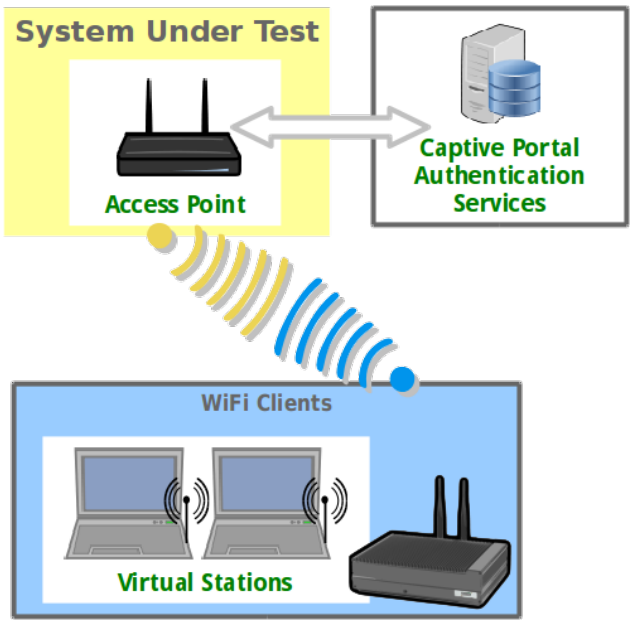
sales@candelatech.com
support@candelatech.com
+1 (360) 380-1618 [PST, GMT -8]

WiFi Captive Portal Bot (portal-bot)

Goal: Execute a battery of of captive portal logins from virtual wifi stations using the newer portal-bot.pl script.

Public access open WiFi service is often gated with a web sign-on form (a captive portal). LANforge virtual stations can emulate sign-in to the captive portal using the `portal-bot.pl` script. This script is by necessity incomplete because many captive portals have different behaviors and login form requirements. With this script, you provide a *bot plugin* that bridges the gap. This cookbook will coach you through a basic portal-bot integration and then you will create ten stations that authenticate through a captive WiFi portal.

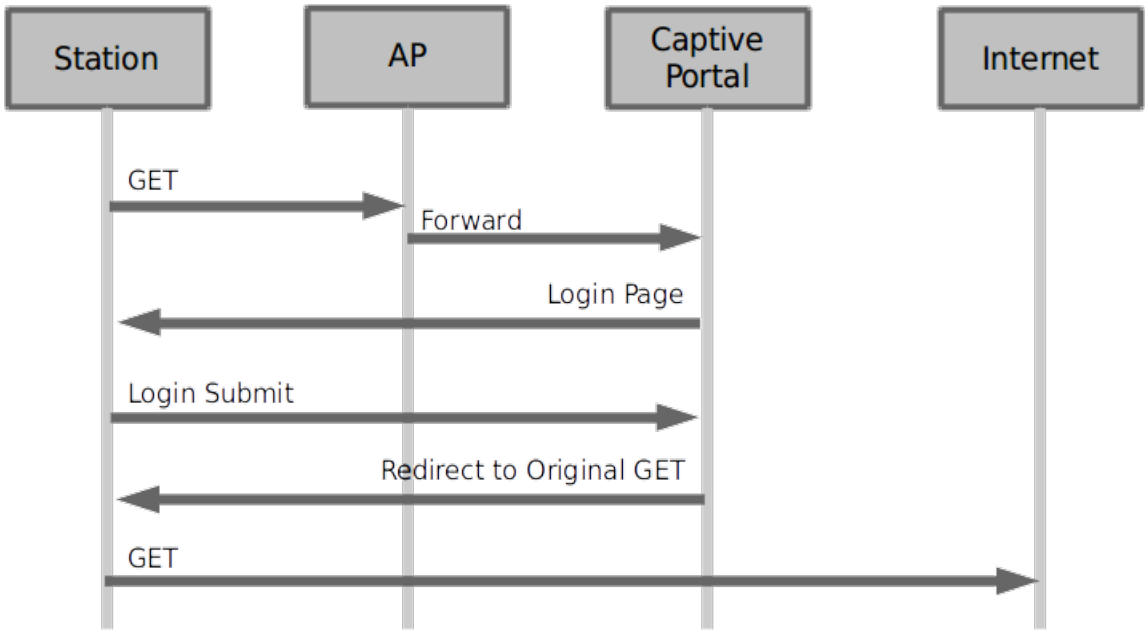


In this example, we will be testing agains a simple LAMP server on the upstream side of the AP. Do no use your LANforge server as the LAMP server because the routing will be difficult. In this chapter, a LAMP server is at 10.26.1.254, and there is an /etc/hosts entry for *basic-portal* to that address.

Basic Interactions of a Captive Portal

The basic order of operations of a captive portal are summarized in these steps:

1. A WiFi station accesses the LAN and is assigned a DHCP address.
2. The AP redirects any DNS and HTTP(s) request from the station. It returns either
   - a login page directly
   - a 301-Redirect to the login page
3. The station user submits this form. This form knows where to submit itself to, but it is possible that the form does not submit to the same address or service that it came from.
4. A successful authentication provides one of these responses:
   - The originally requested page, either as a 301-Redirect or as a proxied result.
   - A portal-div providing a logout or service menu and the original content inside.
   - A redirect page that uses javascript or meta-refresh mechanisms to tell the browser to reload the originally requested page.



Configuring a Demo Captive Portal

Provide Login/Logout pages

If you wish to set up a login and logout page on an Apache/PHP server to test with, you can copy the below files to the `/var/www/html` directory on the LAMP server.

login.php:

```
<!DOCTYPE html !>
<?php
$valid = true;
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   /* custom error reporting, see get_explanation */
   if (!array_key_exists('username', $_POST)) {
      header("HTTP/1.1 400 Bad Request");
      header("X-err-no: 9400");
      header("X-err-msg: missing username");
      $valid = false;
   }
}
?>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php if($valid) { ?>
   <title>Login</title>
<?php } else { ?>
   <title>Bad Request</title>
<?php } ?>
</head>
<body>
   <?php if ($_SERVER['REQUEST_METHOD'] == 'POST') { ?>
      <?php if(!$valid) { ?>
         <h1>Bad Request</h1>
      <?php return; } ?>

      <?= $_POST['username'] ?> access granted.
   <?php } else { ?>
      <form method="post" action="">
         Login:<input type="text" name="username" value="" /><br />
         <input type="submit" name="login" value="Login" />
      </form>
   <?php } ?>
</body>
</html>
```

Provide a Redirect in lieu of Portal Capture

Getting a redirect to the login page does not have to be very complex. The portal-bot script will first start off requesting whatever URL you wish, so request `http://basic-portal/start`. Here is an Apache configuration line to redirect that URI to `login.php`:

httpd.conf

```
<Location /start>
   Redirect /start /login.php
</Location>
```

After adding this redirect, restart your Apacher service using this command:

```
sudo apachectl configtest && sudo apachectl restart
```

Testing your redirect

You can use the command `curl -sqv http://basic-portal/start` to test out the redirect you just created.

```
 > curl -sqv http://basic-portal/start
* STATE: INIT => CONNECT handle 0x25bd9e8; line 1034 (connection #-5000)
* Added connection 0. The cache now contains 1 members
* STATE: CONNECT => WAITRESOLVE handle 0x25bd9e8; line 1071 (connection #0)
*   Trying 10.26.1.254...
* bind-local, addr: (nil)  dev: (nil)
* STATE: WAITRESOLVE => WAITCONNECT handle 0x25bd9e8; line 1151 (connection #0)
* Connected to basic-portal (10.26.1.254) port 80 (#0)
* Marked for [keep alive]: HTTP default
* STATE: WAITCONNECT => DO handle 0x25bd9e8; line 1229 (connection #0)
> GET /start HTTP/1.1
> User-Agent: curl/7.41.0-DEV
> Host: basic-portal
> Accept: */*
>
* STATE: DO => DO_DONE handle 0x25bd9e8; line 1314 (connection #0)
* STATE: DO_DONE => WAITPERFORM handle 0x25bd9e8; line 1441 (connection #0)
* STATE: WAITPERFORM => PERFORM handle 0x25bd9e8; line 1454 (connection #0)
* HTTP 1.1 or later with persistent connection, pipelining supported
< HTTP/1.1 302 Found
< Date: Fri, 04 Sep 2015 22:52:53 GMT
* Server Apache/2.4.7 (Ubuntu) is not blacklisted
< Server: Apache/2.4.7 (Ubuntu)
< Location: http://basic-portal/login.php
< Content-Length: 290
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://basic-portal/login.php">here</a>.</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at basic-portal Port 80</address>
</body></html>
```

Using the Portal Bot bash script

Before we get straight to working with `portal-bot.pl`, let's see how it is used. Your LANforge installation has an example script called `portal-bot.bash-example` for you to copy and modify. This script is intended for you to login and logout separately. The LANforge manager will call `portal-bot.pl` differently when building up the station or tearing down the station, these actions are similar:

i `./portal-bot.bash` will log your station in

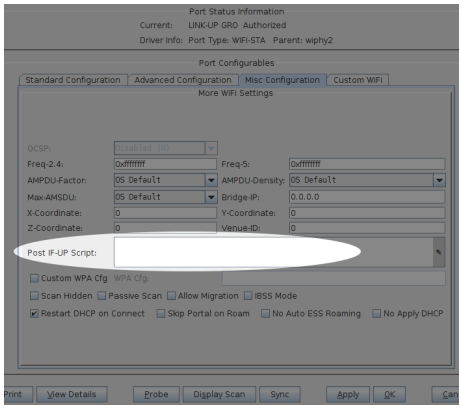
i `./portal-bot.bash --logout` will log your station out

Inside the bash script

The `portal-bot.bash` script is for exercising your `portal-bot.pl` script options from the command line while you develop with it. This is very close to the values you will place in the *Ports→Misc/Post IF-UP* field.

Switches you *won't* use in the GUI

You will never place the `PBOT_NOFORK` option in the *Ports→Misc/Post IF-UP* field because that will interrupt the processing of the LANforge Manager process. You will also never place `$*` in that field, either. You can place the `--verbose` and `--debug` flags in there, but it can fill your disk with log output more quickly.



Below is an example `portal-bot.bash` script with `\` line-continuation characters formatted for clarity:

```
PBOT_NOFORK=1 ./portal-bot.pl \
   --dev          sta100                      \
   --bot          bp.pm                       \
   --ip4          10.26.2.30                  \
   --dns          192.168.100.1               \
   --mgt          /dev/null                   \
   --delays       0,1,3                       \
   --user         "bob"                       \
   --pass         "secret"                    \
   --ap_url       "http://basic-portal/"      \
   --start_url    "http://basic-portal/start" \
   --login_form   "login.php"                 \
   --login_action "login.php"                 \
   --logout_url   "logout.php"                \
   --verbose --debug $*
```

Below is the same script using short switches:

```
PBOT_NOFORK=1 ./portal-bot.pl \
   -i    sta100                      \
   -b    bp.pm                       \
   --ip4 10.26.2.30                  \
   --dns 192.168.100.1               \
   --mgt /dev/null                   \
   --delays 0,1,3                    \
   -u    "bob"                       \
   -p    "secret"                    \
   -a    "http://basic-portal/"      \
   -s    "http://basic-portal/start" \
   -n    "login.php"                 \
   -o    "login.php"                 \
   -t    "logout.php"                \

   -v -d $*
```

Using the portal-bot.bash command on the command-line:

A common misconception is thinking that `$*` is a command-line argument. It is only used in bash scripts. Do not put `$*` on the command-line.

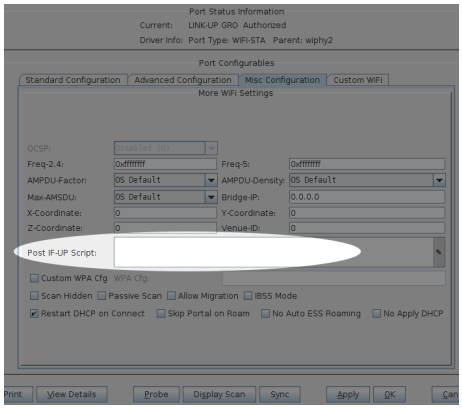
```
PBOT_NOFORK=1 ./portal-bot.pl -i sta100  -b bp.pm --ip4 10.26.2.30 \
        --dns 192.168.100.1 --mgt /dev/null -u "bob" -p "secret" \
        -a "http://basic-portal/"  -s "http://basic-portal/start"
        -n "login.php" -o "login.php" -t "logout.php" -v -d
```

Using the `portal-bot.pl` perl script

Tips:

- First thing to do: edit a copy of that script and adjust it for your station device and it's IP address.

- Add `-d` to add more debugging messages. That makes `dbg()` statements print.
- Add `--print` after you get the script to work. This will print out the format of the arguments useful for putting the statements into the GUI *Ports→Misc/Post IF-UP* field.



The first six arguments are provided by LANforge when you use `portal-bot.pl` with a station. You want to populate these in your bash script, but not in the *Post IF_UP* field.

PBOT_NOFORK
: This environment variable tells the `portal-bot.pl` script to not fork. **Use it only when developing.** Omitting this is normal and allows for multi-processing of web requests from LANforge.

-i
: station name

--bot
: The bot plugin you provide

--ip4
: The IP of the station. This script is useless if there has been no DHCP lease.

--ip6
: Use `''` for no IPv6 address.

--dns
: The DNS addresses provided from the DHCP lease

--mgt
: The FIFO that signals the LANforge server. You don't use it when testing.

The second set of arguments describe your own AP environment:

--user | -u
: portal user name

--pass | -p
: portal user password

--ap_url | -a
: A string to prepend to URLs when talking to the AP. Not necessary, but if you don't use it, you have to provide fully qualified URLs to *--login_form*, *--login_action*, and *--logout_form*.

--start_url | -s
: The first URL requested from the AP, this should provide either a login page or a redirect to a login page. If you get your destination page (like, if you request baidu.com and actually get it), your station has probably not been logged out from the captive portal.

--login_form | -n
: This is what you request to get a login form. Often it is returned in the redirect, but sometimes you cannot get a cookie assignment if you do not request it specifically.

--login_action | -o

Submit your login credentials to this URL.

--delays

Comma separated list of seconds to delay at certain points:

1. `$::delays[0]` Used to delay the very first 'start_url' GET request
2. `$::delays[1]` Used to delay the first POST request in 'submit_login'
3. `$::delays[2]` Used to delay the 'submit_logout' request.
4. `$::delays[3+]` Your bot can utilize further delays if you specify

You may specify skips by adding a zero: `--delays 1,0,2`

You may specify a random time by using 'random': `--delays 1,random,2`

You may specify just one time for all delays: `--delays 2`

You may specify a random range: `--delays 3-20,4-25`

--logout_form | -t

Submit to this URL to log out of the captive portal

-v -d

Verbose and debug output, respectively.

--print

Skips process and prints out formatted arguments.

$*

Expands to all remaining shell arguments

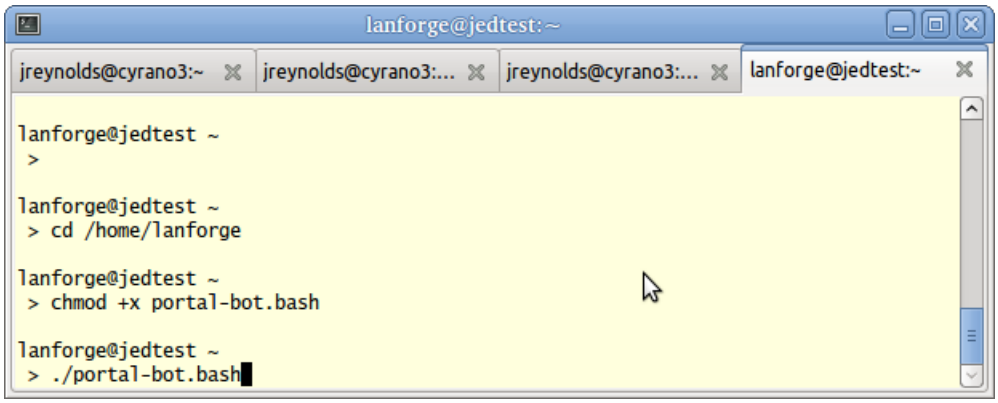
We will connect to our LANforge system*. You want to copy this file to your own `./portal-bot.bash` file, edit it and then make it executable.

i * You can connect via VNC, PuTTY or other SSH client.

i Use **`chmod +x portal-bot.bash`** to make your script executable.

Now let's see how to use this script with station **sta100**. Run the commands:

```
$ cd /home/lanforge
$ chmod +x portal-bot.bash
$ ./portal-bot.bash
```



You will see a lot of output, it will show the contents of the web pages it finds.

```
INTERPRET given HTTP/1.1 200 OK
Date: Fri, 04 Sep 2015 22:57:12 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.11
Vary: Accept-Encoding
Content-Length: 192
Content-Type: text/html

<!DOCTYPE html !>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Login</title>
</head>
<body>
        bob access granted.
   </body>
</html>found ref to data?         bob access granted.result for sta100: 0Kportal-bot result for sta100: 0K(elapsed=0.0452320575714111 0.045232057
5714111)

lanforge@jedtest ~
 >
```

Watching the Logs

Typically you won't need to look at this output in the terminal, and you will not add **-d -v** flags to your LANforge stations. You very likely will need to check the log output from these scripts in case you need to diagnose connection problems during your test. Each virtual station leaves a log in the `/home/lanforge/wifi` directory, like `wifi/portal-bot.sta100.log`

> Watch logs using *tail*: `tail -F wifi/portal-bot.sta100.log`

Executing the LANforge `curl` commands yourself

To find the actual `curl` commands being executed, you want to `grep` the logs. Below is an example of grepping the logs and running the curl command.

```
 $ cd /home/lanforge/wifi
 $ grep Submitting portal-bot-sta100.log
Submitting: /home/lanforge/local/bin/curl -sLki -c /tmp/sta100_cookie.txt -b /tmp/sta100_cookie.txt -4
Submitting: /home/lanforge/local/bin/curl -sLki -c /tmp/sta100_cookie.txt -b /tmp/sta100_cookie.txt -4
```

You might noticed that some of the commands in the log might appear repeated, there are areas of redundant logging. There is a case where you can legitimately see repeated commands: when you have an *Post IF_UP* value configured for the port you are testing with. (Remember that the *Post IF_UP* field should be blank when developing the script.)

Remember, this `curl` command cannot be run without first doing a `source /home/lanforge/lanforge.profile` in your shell (our curl is a custom build). Here is an example. We take a command similar to the one above, add `-qv` and cancel it using Ctl-C :

```
 $ cd /home/lanforge
  $ source lanforge.profile
  # add a -qv to see header details
  $ /home/lanforge/local/bin/curl -qv -sLki -c /tmp/sta100_cookie.txt -b /tmp/sta100_cookie.txt -4 --in
* STATE: INIT => CONNECT handle 0xa80158; line 1397 (connection #-5000)
* Added connection 0. The cache now contains 1 members
*   Trying 10.51.0.254...
* TCP_NODELAY set
* bind-local, addr: 10.41.4.223  dev: sta100
* SO_BINDTODEVICE sta100 failed with errno 1: Operation not permitted; will do regular bind
* Name 'sta100' family 2 resolved to '10.41.4.223' family 2
* Local port: 0
* STATE: CONNECT => WAITCONNECT handle 0xa80158; line 1450 (connection #0)
^C
```

Explaining the `curl`Command

There are many arguments to the curl command, but in general, you should be able to copy and paste the command into a terminal and it should work (see note about `lanforge.profile` above). Below is an example of a `curl` command, with `\` characters as line-continuation marks, formatted for clarity.

```
 $ /home/lanforge/local/bin/curl -qv \
    -sLki                        \
```

```
   -c /tmp/sta100_cookie.txt       \
   -b /tmp/sta100_cookie.txt       \
   -4                              \
   --interface sta100              \
   --localaddr 10.41.4.223         \
   --dns-interface sta100          \
   --dns-ipv4-addr 10.41.4.223     \
   http://basic-portal/start
```

Switch	Example Value	Purpose
-q		Suppress page output
-v		Verbose, prints diagnostic steps
-s		Suppresses page output
-L		Follow redirects
-k		Suppress certificate validation errors
-i		Print HTTP headers
-c	sta100_cookie.txt	Send cookies from file
-b	sta100_cookie.txt	Save cookies to file
-4		Use IPv4
--interface	sta100	bind to this interface
--localaddr	10.41.4.223	bind to this address
--dns-interface	sta100	send DNS queries from this interface
--dns-ipv4-addr	10.41.4.223	bind to this address when sending DNS queries
--dns-interface	sta100	send DNS queries from this interface
-X	GET	Use HTTP GET method
	POST	Use HTTP POST method
-d	'username=bob'	URL encoded form parameters used during POST method

Your `portal-bot.bash` script is intended to be a way of focusing on the development of your bot plugin and not repetitively typing a long curl command.

Writing your Bot Plugin

Your bot plugin, the Perl module you will write for your captive portal, is central to the operation of the *portal-bot.pl* script. It is also important that you do not alter the `portal-bot.pl` script unless absolutely necessary, because your changes could be overwritten by upgrades. Any alteration to the time at which the `fork()` call is made in this script can make the LANforge server grind to a halt.

> i Only edit your bot perl module, please.

The Bot Subroutines

The example bot, *bp.pm*, provided with LANforge defines four subroutines. In order:

find_redirect_url

This subroutine receives the response of the HTTP(S) GET of your *--start_url* parameter. Look through this to see if:

- you are already getting destination content--if so, you were not logged out,
- you get a login form directly and not a redirect,
- or you get a redirect to a login page (possibly on a separate port like :8080)

If you get a redirect to another port, compare the *--login_url* value to this. If it is different, consider updating your login_url parameter.

There might be many form parameters, like ones for a session id, a `PHP_SESSID`, a cookie, a base64 encoded string indicating your originally requested url (or just a plain URL-encoded url), and any possible co-branding parameters that might indicate any advertising campaigns associated with this captive portal. Missing some of these might make submitting the form give you an error. Store these values as necessary in your `bot::` namespace. You do not submit your login page in this method.

> Define a package scope variable using `our $thing;` after your *package* statement.

submit_login

Here is where you submit your login page forms. The `botlib::request()` function is provided to make GET and POST requets verbose logging and debugging. The page is returned as lines in the `@response` array.

```
my $post_data = "username=".uri_escape($user_name);
my @response  = ();
request({'curl_args'=> $::curl_args,
  'url'       => $post_url,
  'method'    => 'POST',
  'delay'     => '0,3',       # see --delays option
  'post_data' => $post_data,
  'print'     => 1},          # turns on debugging
  \@response);
```

The submit_login function uses the `$::delay[1]` parameter if `--delays` were set. See paragraph on randomDelay.

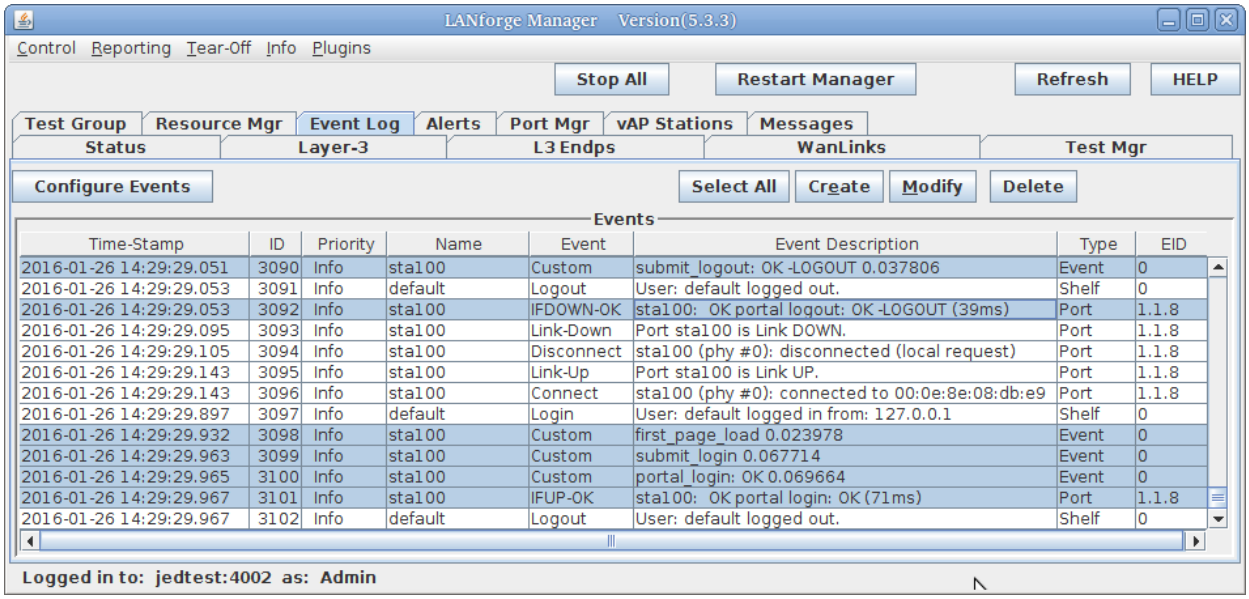
interpret_login_response

Here you determine if you are getting an access denied error or are being forwarded to your original *start_url* destination. Set your `$result` variable to **OK** or **FAIL**. Use the *logg()* method to add information for the wifi/portal-bot log.

In order to add events, such as page load time, you want to use the `botlib::newEvent()` function:

```
my $page_time = botlib::time_milli() - $::start_at;
newEvent("portal_login: $result", $page_time, $::dev);
```

Your event log will gain messages like these:



get_explanation

Some web applications can provide customized error messages in their response. You can add a `get_explanation()` function to your bot to collect this information. The `botlib::dbgdie()` method will

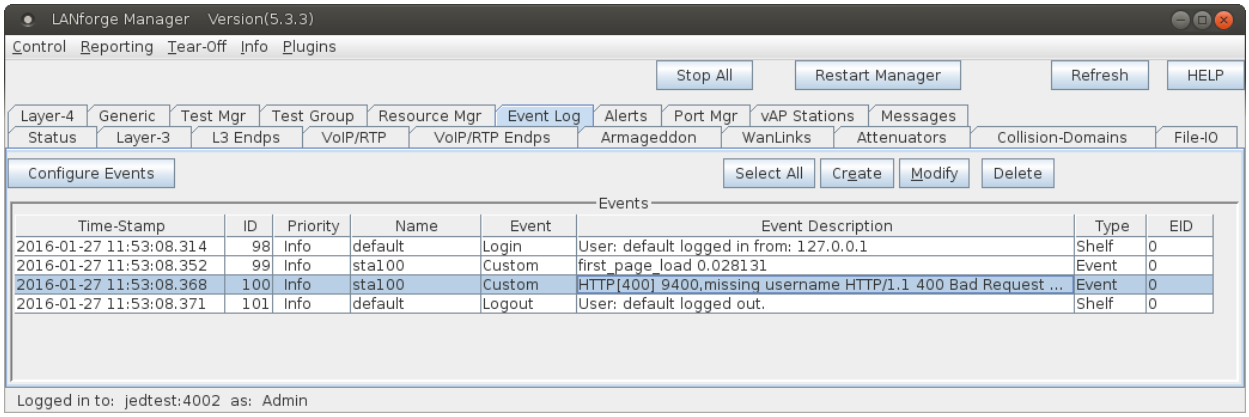
take advantage of this method if available. Below is an excerpt from the method found in `bp.pm`:

```
sub get_explanation {
   for $line (@$ra_result) {
      ($err_code) = $line =~ /^X-err-no: (.*)$/
         if ($line =~ /^X-err-no: /);
      ($err_msg ) = $line =~ /^X-err-msg: (.*)$/
         if ($line =~ /^X-err-msg: /);
   }
   return "$err_code, $err_msg";
}
```

Notice how this parses out the HTTP headers found if the parameter `username` were missing when doing a POST to `basic-portal/login.php`:

```
header("X-err-no: 9400");
header("X-err-msg: missing username");
```

You will see these messages show up in the LANforge Events log:



submit_logout

Many captive portals do not publicise their logout URLs, so it might be available only on an admin page for the AP. You will know when the *logout_url* parameter works if you can do a logout with that station, and then successfully log back in using the same station and seeing the captive portal sign-in page again.

randomDelay

The delay parameter to `botlib::request()` has many overloads to the call:

- A simple number is a simple delay in seconds. No other units are used.
- If you specify 'random' in the *delay* parameter, the `botlib::randomDelay()` is called, producing a range between [1 - 119] seconds.
- If you specify '3-16', `randomDelay(3, 16)` is called to produce a random range between [3 - 16] seconds.
- If you specify two numbers separated by a comma, it looks at your `@::delays` list, and picks the second argument if it can, the last item of `@::delays` if the list is too short, or the first argument if there are no items in the delay list.

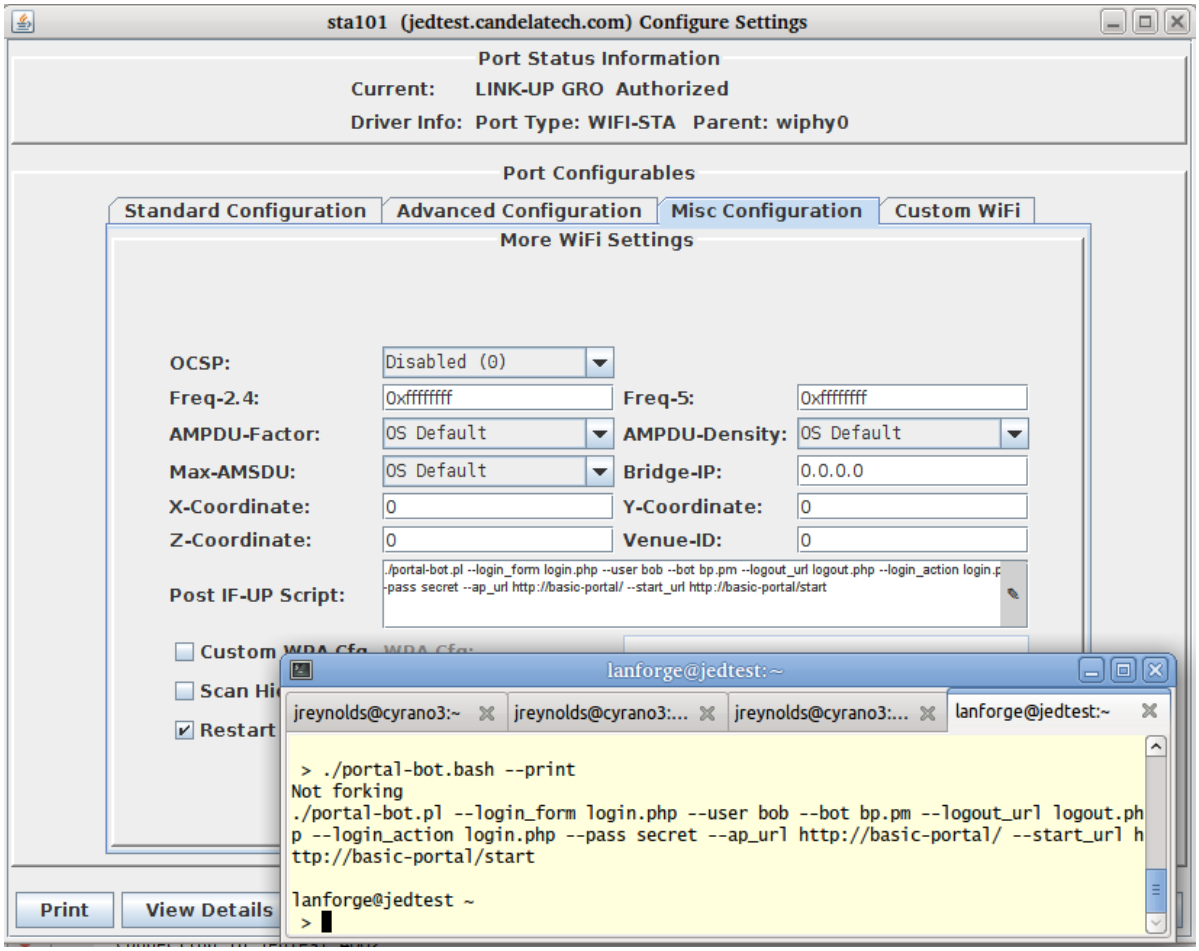
We have now covered all of the scripting development areas for the `portal-bot.pl` plugin you will write.

Configuring your Stations

A Single Station

We assume you have `portal-bot.bash` working at this point. This is how you can configure a single station:

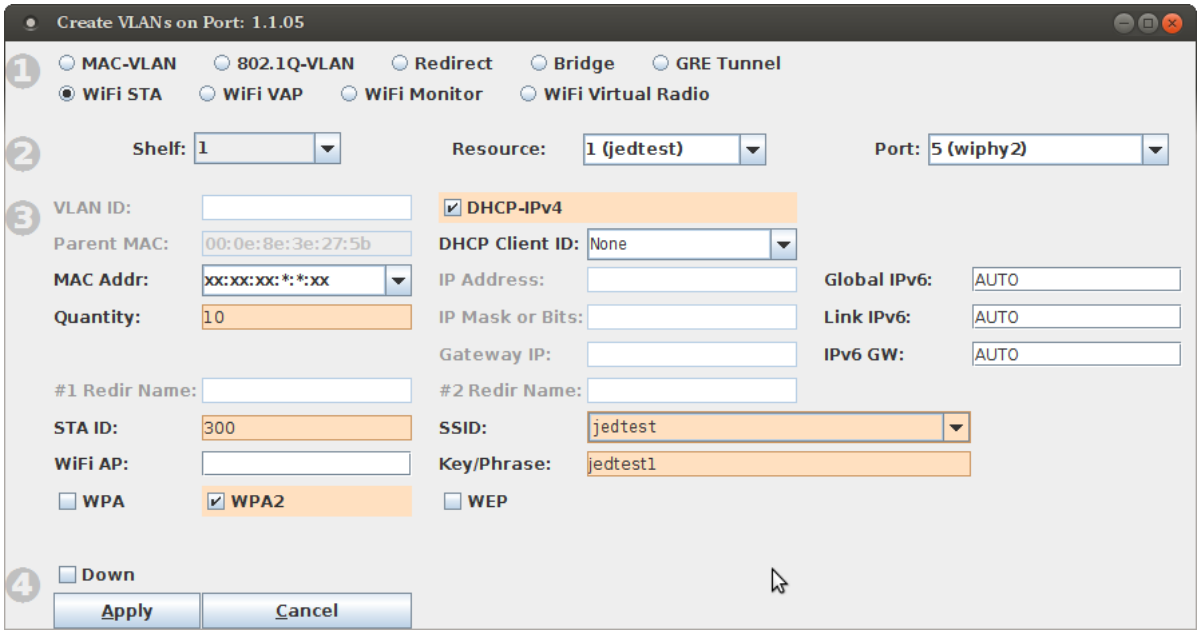
1. Use the `portal-bot.pl --print` command to print out the arguments.
2. Copy the result (starting with "portal-bot.pl") into the Port->Misc window. *Avoid populating this field while you are developing the script!* If you place a value into that field, your portal-bot script will not only execute, but the Manager process will also execute the script specified in the POST_IFUP field. This can be really confusing.



Multiple Stations

To get multiple virtual stations logging in an out using the GUI, we just need a few of those parameters for the station configuration. We will use the *Batch Modify* feature to alter a series of stations.

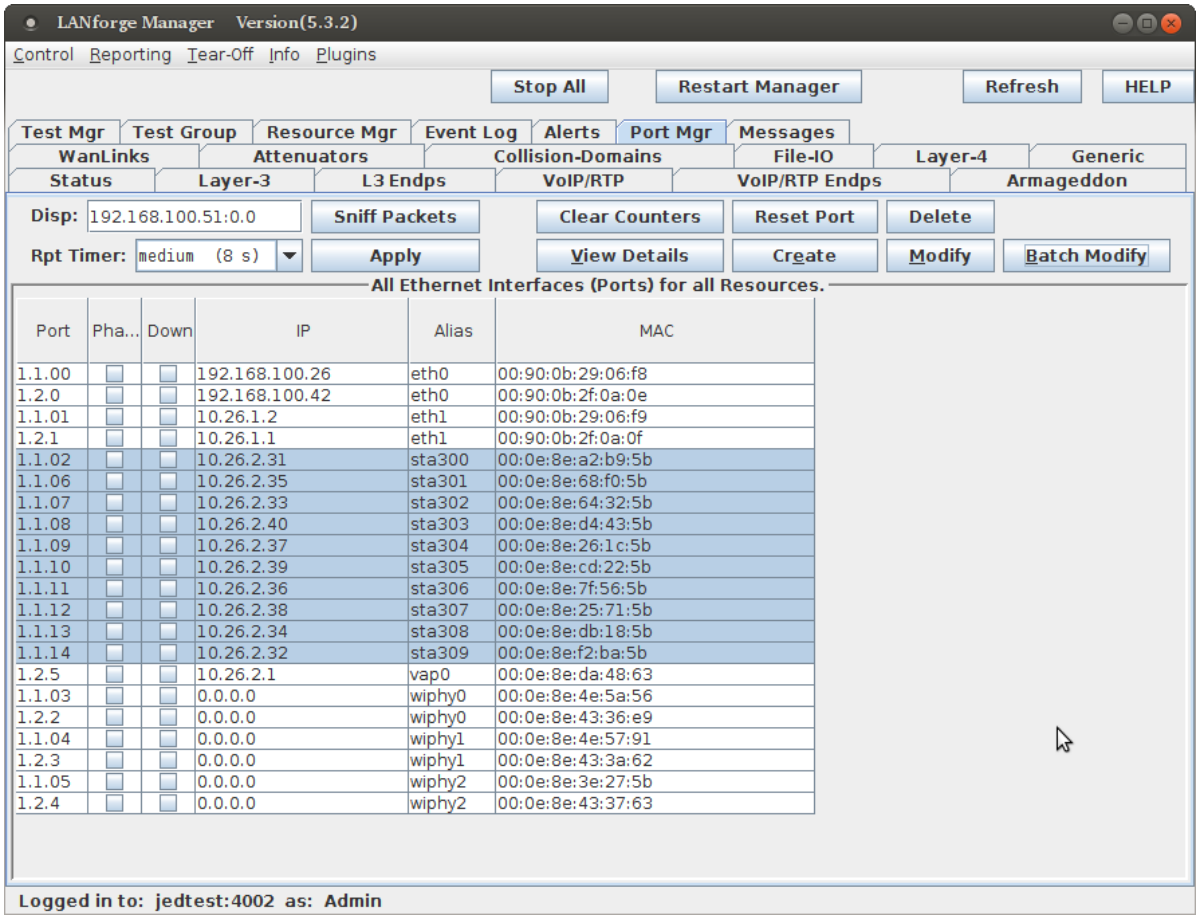
1. In the **Ports** tab, create a series of stations. In this example we will create them with:



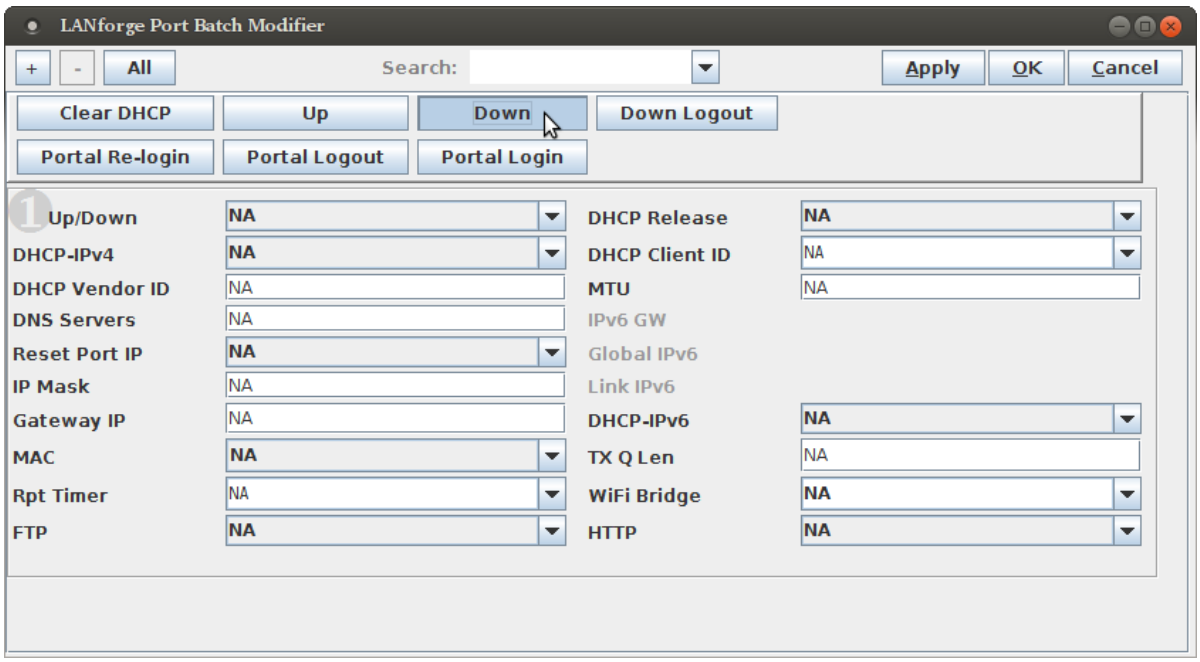
   - Port: wiphy2
   - Select DHCP-IPv4
   - Quantity: 10
   - STA ID: 300
   - SSID: jedtest

- Passphrase: jedtest1
- Select WPA2
- Select Down

2. Highlight them and click **Batch Modify**.



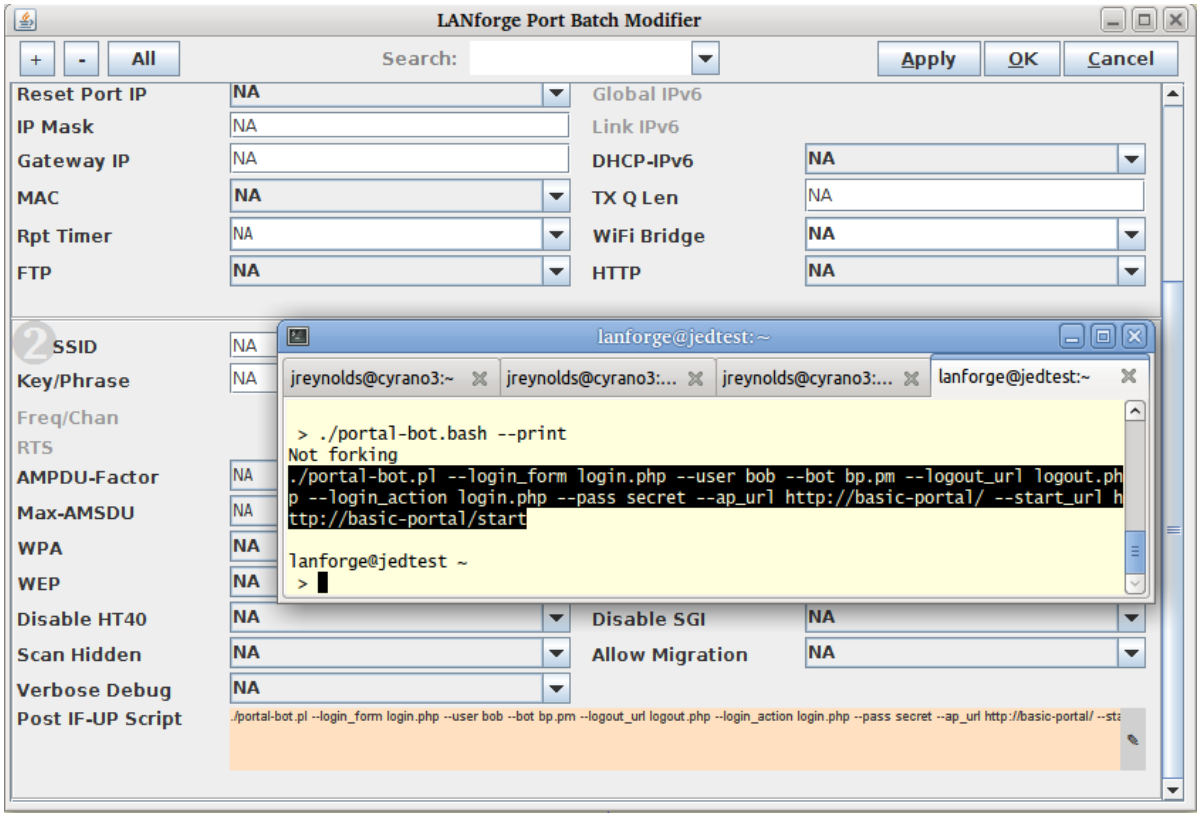
3. Click the **Down** button.



4. In your terminal, invoke the `portal-bot.bash` with the `--print` argument:

```
./portal-bot.bash --print
portal-bot.pl --bot bp.pm --user bob --pass bob1 --ap_url http://basic-portal/
 --start_url  http://basic-portal/start --login_form login.php --login_action login.php
 --logout_form logout.php
```

5. Use the **[+]** button to expand to Box 2. We will enter the following version of our command into the *Post IF-UP Script* area. (The picture shows the short switches.)



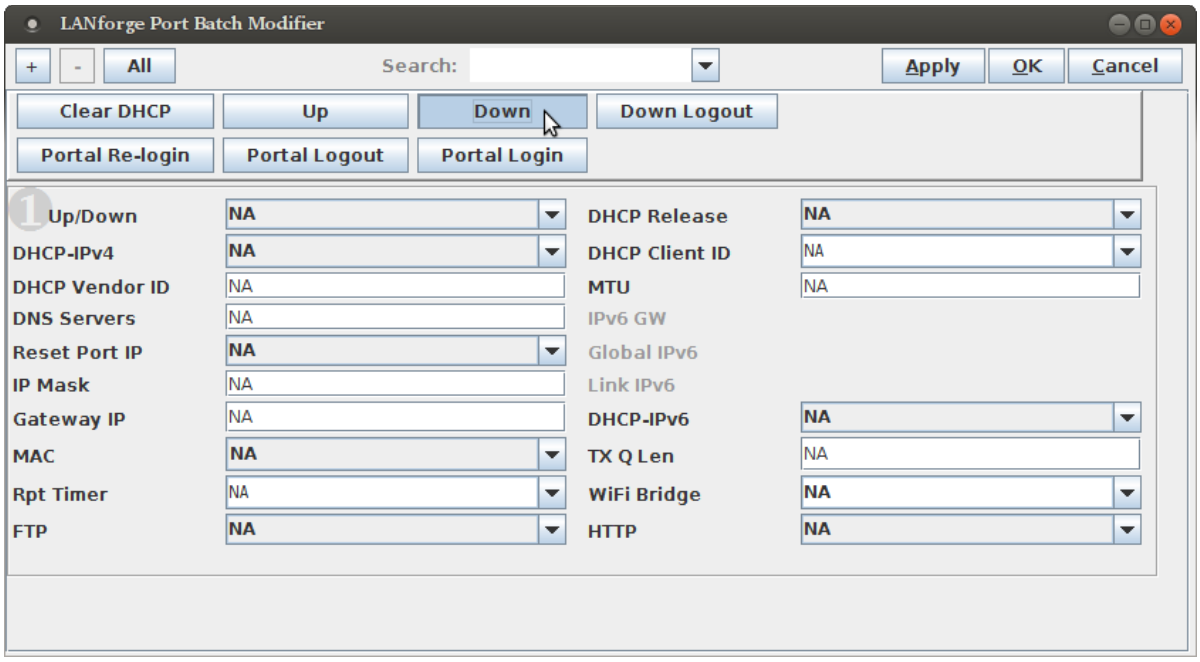
Click **OK**

6. In the **Ports** tab, double click *sta300* and in the *Misc Configuration* tab, you will see the *Post IF-UP Script* values.

Testing a Station

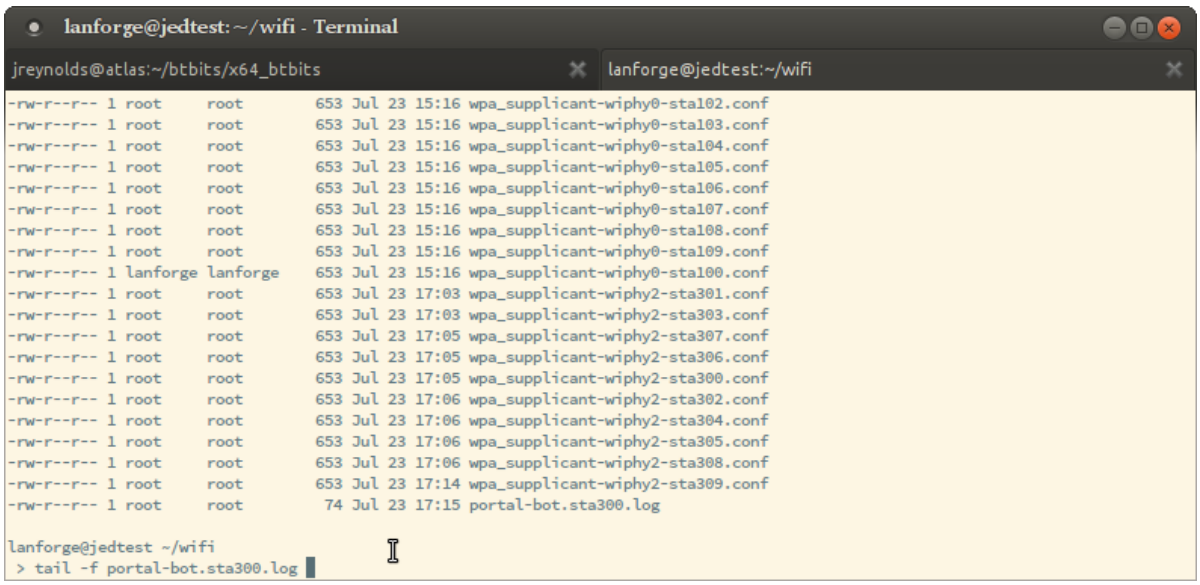
Exercising these stations starts with bringing them up and down using the *Batch Modify* tool.

1. Highlight one station, `sta300`, and click **Batch Modify**.
2. Click the **Down** button to admin-down the station.



3. In a shell on the LANforge, got to `/home/lanforge/wifi` and tail the log for station 300:

```
tail -f portal-bot.sta300.log
```

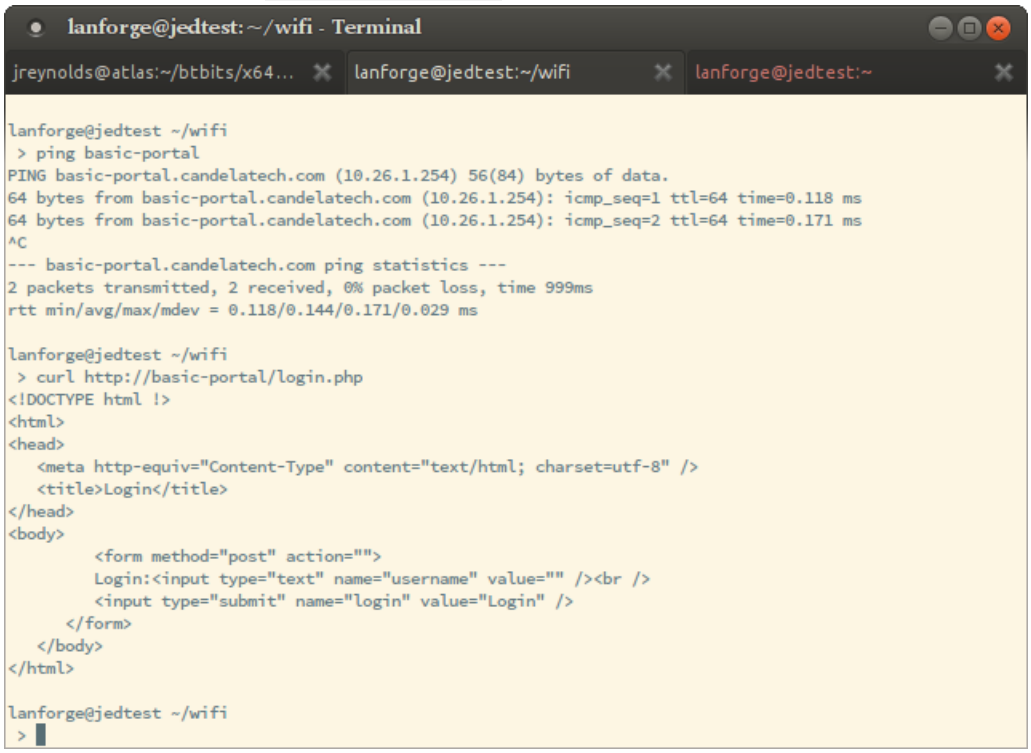


4. Click the **Up** button to admin-up the station.
5. Click the **Portal Login** button force the station to login if you do not see any messages in the log file you are tailing.

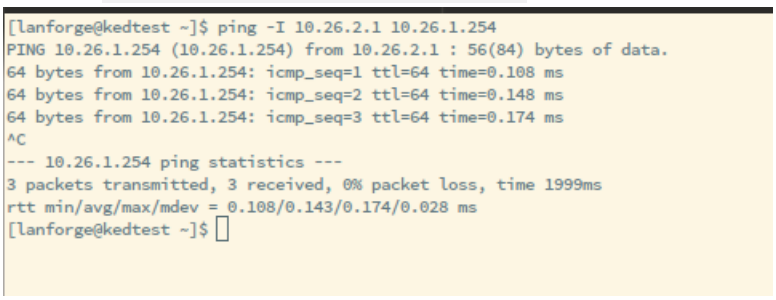
Troubleshooting Techniques

If your station cannot talk to the captive portal, like you have a time-out, these steps will help identify where there is a misconfiguration:

1. **Ping the portal from LANforge**: `ping basic-portal`



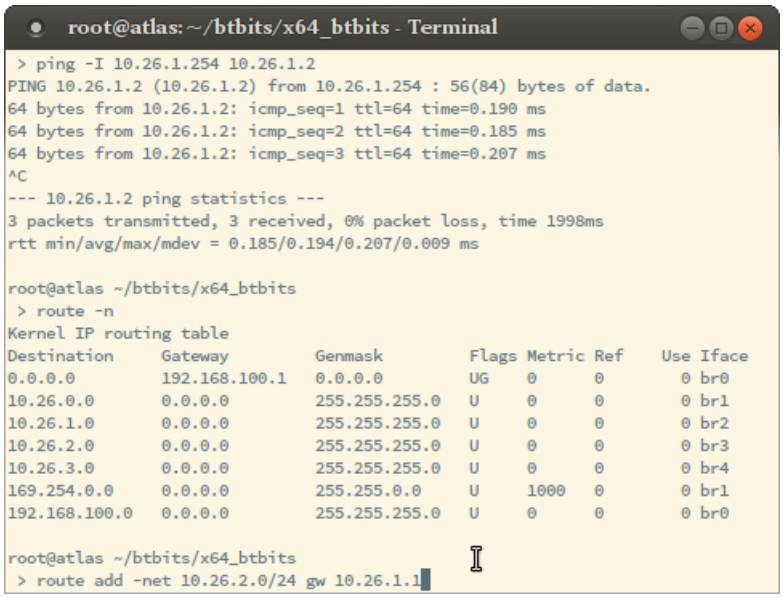
2. **Ping the portal from *sta300***: `ping -I 10.27.0.16 basic-portal`



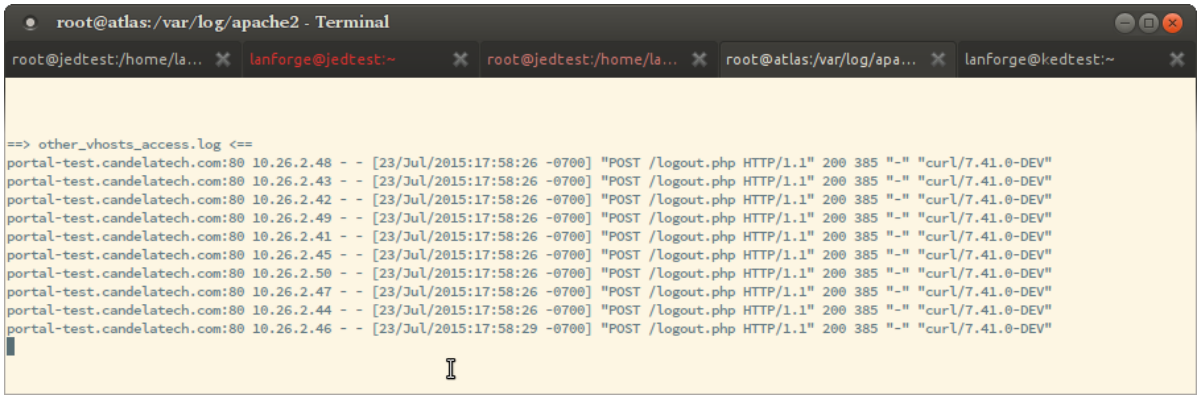
3. **Use curl** to download the portal page by hand: `curl -sqv http://basic-portal/login.php`
4. **Check the route** on the portal side if you are routing. Some examples:

```
route -n
```

```
route add -net 10.27.0.0/23 gw 10.26.1.1
```



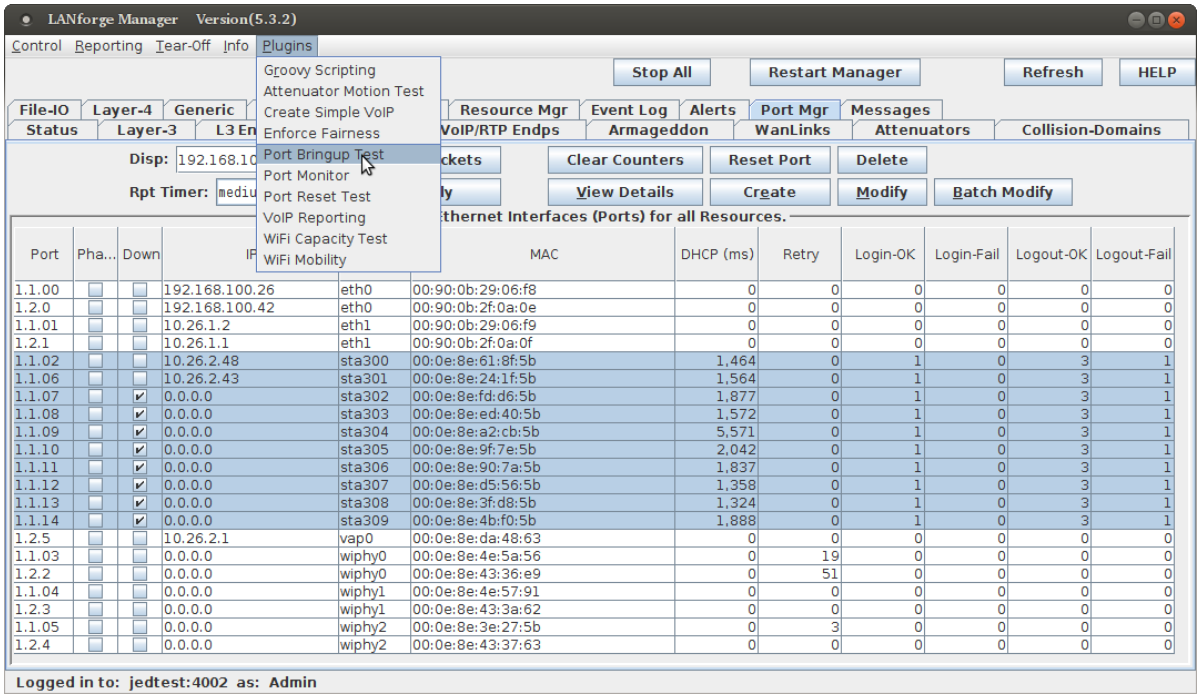
5. **Check access logs for the portal.** There might be a hostname issue.



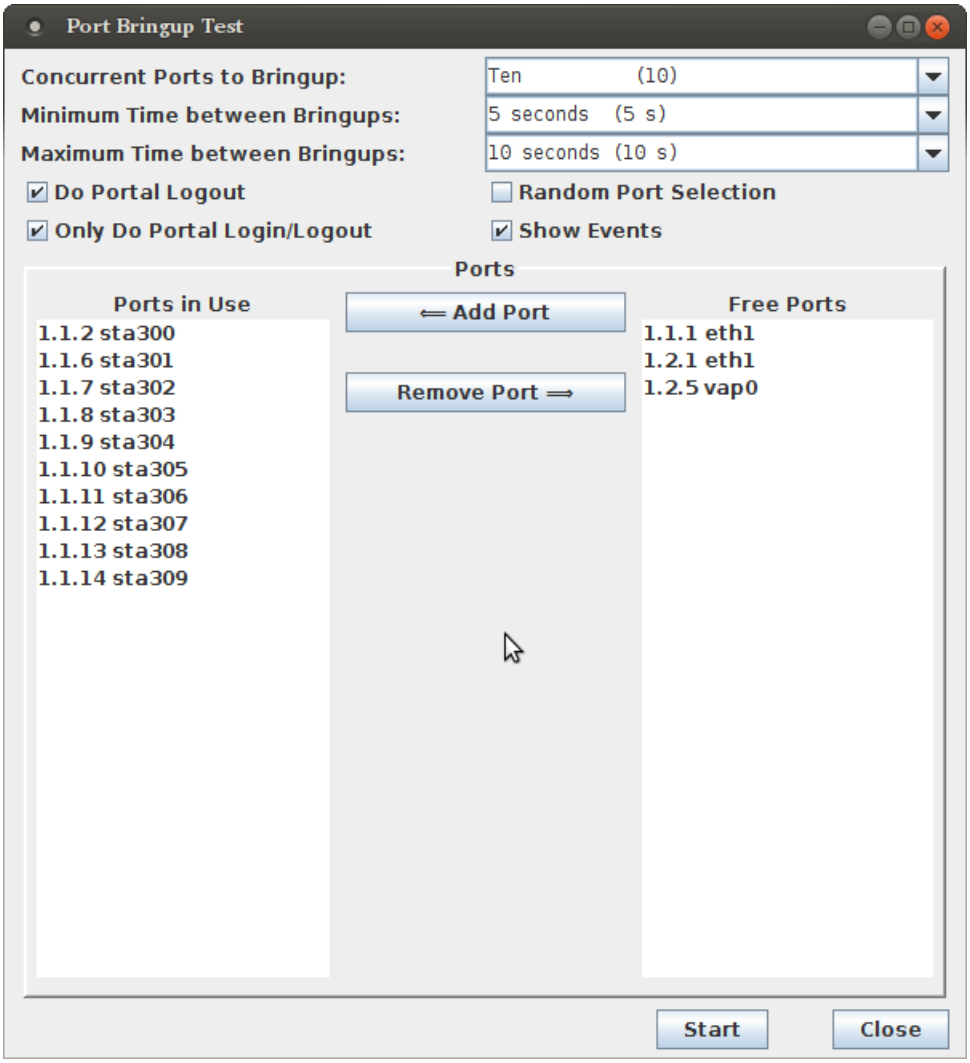
Using the Port Bringup Plugin

Using the Port Bringup Plugin is a much more fun way to get data than looking at log files.

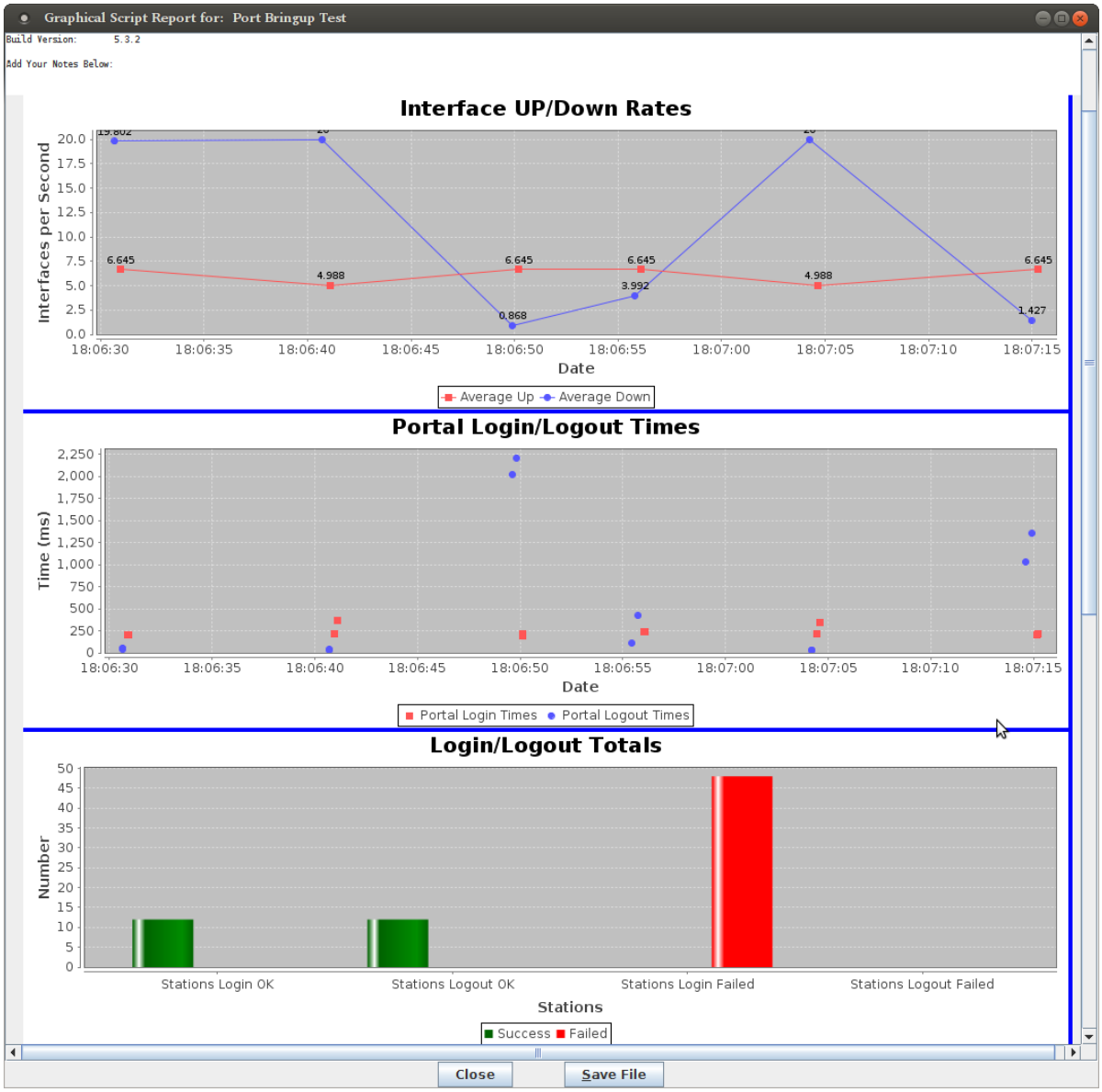
1. In the *Plugins* menu, select **Port Bringup Test**.



2. Highlight a series of stations and click **Add Port**:



3. Click **Start**
4. You will see the reporting window. It often takes many seconds or a few minutes for stations to aquire DHCP addresses and start reporting information into the plugin.



Candela Technologies, Inc., 2417 Main Street, Suite 201, Ferndale, WA 98248, USA
www.candelatech.com | sales@candelatech.com | +1.360.380.1618