' is 'OpenWrt-5hi'. The 'Mgt IP' is '0.0.0.0'. The 'Num Ant Radio 1' and 'Num Ant Radio 2' are both '0'. The 'BSSID-1' is '32:23:03:81:9c:29', 'BSSID-2' is '30:23:03:81:9c:27', and 'BSSID-3' is '30:23:03:81:9c:28'. The 'Active' checkbox is checked. The 'AP DUT' checkbox is checked. The 'WPA' checkbox is checked. The 'Provides DHCP on LAN' checkbox is checked. The 'Notes' field is empty. The 'Apply', 'OK', and 'Cancel' buttons are at the bottom.

Name	mr8300		
Image file	NONE	Choose Image	x
SW Info	046ab4f	HW Info	Linksys-MR8300
Model Number	Linksys-MR8300	Serial Number	
Serial port		WAN	
LAN		API version	0
SSID-1	OpenWrt-2	Password-1	12345678
SSID-2	OpenWrt-5lo	Password-2	12345678
SSID-3	OpenWrt-5hi	Password-3	12345678
Mgt IP	0.0.0.0	Num Ant Radio 1	0
Num Ant Radio 2	0	Num Ant Radio 3	0
BSSID-1	32:23:03:81:9c:29	BSSID-2	30:23:03:81:9c:27
BSSID-3	30:23:03:81:9c:28	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> AP DUT
<input type="checkbox"/> STA DUT	<input type="checkbox"/> WEP	<input type="checkbox"/> WPA	<input type="checkbox"/> WPA2
<input type="checkbox"/> WPA3	<input type="checkbox"/> 802.11r	<input type="checkbox"/> 802.1x EAP-TTLS	<input type="checkbox"/> Provides DHCP on LAN
<input type="checkbox"/> Provides DHCP on WAN			
Notes			

- C. This configuration uses some advanced virtual-router features in LANforge to have LANforge serve the DUT DHCP and provide a private NAT'ed network for the DUT the stations connected to it. The uplink DUT is part of how this is configured. The LAN address is the gateway for the virtual router. In this case, the LANforge machine's DUT uplink port is eth1, and it is cabled to a network that uses 192.168.3.5 as the gateway/router. Other test beds may use different Ethernet ports to provide these features.

Create/Modify DUT

Name: uplink

Image file: NONE [Choose Image] [x]

SW Info: [] HW Info: []

Model Number: [] Serial Number: []

Serial port: [] WAN: []

LAN: 192.168.3.5/24 API version: 0

SSID-1: [] Password-1: []

SSID-2: [] Password-2: []

SSID-3: [] Password-3: []

Mgt IP: 0.0.0.0 Num Ant Radio 1: 0

Num Ant Radio 2: 0 Num Ant Radio 3: 0

BSSID-1: 00:00:00:00:00:00 BSSID-2: 00:00:00:00:00:00

BSSID-3: 00:00:00:00:00:00 Active AP DUT

STA DUT WEP WPA WPA2

WPA3 802.11r 802.1x EAP-TTLS Provides DHCP on LAN

Provides DHCP on WAN

Notes: []

[Apply] [OK] [Cancel]

- D. The Chamber View scenario creates stations on the available radios, provides the Upstream port (from perspective of the DUT), as well as uplink + NAT for the interface that routes DUT traffic upstream of LANforge.

Create/Modify Scenario

Scenario Name: tip-auto [Delete Scenario] [Create Profile] [Create Traffic Profile] [Add Row]

Del	Resource Profile	Amount	Uses-1	Uses-2	Frequency	Maps To	Tra
X	1.1 STA: STA-AC	64 (64)	wiphy0	AUTO	AUTO (-1 Mhz)	DUT: mr8300 Radio-2	NA
X	1.1 STA: STA-AC	100 (100)	wiphy1	AUTO	AUTO (-1 Mhz)	DUT: mr8300 Radio-1	NA
X	1.1 STA: STA-AC	64 (64)	wiphy2	AUTO	AUTO (-1 Mhz)	DUT: mr8300 Radio-3	NA
X	1.1 Upstream: upstream-dhcp	1 (1)	eth0	AUTO	AUTO (-1 Mhz)	DUT: mr8300 LAN	NA
X	1.1 Uplink: uplink-nat	1 (1)	eth1	eth0	AUTO (-1 Mhz)	DUT: uplink LAN 192.168.3.5/24	NA

[Build New] [Load Scenario] [Update and Save Scenario] [Apply and Save Scenario] [Cancel]

- E. Once chamber-view is set up, you need to create some test configurations for the various automated tests you wish to run. In this example, one test we use is the Dataplane test to test throughput at different packet sizes.

The screenshot shows the 'Dataplane Test' configuration window with the following settings:

- Settings:** Selected DUT: mr8300, Duration: 15 sec (15 s)
- Advanced Configuration:** Downstream/WiFi Port: 1.1.6 sta00000, Upstream Port: 1.1.1 eth0, Rate: 85%, Opposite Rate: 0Kbps, Path Loss: 10
- Channels:** Mode: Auto, Packet Size: 60, 142, 256, 512, 1024, MTU, 4000, 9000
- Spatial Streams:** AUTO, 1, 2, 3, 4
- Security:** AUTO, Open, WEP, WPA, WPA2, WPA3
- Bandwidth:** AUTO, 20, 40, 80, 160
- Traffic Type:** UDP, TCP, Arm-UDP
- Direction:** DUT Transmit, DUT Receive
- Attenuator 1:** NONE (0), checkboxes 1-8, 0..+50..950
- Attenuator 2:** NONE (0), checkboxes 1-8, 0..+50..950
- Turntable:** NONE (0), 0..+45..359

Buttons at the bottom: Start, Another Iteration, Pause, Cancel

- F. On the Dataplane Advanced tab, select a name and save the config. We will use that name later when configuring the automated tests.

The screenshot shows the 'Dataplane Test' configuration window with the 'Advanced Configuration' tab selected. The settings are:

- Save:** dpt-mtu
- Load:** DEFAULT
- Delete:** DEFAULT
- IP ToS:** Best Effort (0)
- Loop Iterations:** Single (1)
- Pause After Each Iteration
- Multi-Conn:** One (1)
- Multi-Pkt:** 1000

Buttons at the bottom: Start, Another Iteration, Pause, Cancel

G. Configured the AP-Auto test as well.

AP Automated Test

Capacity Configuration | Pass/Fail Configuration | Report Configuration

Settings | Advanced Configuration | Stability Configuration

Open DUT: mr8300 OpenWrt-2 | PSK DUT: NA | Enterprise DUT: NA

Selected DUT 2G: mr8300 OpenWrt-2 | Selected DUT 5G: mr8300 OpenWrt-5hi | Upstream Port: 1.1.1 eth0

2.4Ghz Radios: 1.1.4 wiphy1 | 5Ghz Radios: 1.1.3 wiphy0

1.1.5 wiphy2

Tests to run: Basic Client Connectivity Throughput vs Pkt Size Dual Band Performance Capacity Stability Long-Term

Estimated Test Duration: 29.333 m

Another Iteration Pause

H. Configure the AP-Auto Stability configuration.

AP Automated Test

Capacity Configuration | Pass/Fail Configuration | Report Configuration

Settings | Advanced Configuration | Stability Configuration

Stability Duration: 3600 (1 hr) Reset Radios

Concurrent Ports to Reset: Single (1)

Minimum Time between Resets: 10 seconds (10 s) | Maximum Time between Resets: 1 minute (1 m)

VOIP Call Count: Twenty (20)

Video Emulation Rate: SD 360p (700 Kbps) | Video Buffer Size: 1m (976.5625 KB)

Stability UDP Min Download Rate: 500000 (500 Kbps) | Stability UDP Max Download Rate: Same

Stability UDP Min Upload Rate: 500000 (500 Kbps) | Stability UDP Max Upload Rate: Same

Stability TCP Min Download Rate: 500000 (500 Kbps) | Stability TCP Max Download Rate: Same

Stability TCP Min Upload Rate: 500000 (500 Kbps) | Stability TCP Max Upload Rate: Same

Stability stall threshold UDP Download: 100000 (100 Kbps) | Stability stall threshold UDP Download: 100000 (100 Kbps)

Stability stall threshold TCP Download: 100000 (100 Kbps) | Stability stall threshold TCP Download: 100000 (100 Kbps)

Stability stall threshold Video: 100000 (100 Kbps) | Stability stall threshold VOIP: 20000 (20 Kbps)

Another Iteration Pause

- I. Configure the AP-Auto Advanced configuration, and save the configuration when done.

AP Automated Test

Capacity Configuration | Pass/Fail Configuration | Report Configuration

Settings | **Advanced Configuration** | Stability Configuration

Save: DEFAULT

Load: DEFAULT

Delete: DEFAULT

IP ToS: Best Effort (0) | Multi-Conn: One (1)

Skip 2.4Ghz Tests | Skip 5Ghz Tests | Skip Dual-Band Tests

Loop Iterations: Single (1)

2.4Ghz Station Count: 100 (100) | 5Ghz Station Count: Large (128)

Dual-Band Station Count: 228 (228)

Duration: Default (20 sec)

Long-Term Download Rate: 85% | Long-Term Upload Rate: 85%

Long-Term Duration: 3600 (1 hr) | Long-Term Graph Interval: 30 (30 sec)

Long-Term Station Count: Two (Default) (2)

Hunt Retries: Default (1)

Frame Sizes: 200, 512, 1024, MTU

Capacity Amounts (stations): 1, 2, 5, 10, 20, 40, 64, 128, 256, 512, 1024, MAX

Packet Loss Threshold: 1% (1%)

Start | Another Iteration | Pause | Cancel

- J. Configure a capacity-test, and save the configuration.

WiFi Capacity Test

PDU Mix Settings | **Advanced Settings** | Pass/Fail Settings | Select Output | Notes

Settings | Select Ports | Test Groups

Station Increment: 1, 2, 5, 10, 20, 45, 60, 100 | Loop Iterations: Single (1)

Duration: 1 min (1 m) | Use Test Groups | Subset of Test Group

Protocol: TCP-IPv4 | Layer 4-7 Endpoint: NONE

Payload Size: AUTO | MSS: AUTO

Total Download Rate: 1G (1 Gbps)

Total Upload Rate: 1G (1 Gbps)

Percentage TCP Rate: 10% (10%)

Station-Down Quiesce period: 0 (0 sec)

Save: DEFAULT

Load: DEFAULT

Delete: DEFAULT

Start | Pause | Cancel

3. Configure Testbed Controller. This system will use the lanforge-scripts project (installed at /home/lanforge/scripts on LANforge systems). You will probably want to have a separate code repository for the Testbed automation. Candela can help you build scripts specific to your environment. This example shows how it was implemented for one project.

- A. Create a directory structure that looks like this. The 'cicd' directory will contain glue logic to talk to the Test Orchestrator and to the LANforge automation. The testbeds directory holds information for each testbed in your configuration. The lanforge directory has a sub-directory called 'lanforge-scripts' that points to the lanforge-scripts project.

```

greearb@ben-home:~/git/tip/wlan-testing
File Edit View Search Terminal Help
[greearb@ben-home wlan-testing]$
[greearb@ben-home wlan-testing]$
[greearb@ben-home wlan-testing]$
[greearb@ben-home wlan-testing]$
[greearb@ben-home wlan-testing]$
[greearb@ben-home wlan-testing]$ ls
cicd lanforge NOTES.txt opensync README.md sync_repos.bash testbeds
[greearb@ben-home wlan-testing]$

```

- B. Inside the testbeds directory will be a sub-directory for each testbed in the project. In this example, we have two testbeds. We will focus on the one called 'ben-home'

```

greearb@ben-home:~/git/tip/wlan-testing/testbeds
File Edit View Search Terminal Help
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$
[greearb@ben-home testbeds]$ ls
ben-home ferndale-basic-01
[greearb@ben-home testbeds]$

```

- C. The ben-home testbed directory contains the Chamber View scenario, AP Auto, Dataplane and capacity configuration files. The automation will apply the OpenWrt-overlay on top of the base OpenWrt image after upgrading the software on the AP. This is an optional step and may not be useful for other DUTs.

```

greearb@ben-home:~/git/tip/wlan-testing/testbeds/ben-home
File Edit View Search Terminal Help
[greearb@ben-home ben-home]$
[greearb@ben-home ben-home]$
[greearb@ben-home ben-home]$
[greearb@ben-home ben-home]$
[greearb@ben-home ben-home]$
[greearb@ben-home ben-home]$ ls
228_sta_scenario.txt NOTES.txt run_basic_fast.bash tmp_gitlog.html
AP-Auto-ap-auto-228.txt OpenWrt-overlay test_bed_cfg.bash tmp_gitlog.txt
dpt-pkt-sz.txt run_basic.bash testbed_notes.html WCT-228sta.txt
[greearb@ben-home ben-home]$

```

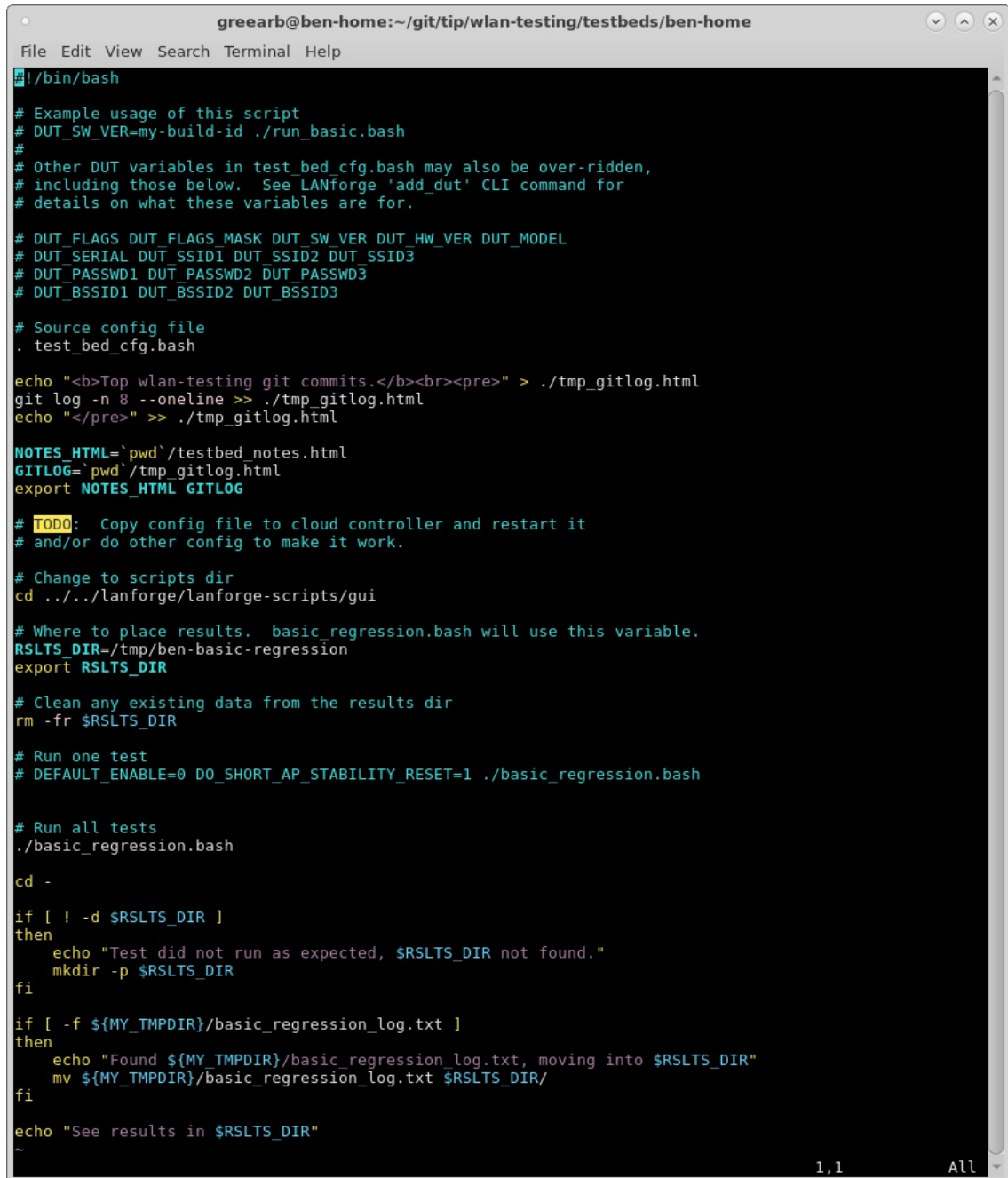
- D. The text configuration files were created by using the lf_testmod.pl script from the lanforge-scripts project. Please see the scripts/gui/README.txt for details.

```

greearb@ben-home:~/btbits/x64_btbits/server/lf_scripts/gui
File Edit View Search Terminal Help
Notes on writing GUI automated tests (such as AP-Auto, TR-398, Dataplane, Capacity test, etc)
AP-Auto:
In the GUI, configure the AP-Auto test as wished, and save the test config on the
Advanced Configuration page. In this example, I use the name: ap-auto-32-64-dual
In LANforge CLI, you can dump the saved configuration:
default@btbits>> show_text blob Plugin-Settings AP-Auto-ap-auto-32-64-dual
TEXT-BLOB:Plugin-Settings.AP-Auto-ap-auto-32-64-dual::[BLANK]
show_events: 1
show_log: 1
port_sorting: 0
notes0: Chamber to Chamber test.
bg: 0xE0ECF8
test_rig: TR-398 test bed
....
Save this text to a file for later use: AP-Auto-ap-auto-32-64-dual.txt
# Save AP-Auto configuration text using the ../lf_testmod.pl script:
../lf_testmod.pl --mgr 192.168.100.156 --action show --test_name AP-Auto-ap-auto-32-64-dual > test_configs/mytest.txt
# Save WiFi-Capacity configuration (saved as 'fb-192' using the ../lf_testmod.pl script:
../lf_testmod.pl --mgr 192.168.100.156 --action show --test_name Wifi-Capacity-fb-192 > test_configs/mytest.txt
# Save Chamber View scenario:
../lf_testmod.pl --mgr 192.168.100.156 --action show --test_name simpleThput --test_type Network-Connectivity > test_configs/my
scenario.txt
--More-- (50%)

```

- E. The `run_basic.bash` script is a helper script that sets some environment variables and then calls the `lanforge-scripts/gui/basic_regression.bash` script.



```
greearb@ben-home:~/git/tip/wlan-testing/testbeds/ben-home
File Edit View Search Terminal Help
~/bin/bash
# Example usage of this script
# DUT_SW_VER=my-build-id ./run_basic.bash
#
# Other DUT variables in test_bed_cfg.bash may also be over-riden,
# including those below. See LANforge 'add_dut' CLI command for
# details on what these variables are for.
#
# DUT_FLAGS DUT_FLAGS_MASK DUT_SW_VER DUT_HW_VER DUT_MODEL
# DUT_SERIAL DUT_SSID1 DUT_SSID2 DUT_SSID3
# DUT_PASSWD1 DUT_PASSWD2 DUT_PASSWD3
# DUT_BSSID1 DUT_BSSID2 DUT_BSSID3
#
# Source config file
. test_bed_cfg.bash
#
echo "<b>Top wlan-testing git commits.</b><br><pre>" > ./tmp_gitlog.html
git log -n 8 --oneline >> ./tmp_gitlog.html
echo "</pre>" >> ./tmp_gitlog.html
#
NOTES_HTML=`pwd`/testbed_notes.html
GITLOG=`pwd`/tmp_gitlog.html
export NOTES_HTML GITLOG
#
# TODO: Copy config file to cloud controller and restart it
# and/or do other config to make it work.
#
# Change to scripts dir
cd ../../lanforge/lanforge-scripts/gui
#
# Where to place results. basic_regression.bash will use this variable.
RSLTS_DIR=/tmp/ben-basic-regression
export RSLTS_DIR
#
# Clean any existing data from the results dir
rm -fr $RSLTS_DIR
#
# Run one test
# DEFAULT_ENABLE=0 DO_SHORT_AP_STABILITY_RESET=1 ./basic_regression.bash
#
# Run all tests
./basic_regression.bash
#
cd -
if [ ! -d $RSLTS_DIR ]
then
echo "Test did not run as expected, $RSLTS_DIR not found."
mkdir -p $RSLTS_DIR
fi
if [ -f ${MY_TMPDIR}/basic_regression_log.txt ]
then
echo "Found ${MY_TMPDIR}/basic_regression_log.txt, moving into $RSLTS_DIR"
mv ${MY_TMPDIR}/basic_regression_log.txt $RSLTS_DIR/
fi
echo "See results in $RSLTS_DIR"
~
1,1 All
```


F. The run_basic_fast.bash script disables most of the tests and just runs a few specific ones.

```
greearb@ben-home:~/git/tip/wlan-testing/testbeds/ben-home
File Edit View Search Terminal Help
#!/bin/bash
# Example usage of this script
# DUT_SW_VER=my-build-id ./run_basic.bash
#
# Other DUT variables in test_bed_cfg.bash may also be over-ridden,
# including those below. See LANforge 'add_dut' CLI command for
# details on what these variables are for.
#
# DUT_FLAGS DUT_FLAGS_MASK DUT_SW_VER DUT_HW_VER DUT_MODEL
# DUT_SERIAL DUT_SSID1 DUT_SSID2 DUT_SSID3
# DUT_PASSWD1 DUT_PASSWD2 DUT_PASSWD3
# DUT_BSSID1 DUT_BSSID2 DUT_BSSID3
#
# Source config file
. test_bed_cfg.bash
#
echo "<b>Top wlan-testing git commits.</b><br><pre>" > ./tmp_gitlog.html
git log -n 8 --oneline >> ./tmp_gitlog.html
echo "</pre>" >> ./tmp_gitlog.html
#
NOTES_HTML=`pwd`/testbed_notes.html
GITLOG=`pwd`/tmp_gitlog.html
export NOTES_HTML GITLOG
#
# TODO: Copy config file to cloud controller and restart it
# and/or do other config to make it work.
#
# Change to scripts dir
cd ../../lanforge/lanforge-scripts/gui
#
# Where to place results. basic_regression.bash will use this variable.
RSLTS_DIR=/tmp/ben-basic-regression-fast
export RSLTS_DIR
#
# Clean any existing data from the results dir
rm -fr $RSLTS_DIR
#
# Run a subset of available tests
# See 'Tests to run' comment in basic_regression.bash for available options.
#
#DEFAULT_ENABLE=0 WCT_DURATION=20s DO_SHORT_AP_BASIC_CX=1 DO_WCT_BI=1 ./basic_regression.bash
#
DEFAULT_ENABLE=0 WCT_DURATION=20s DO_SHORT_AP_BASIC_CX=1 DO_WCT_BI=0 ./basic_regression.bash
#
# Run all tests
#./basic_regression.bash
#
cd -
if [ ! -d $RSLTS_DIR ]
then
echo "Test did not run as expected, $RSLTS_DIR not found."
mkdir -p $RSLTS_DIR
fi
if [ -f ${MY_TMPDIR}/basic_regression_log.txt ]
then
echo "Found ${MY_TMPDIR}/basic_regression_log.txt, moving into $RSLTS_DIR"
mv ${MY_TMPDIR}/basic_regression_log.txt $RSLTS_DIR/
fi
echo "See results in $RSLTS_DIR"
42,1 All
```

- G. Both of the scripts use the test_bed_cfg.bash file to configure the details about the device-under-test and the LANforge system. This configuration will allow the automation to properly do its work.

```
greearb@ben-home:~/git/tip/wlan-testing/testbeds/ben-home
File Edit View Search Terminal Help
Example test-bed configuration
# Scripts should source this file to set the default environment variables
# and then override the variables specific to their test case (and it can be done
# in opposite order for some results
#
# After the env variables are set,
# call the 'lanforge/lanforge-scripts/gui/basic_regression.bash'
# from the directory in which it resides.

PWD=`pwd`
AP_SERIAL=${AP_SERIAL:-/dev/ttyUSB0}
LF_SERIAL=${LF_SERIAL:-/dev/ttyS1}
LFPASSWD=${LFPASSWD:-r} # Root password on LANforge machine
AP_AUTO_CFG_FILE=${AP_AUTO_CFG_FILE:-$PWD/AP-Auto-ap-auto-228.txt}
WCT_CFG_FILE=${WCT_CFG_FILE:-$PWD/WCT-228sta.txt}
DPT_CFG_FILE=${DPT_CFG_FILE:-$PWD/dpt-pkt-sz.txt}
SCENARIO_CFG_FILE=${SCENARIO_CFG_FILE:-$PWD/228_sta_scenario.txt}

# LANforge target machine
LFMANAGER=${LFMANAGER:-192.168.3.190}

# LANforge GUI machine (may often be same as target)
GMANAGER=${GMANAGER:-192.168.3.190}
GMPORT=${GMPORT:-3990}
MY_TMPDIR=${MY_TMPDIR:-/tmp}

# Test configuration (10 minutes by default, in interest of time)
STABILITY_DURATION=${STABILITY_DURATION:-600}
TEST_RIG_ID=${TEST_RIG_ID:-Ben-Home-OTA}

# DUT configuration
#DUT_FLAGS=${DUT_FLAGS:-0x22} # AP, WPA-PSK
DUT_FLAGS=${DUT_FLAGS:-0x2} # AP, Open
DUT_FLAGS_MASK=${DUT_FLAGS_MASK:-0xFFFF}
DUT_SW_VER=${DUT_SW_VER:-OpenWrt-Stock}
DUT_HW_VER=Linksys-MR8300
DUT_MODEL=Linksys-MR8300
DUT_SERIAL=${DUT_SERIAL:-NA}
DUT_SSID1=${DUT_SSID1:-OpenWrt-2}
DUT_SSID2=${DUT_SSID2:-OpenWrt-5lo}
DUT_SSID3=${DUT_SSID3:-OpenWrt-5hi}
DUT_PASSWD1=${DUT_PASSWD1:-12345678}
DUT_PASSWD2=${DUT_PASSWD2:-12345678}
DUT_PASSWD3=${DUT_PASSWD3:-12345678}
DUT_BSSID1=32:23:03:81:9c:29
DUT_BSSID2=30:23:03:81:9c:27
DUT_BSSID3=30:23:03:81:9c:28

export LF_SERIAL AP_SERIAL LFPASSWD
export AP_AUTO_CFG_FILE WCT_CFG_FILE DPT_CFG_FILE SCENARIO_CFG_FILE
export LFMANAGER GMANAGER GMPORT MY_TMPDIR
export STABILITY_DURATION TEST_RIG_ID
export DUT_FLAGS DUT_FLAGS_MASK DUT_SW_VER DUT_HW_VER DUT_MODEL
export DUT_SERIAL DUT_SSID1 DUT_SSID2 DUT_SSID3
export DUT_PASSWD1 DUT_PASSWD2 DUT_PASSWD3
export DUT_BSSID1 DUT_BSSID2 DUT_BSSID3

~
1,1 All
```

- H. The testbed_notes.html is a snippet of HTML that will be added to reports to describe this test bed.

```
greearb@ben-home:~/git/tip/wlan-testing/testbeds/ben-home
File Edit View Search Terminal Help
<!-- Snippet of HTML that will be added to the index.html to describe/document this testbed -->
<P>
This test-bed consists of a CT523 LANforge test system, with 2 wave-1 3x3 radios and one a/b/g/n
ath9k radio. It is connected over-the-air in a home office to the DUT, which is placed about 3
feet away. The DUT is a Linksys MR8300 (which seems to be similar hardware to the EA8300).
Local intereferers include an a/b/g/n AP serving the home. In general, there is not much wifi
traffic.
<P>
~
~
"testbed_notes.html" 8L, 491C
1,1 All
```

- I. The cid directory contains a directory for each of the test-beds, named similar to the testbeds directory. In this case, we are using the ben-home testbed. This directory will hold a file called TESTBED_INFO.txt. This describes the DUT hardware platform so that the test orchestrator knows what testbed can handle which binary images (this project is doing builds for multiple hardware platforms.)



```
greearb@ben-home:~/git/tip/wlan-testing/cid/ben-home
File Edit View Search Terminal Help
TESTBED_HW=ea8300
# Controller's view of the test bed, from wlan-testing/cid/[testbed] directory
TESTBED_DIR=../../testbeds/ben-home
~
~
~
1,13 All
```

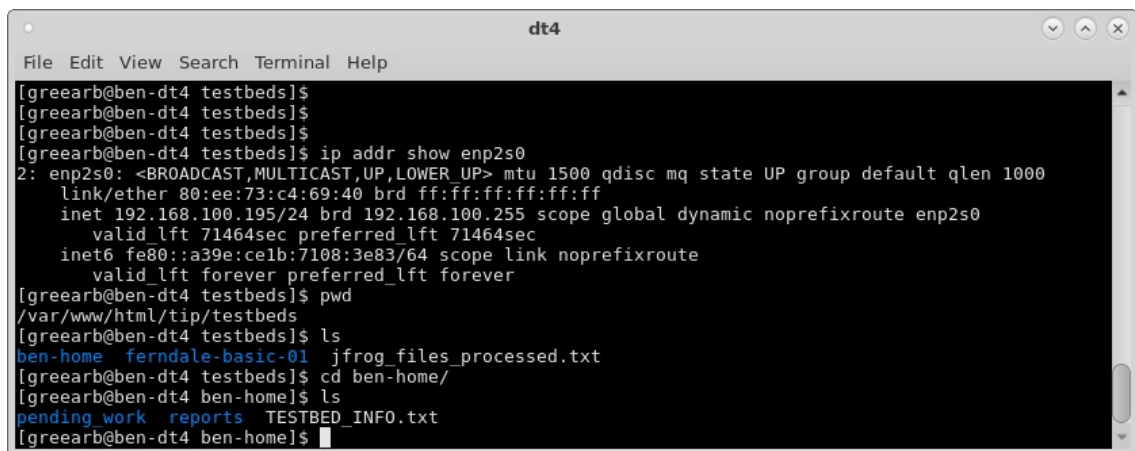
- J. The cid directory also contains a script that queries the test orchestrator. The code in it is beyond the scope of this guide, but roughly speaking, it queries the test orchestrator's web page to find new work items, and when found, it will execute the test and upload results back to the orchestrator. An example run of this tool is:

```
../testbed_poll.pl --jfrog_passwd secret --url http://192.168.100.195/tip/testbeds/ben-home/pending_work/
```

For full automation, this program should run in a loop, waiting 2 minutes between tries to give the orchestrator time to notice the results and remove the old work-item.

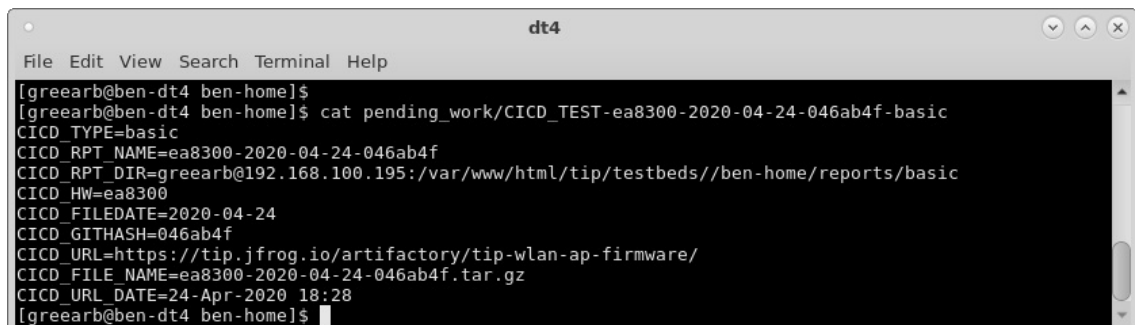
4. Configure test orchestrator. This system is visible to all the test beds. It could be a VM running in a cloud, or some other system running in your lab. It will have a web server, and directories dedicated to the automation.

- A. Create a directory structure that looks like this. A copy of the cid/[testbeds] should be placed here. In particular, the TESTBED_INFO.txt is important.



```
dt4
File Edit View Search Terminal Help
[greearb@ben-dt4 testbeds]$
[greearb@ben-dt4 testbeds]$
[greearb@ben-dt4 testbeds]$
[greearb@ben-dt4 testbeds]$ ip addr show enp2s0
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 80:ee:73:c4:69:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.195/24 brd 192.168.100.255 scope global dynamic noprefixroute enp2s0
        valid_lft 71464sec preferred_lft 71464sec
    inet6 fe80::a39e:ceb:7108:3e83/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[greearb@ben-dt4 testbeds]$ pwd
/var/www/html/tip/testbeds
[greearb@ben-dt4 testbeds]$ ls
ben-home  ferndale-basic-01  jfrog_files_processed.txt
[greearb@ben-dt4 testbeds]$ cd ben-home/
[greearb@ben-dt4 ben-home]$ ls
pending_work  reports  TESTBED_INFO.txt
[greearb@ben-dt4 ben-home]$
```

- B. The orchestrator will poll the build artifact system to find new builds and build work-items. It will place the work-items in the test-beds that it wishes to execute the tests.



```
dt4
File Edit View Search Terminal Help
[greearb@ben-dt4 ben-home]$
[greearb@ben-dt4 ben-home]$ cat pending_work/CICD_TEST-ea8300-2020-04-24-046ab4f-basic
CICD_TYPE=basic
CICD_RPT_NAME=ea8300-2020-04-24-046ab4f
CICD_RPT_DIR=greearb@192.168.100.195:/var/www/html/tip/testbeds//ben-home/reports/basic
CICD_HW=ea8300
CICD_FILEDATE=2020-04-24
CICD_GITHASH=046ab4f
CICD_URL=https://tip.jfrog.io/artifactory/tip-wlan-ap-firmware/
CICD_FILE_NAME=ea8300-2020-04-24-046ab4f.tar.gz
CICD_URL_DATE=24-Apr-2020 18:28
[greearb@ben-dt4 ben-home]$
```

- C. The testbed controller will find this work item, do the work, and then upload the results to the location specified in the work-item. The test-orchestrator will then generate summary reports for all available results, including comparison graphs for the different test runs. It will upload these results to some web page so that users can view the end results.