

Querying the LANforge GUI for JSON Data

Goal: The LANforge GUI now has an embedded webserver and a headless mode of operation. Read on for how to configure and query the client for JSON formatted data.

Updated 2019-11-21: New features in 5.4.1.

Some of the CLI API parameter names have changed. Notably:
 nc_show_ports flags changed to probe_flags. Be aware that older scripts might break on upgrade.

Updated 2018-07-24: New features in 5.3.8.

The LANforge GUI (as of release 5.3.6) can be configured to start an embedded web server that can provide data about ports and layer-3 connections. This service can be queried with with any browser or AJAX connection. We're going to increasingly refer to it as the LANforge **client**. This feature provides these benefits:

- More rapid polling: using CLI scripts to poll ports on the LANforge manager can add stress and contention to the LANforge manager; polling the GUI will not tax your test scenario.
- Expanded array of data: the views found in the GUI, like Port Mgr and Layer-3 tabs, contain synthesized data columns not available through the CLI scripting API. Most of these columns can be returned in JSON format.
- Reduced effort when integrating with third party test libraries: many other testing libraries expect JSON formatted input.
- Web socket delivery of event data allows real-time reporting of interface changes and station scan results. This is also a channel for querying additional diagnostic data.
- There is a /help web page that allows you to build POST commands.
- A headless -daemon mode that will run the client without any GUI windows. This requires much less memory and has been queried for weeks at a time without crashing or memory leaks.

Present and potential drawbacks of the JSON feature:

- Actively being developed: the JSON views/schema of the objects is at a demonstation state. URLs and JSON structures have changed in 5.3.8.
- Now no longer possible to create Groovy plugins to add JSON features if you want to use the headless mode. JSON Features are compiled into the LANforge GUI from Java sources.
- In 5.3.8 we have limited the view of ports, have added URLs to post direct CLI commands, and have applied HTML application/x-www-formurlencoded form posting submissions in name/value pairs. There is no multipart/form-data JSON submission at this time.

Client Settings

The LANforge GUI is started using a script (lfclient.bash or lfclient.bat). From a terminal, we call that script with the -httpd switch. By default the GUI will listen on port 8080:

- \$ cd /home/lanforge
 \$ (lfaliant hash http://incode/lanforge)
- \$./lfclient.bash -httpd

You can specify the port to listen on:

\$./lfclient.bash -httpd 3210



System

Under Test

You can run the client headless with the -daemon switch as well:

\$./lfclient.bash -httpd -daemon

There is a setting in the 5.3.8 *Control→Preferences* menu for setting a *minimized mode* and the HTTP port number as well.

2 LANforge-GUI Preferences		
Caparal LANFarras Managar Taba		
Confirmations		
Confirm Exit	Confirm Netsmith Sync	
🔲 Auto-Submit	Skip delete warning in Reporting Manager	
🗹 Skip delete warning in Netsmith		
Display		
🗹 Anti-Alias lines in Netsmith	Invert RX-Signal X Axis	
🗹 Show Commas	Enable system look and feel (requires restart)	
Display Graph Duration: 2 Minutes (2 m	min) 🔻 Dynamic Report Duration 30 Minutes (30 min) 💌	
Set Extended Title		
Features		
🗹 Poll Mgr		
☑ Enable HTTP Service	HTTP Port 8080	
🔲 Run GUI Minimized	User ID	
Enable hotkey for entity deletion (A/t-t)	
Tool Tips		
☑ Show Tooltips	Duration(ms) 10000 V Delay(ms) 250 V	
R	eset Config OK	

Making Queries

From the terminal we can query the port to find a basic message from the GUI:

```
$ curl -sq http://localhost:8080/
```

This first page (/) will give you a JSON list of the resource URLs available. Most URLs will provide JSON as their default content type. Notably, /help defaults to HTML.

\leftrightarrow \rightarrow C \textcircled{a}) localhost:8080 👽 🏠 🔍 Search 🔯 🛃 🗐	
JSON Raw Data Headers	5	_
Save Copy	♀ Filter JSON	Ī
handler:	"candela.lanforge.HttpStatus"	
uri:		
text:	"These urls are presently available"	
mappings:		
4	"(This page) Provides status and configuration."	
<pre></pre>	"Post key-value command values to a specifi command, eg: -H 'Content-type: application/x-www-form-urlencoded' -X POSTdata 'shelf=1& resource=1&port=1' http://cholla4:8088/cli-form/nc_show_port"	
▼/cli:	"Post literal CLI commands, eg: -H 'Content-type: application/x-www-form-urlencoded' -X POSTdata-urlencode 'cmd=nc_show_port 1 1 1' http://cholla4:8080/cli"	
<pre>/cli-json/:cmd:</pre>	"Post json object command values, eg: -H 'Content-type: application/json' -H 'Accept: application/json' -X POST '{\"alias\":\"cx0001 \",\"state\":\"RUNNING\"}'h <i>ttp://cholla4:8080/cli-json/set_cx_state</i> "	
/alerts/:	"List alerts that have been reported"	
<pre>/resource/:shelf_id:</pre>	"Lists ports on resource 1.2, eg: /resource/1/2/, see /port/:shelf_id/:resource_id/list"	1
▼ /=:/.	"(This made) Provides access to less formatted data and other hacks. see /misc/helo."	1

By default, most URLs will treat a default Accept: */* header as text/html. Compare the two techniques below:

JSON Output



Clearly, the JSON output is difficult to read. We cover formatting output below.

HTML Output

Most of the queries to the client will return JSON by default. The notable exception is the /help URL. To get HTML output in the terminal, you have to specify Accept: text/html to curl:

Formatting Results

JSON formatted text is pretty difficult to read, there are a few different utilities that can help you look at it: jq, json_pp, json_reformat, tidy, xmllint, yajl and jsonlint.

Example of installing formatters

On Fedora, install:

\$ sudo dnf install -y jq perl-JSON-PP tidy libxml2 yajl

On Ubuntu, install:

\$ sudo apt install -y jq libjson-pp-perl perltidy xmllint libxml2-utils yajl-tools

Now we can perform a query:

```
$ curl -sq /port/1/1/1
{
    "candela.lanforge.HttpPort" : {
        "duration" : "1"
    },
    "handler" : "candela.lanforge.HttpPort$JsonResponse",
    "interface" : {
        "stuff":...
    },
    "uri" : "port/:shelf_id/:resource_id/:port_id"
}
```

Notice that the URI object list paths with colon-tagged positions in them, e.g.: /cli-form/:cmd. These are interpreted as URL parameters and not query string parameters, they cannot be moved into the query string.

Making your shell friendly

To save you typing, you might want to add this function to your .bash_aliases file:

```
function Json() {
    curl -sqv -H 'Accept: application/json' "http://localhost:8080${@}" \
    | json_reformat | less
}
```

```
}
```

Then you can make your calls this way:

\$ Json /port/1/1/1

Browsing results in table format

We can view a URL in a browser as well:

/port	× +					
← → ♂ ଢ	i) localhost:8080/pc	ort/1/1/1/	··· 🛛 🕁 🤇	🔾 Search	III\ 🙂	∎" » ≓
EID AP Activity (Channel Devid	e Down	IP Parent Dev	Phantom	Port SSID	
					1010 0010	
1.1.1 0.0	eth1	false 0.0	0.0.0	false 1	1.1.01	

Viewing Alerts and Events

You can both view and stream event data. Querying events and alerts are both quite similar:



A busy LANforge system will generate hundreds of thousands of events. Only the last few thousand can be recalled.

You can inspect a singular event:

```
$ Json /events/2249259
{
   "handler": "candela.lanforge.HttpEvents$FixedJsonResponder",
   "uri": "events/:event_id",
    "candela.lanforge.HttpEvents": {
        "duration": "0"
   },
    "event": {
        "eid": "1.3.21",
        "entity id": "NA",
        "event": "Connect",
        "event description": "sta3106 (phy #1): connected to 00:0e:8e:d5:fa:e6",
        "id": "2249259",
        "name": "sta3106",
        "priority": " Info",
        "time-stamp": "2018-07-24 14:39:33.776",
        "type": "Port"
    }
```

We can view /alerts similarly.

```
$ Json /alerts/92
{
    "handler" : "candela.lanforge.HttpEvents$FixedJsonResponder",
    "uri" : "alerts/:event_id",
    "alert" : {
        "name" : "wlan0",
        "time-stamp" : "2018-07-02 16:23:30.880",
        "entity id" : "NA",
        "id" : "92",
        "eid" : "1.1.5",
```

Streaming Events

Continually polling the **/events** URL is not as effective as streaming a websocket providing the same data. We need a web socket client. Websockets are built into modern browsers and there are python and perlutilities for the job as well. An easy to use python client is **wsdump**.

Installing wsdump

There is a useful python utility called wsdump (or wsdump.py). Try to install the python-websocket package to get it. There are many similar matches, but there is not one dedicated package that provides it. On Fedora:

```
root@fedora$ dnf whatprovides `which wsdump`
root@fedora$ dnf install -y python3-websocket-client
```

```
root@ubuntu$ ls -l /usr/bin/wsdump
/usr/bin/wsdump → /etc/alternatives/wsdump
root@ubuntu$ ls -l /etc/alternatives/wsdump
/etc/alternatives/wsdump → /usr/bin/python2-wsdump
root@ubuntu$ dpkg-query -S /usr/bin/python2-wsdump
python-websocket: /usr/bin/python2-wsdump
```

root@ubuntu\$ sudo apt install python-websocket

You might need to install pip, and that might be in the python3-pip package. Then you can install via:

```
$ sudo apt install python-pip # or sudo dnf install python-pip
$ sudo pip install --upgrade pip
$ pip search websocket
$ sudo pip install websocket-client
```

Streaming Using wsdump

Here's an example of wsdump below. Don't forget you are now using h the ws:// schema and not the http:// schema!

```
$ /usr/bin/wsdump ws://localhost:8081/
```

It might take a few second to start showing results if your system is not very active. You should be able to prompt output by executing this message in the **Messages** tab: **gossip hi ben**!



Streaming Using javascript

You can also use a web page to follow events because websockets are built into modern browsers. This is a screenshot of the

WebSocket Test
URL (ws://) ws://jed-F23:8081/ Connect Disconnected
<pre>{"wifi-event":"1.3: IFNAME=sta3120 <3>CTRL-EVENT-CONNECTED - Connection to 00:0e:8e:d5:fa:e6 completed [id=0 id_str=]"}</pre>
{=wifi-event=:=1.1: vap1000: del station 00:0e:8e:a1:7d:45=}
<pre>{"flags":0,"event_type":0,"event_id":400895,"eid_type":2,"shelf":1,"resource":3,"port":33,"endp":0,"extra":0,"pri sta3120 is Link DOWN.","name":"sta3120","eid":"1.3.33","event_eid":{"type":11,"event_id":400895,"index":- 1,"flags":0,"is_alert":true},"is_alert":true}</pre>
{"deleted-alert":400895}
<pre>{=flags::0,=event_type::0,=event_id=:400896,=eid_type=:2,=shelf=:1,=resource=:3,=port=:33,=endp=:0,=extra=:0,=pri sta3120 is Link DOWN.=,=name=:=sta3120=,=eid=:=1.3.33=,=event_eid=:{=type=:11,=event_id=:400896,=index=:- 1,=flags=:0,=is_alert=:true},=is_alert=:true}</pre>
<pre>{"flags":0,"event_type":0,"event_id":1023429,"eid_type":2,"shelf":1,"resource":3,"port":33,"endp":0,"extra":0,"pr sta3120 is Link DOWN.","name":"sta3120","eid":"1.3.33","event_eid":{"type":11,"event_id":1023429,"index":` 1,"flags":0,"is_alert":false},"is_alert":false}</pre>
{"wifi-event":"1.3: sta3120: del station 00:0e:8e:d5:fa:e6"}
<pre>{"wifi-event":"1.3: sta3120 (phy #1): deauth 00:0e:8e:a1:7d:45 -> 00:0e:8e:d5:fa:e6 reason 3: Deauthenticated because sending station is leaving (or has left) the IBSS or ESS=}</pre>
<pre>{"flags":0,"event_type":7,"event_id":1023430,"eid_type":2,"shelf":1,"resource":3,"port":33,"endp":0,"extra":0,"pr</pre>
<pre>{"wifi-event":"1.3: sta3120 (phy #1): disconnected (local request) reason: 3: Deauthenticated because sending station is leaving (or has left) the IBSS or ESS=}</pre>
<pre>{"flags":0,"event_type":50,"event_id":1023431,"eid_type":2,"shelf":1,"resource":3,"port":32,"endp":0,"extra":0,"p sta3119 IP changed from 0.0.0.0 to 10.41.13.111","name":"sta3119","eid":"1.3.32","event_eid": {"type":11,"event_id":1023431,"index":-1,"flags":0,"is_alert":false},"is_alert":false}</pre>
DISCONNECTED
III III III III III III III III III II

Data Views URLs

/shelf

The /shelf/1/ URL provides a list of resources in your realm:

```
$ Json /shelf/1
{
    "handler": "candela.lanforge.HttpResource$JsonResponse",
    "uri": "shelf/:shelf_id",
    "candela.lanforge.HttpResource": {
        "duration": "0"
    },
    "resources": [
        {
            "1.1": {
                "_links": "/resource/1/1",
                "hostname": "idtest.candelatech.com"
            }
        },
        {
            "1.2": {
                "_links": "/resource/1/2",
                "hostname": "hedtest"
            }
        }
   ]
```



/resource

The /resource URL provides a digest of ports available at the requested resource.

{		
"handler" : "cande	ela.lanforge.HttpResource\$JsonResponse",	
"resource" : {		
"free swap" : 5	26332,	
"free mem" : 46	34228,	
"load" : 0.4,		· · · ·
"bps-rx-3s" : 7	850,	
"sw version" :	" 5.3.8 64bit",	
"entity id" : "	'NA",	
"tx bytes" : 40	533976395,	
"phantom" : fal	.se,	
"eid" : "1.1",		
"hostname" : "i	dtest.candelatech.com",	
"hw version" :	"Linux/x86-64",	-
1		
<u> </u>		
	× +	
/resource		
/resource		
→ C' ŵ	(i) localhost:8080/resource/1/1/ 110%	
→ C' û	i localhost:8080/resource/1/1/	
→ C' û	i localhost:8080/resource/1/1/	
	() localhost:8080/resource/1/1/	
	i localhost:8080/resource/1/1/	
(←) → C ⁱ (resource)	i localhost:8080/resource/1/1/	
	i localhost:8080/resource/1/1/	
$\leftarrow \rightarrow \mathbb{C}$	i localhost:8080/resource/1/1/	
$\overleftarrow{\leftarrow} \rightarrow \mathbb{C} \widehat{\Box}$ $/resource$ Shelf: 1 Resource:	i localhost:8080/resource/1/1/	
$\overleftarrow{\leftarrow} \rightarrow \mathbb{C} \widehat{\Box}$ $/resource$ Shelf: 1 Resource: $\cdot \text{ bps-rx-3s: 885}$	i localhost:8080/resource/1/1/	
$\leftarrow \rightarrow \mathbb{C}$ $\widehat{\square}$ /resource Shelf: 1 Resource: • bps-rx-3s: 885 • bps-tx-3s: 100	i localhost:8080/resource/1/1/	
$\leftarrow \rightarrow \mathbb{C}$ $\widehat{\square}$ /resource Shelf: 1 Resource: • bps-rx-3s: 885 • bps-tx-3s: 100 • cli-port: 4003	i localhost:8080/resource/1/1/	
$\leftarrow \rightarrow \mathbb{C}$ $\widehat{\square}$ /resource Shelf: 1 Resource: • bps-rx-3s: 885 • bps-tx-3s: 100 • cli-port: 4003 • apui Intel(P)	i localhost:8080/resource/1/1/	

/port

The **/port** URL provides a digest of ports and their state. You can request multiple ports by ID on this resource by appending the port IDs with commas. You can list ports on a resource:

We can query multiple ports at a time by their number or their name by placing a comma between the specifiers. Additionally, we can query for just the fields we desire. All field names are lower-case: ? fields=tx+crr,rx+fifo.



igodolarrow $ ightarrow$ $igodolarrow$ $igodolarrow$	(i) localhost:8080/port/1/1/3,4/		
EID 4Way ANQP AP Activity Alias Beacon	Bytes RX Bytes TX CX CX LL LL CX Ago	Channel Collisions Connections Crypt	HCP ms) Device Down G ⁱ

/cx

The /cx URL allows us to query Layer-3 connection information.

```
$ Json /cx
{
    "uri" : "cx",
    "handler" : "candela.lanforge.GenericJsonResponder",
    "connections" : [
       "41.1" : {
            "entity id" : "NA",
            "name" : "udp:r3r2:3000",
            "_links" : "/cx/41"
        },
        "50.1" : {
            "name" : "udp:r3r2:3009",
            "entity id" : "NA",
            "links" : "/cx/50"
        }
```

And individual connections:

```
$ Json /cx/udp:r3r2:3000$ Json 'cx/udp:r3r2:3000'
{
    "uri" : "cx/:cx_id",
    "41.1" : {
        "drop pkts b" : 0,
        "type" : "LF/UDP",
        "rx drop % a" : 0,
        "rpt timer" : "1000",
        "pkt rx a" : 0,
        "avg rtt" : 0,
        "rx drop % b" : 0,
        "name" : "udp:r3r2:3000",
        "endpoints (a +> b)" : "udp:r3r2:3000-A <=> udp:r3r2:3000-B",
        "drop pkts a" : 0,
        "entity id" : "NA",
```

Technically, colons in URLs need to be encoded as **%3A**, so the above URL should be /cx/udp%3Ar3r2%3A3000, but curl is pretty darned forgiving.

/endp

Endpoints may be listed and inspected:

```
$ Json /endp/sta3000-ep-B
{
    "candela.lanforge.HttpEndp" : {
        "duration" : "1"
    },
    "uri" : "endp/:endp_id",
    "endpoint" : {
        "rx rate ll" : 0,
        "pdu/s tx" : 0,
        "bursty" : false,
        "rx rate" : 0,
        "tx pkts ll" : 0,
        "rx bytes" : 0,
        "run" : false,
        "tcp rtx" : 0,
```

Creating Ports

It is possible to create ports and connections by using the CLI commands. Your LANforge test scenarios (located in the /home/lanforge/DB/ directory) contain all the CLI commands that create your ports and connections. You can submit those commands over HTTP in two ways:

• /cli-json/\$command An example of using the *gossip* command:

```
curl -X POST -H 'Content-type: application/json' \
          -d '{"message":"hello world"}' http://localhost:8080/cli-json/gossip
```

Then check your LANforge GUI messages.

• /cli-form/\$command An example of using the *gossip* command:

curl -X POST -d 'message=hello+world' http://localhost:8080/cli/gossip

Then check your LANforge GUI messages.

• /cli/: use this method to submit a raw URL-encoded command. This might be useful if you are copying commands directly out of a database:

curl -X POST -d 'cmd=gossip hello' http://localhost:8080/cli/

Except for /cli-json, these methods accept application/x-www-form-urlencoded content type submissions. This is default for the NanoHttp library and default for curl.

These CLI commands do not return data, only a result code. All data that the Perl scripts would collect from command line queries is sent directly to the GUI. Some CLI commands send data over the websocket, like the **diag** command.

Command help

Commands are often complex and include a number of bitwise flags to set the state and features of ports. There is presently no tag-substitution for port flags, but there is a help utility that can help you compute them.

http://localhost:8080/help/

\leftrightarrow \rightarrow c $\hat{\omega}$	(i) localhost:8080/help/ ♥ ☆ Q Search >> =	
<u>add_arm_endp</u>	<u>CLI Reference for add_arm_endp</u>	
<u>add_bgp_peer</u>	<u>CLI Reference for add_bgp_peer</u>	
add_bond	CLI Reference for add_bond	=
add_br	<u>CLI Reference for add_br</u>	
add cd	CLI Reference for add_cd	
add cd endp	CLI Reference for add_cd_endp	
add_cd_vr	<u>CLI Reference for add_cd_vr</u>	
add_channel_group	CLI Reference for add_channel_group	
add cx	<u>CLI Reference for add_cx</u>	
add endp	CLI Reference for add endp	

Select a command to see the field helper screen:

```
http://localhost:8080/help/set port
```

Type values into the field inputs and the CLI command will be refreshed:

Command Composer [set_port]	
This is the curl command:	
\$ echo '' > /tmp/curl_data \$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/set_port	
This is the CLI command:	
1 3 sta3000 NA NA NA NA 2147483649 NA NA NA NA 16384 3 NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA	

Click the **Parse Command** button and the values in the command box will be displayed in the curl command and the field inputs. (Notice this form is doing a GET request.)

This is the curl command:

```
$ echo 'shelf=1&resource=3&port=sta3000&current_flags=2147483649&interest=16384&report_timer=3' > /tmp/curl_data
$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/set_port
```

This is the CLI command:

You may find a list of flag fields that are organized by field names. The text area below the selection list is the sum of the selected fields. Copy the flag values into the input field above to incorporate it into your command.

current.advert-flow-control current.auto-negotiate flags2.BYPASS_DISCONNECT flags2.BYPASS_ENABLED flags2.BYPASS_POWER_DOWN flags2.BYPASS_POWER_ON flags2.SUPPORTS_BYPASS flags2.USE_STP interest.AUX-MGT interest.Alias interest.BPLDCE	
interest.BYPASS interest.CPU MASK	Ξ
interest.DHCP	
interest.DHCP-RLS	
interest.DHCPv6	
Interest.GEN_OFFLOAD	
interest.IPV6_ADDRS	

Creating a WiFi Station

Please refer to the scripts **lf_associate_ap.pl** and **lf_vue_mod.sh** for examples of how to produce lists of CLI commands involved in creating stations. Please refer to:

- 1. Learn CLI Commands used to operate WiFi stations
- 2. and Changing Station WiFi SSID with the CLI API

These will provide ways of collecting the CLI commands in log files for you to place into the command /help/ page.

• Use ssh to log into your LANforge manager. Use the 1f_vue_mod.sh script to create a station:

```
$ cd scripts
$ ./lf_vue_mod.sh --mgr localhost --resource 3 --create_sta --name sta3101 \
--radio wiphy1 --ssid idtest-1000-open --passphrase '[BLANK]' \
--log_cli /tmp/clilog.txt
$ cat /tmp/clilog.txt
$ cat /tmp/clilog.txt
set_wifi_radio 1 3 wiphy1 NA -1 NA NA NA NA NA NA NA 0x1 NA
add_sta 1 3 wiphy1 sta3101 1024 idtest-1000-open NA [BLANK] AUTO NA 00:0e:8e:c1:df:45 8 NA NA NA
set_port 1 3 sta3101 0.0.0.0 255.255.0.0 0.0.0.0 NA 2147483648 00:0e:8e:c1:df:45 NA NA NA 84050[]
```

1. Enter each command into the your browser toolbar by altering the command into a url:

http://localhost:8080/help/**set_wifi_radio**?cli=1 3 wiphy1 NA -1 NA NA NA NA NA NA

Produces:

```
$ echo 'shelf=1&resource=3&radio=wiphy1&channel=-1&flags=0x1' > /tmp/curl_data
$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' \
http://localhost:8080/cli-form/set wifi radio
```



Produces:



Produces:

```
$ echo 'shelf=1&resource=3&port=sta3101&ip_addr=0.0.0.0&netmask=255.255.0.0&gatewa
$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' \
http://localhost:8080/cli-form/set_port
```

2. Verify with the LANforge GUI the changes you wish to make.

Creating Connections

Using the /cli-json/add_endp and /cli-json/add_cx URLs, it is possible to create Layer-3 connections. Create the Layer-3 endpoints first, of course.

Create L3 Endpoints

Construct your command using the /help/add_endp page. For an example, use these parameters:

alias	enter udp1000-A
shelf	1
resource	2
port	b2000
type	select type.lf_udp
min_rate	1000000 (1 Mbps)
max_rate	SAME
payload_pattern	select payload_pattern.increasing

Command Composer [add_endp]

This is the curl command:

\$ echo 'alias=udp1000-A&shelf=1&resource=2&port=b2000&type=lf_udp&min_rate=1000000&payload_pattern=increasing' > /tmp/curl_data \$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/add_endp

This is the CLI command:

udp1000-A 1 2 b2000 lf_udp NA NA 1000000 NA NA NA NA increasing NA NA NA NA

•

Click **Parse Command** and copy the resulting **cur1** command into a text editor:



And for the **B** endpoint, choose a station:

alias	enter udp1000-B
shelf	1
resource	7
port	sta7000
type	select type.lf_udp
min_rate	54000 (54 Kbps)
max_rate	SAME
payload_pattern	select payload_pattern.increasing

6	LANforge CLI Help - Mozilla Firefox	
LANforge CLI Help	X Querying the LANforge GUI for JSC X +	
↔ → ℃ ŵ	(i) localhost:8080/1 00% ··· ♥ ☆ Q Search 👱 IIN 🛃 🖸 🖸	=
	Command Composer [add_endp]	
This is the curl command:	:	
\$ echo 'alias=udp1000-B&she \$ curl -sqv -H 'Accept: app	elf=1&resource=7&port=sta7000&type=lf_udp&min_rate=54000&payload_pattern=increasing' > /tmp/curl_data wlication/json' -X POST -d '@/tmp/curl_data' http://atlas:80&Ocli-form/add_endp	
This is the CLI command:		
4		Þ

Click **Parse Command** and copy the resulting **cur1** command into a text editor:

	_					J. C. Harden and C.	
-	ø					(Untitled)	
1	File	Edit	Search	Options	Help	4	
	1 2 3 4 5 6 7 8	#!/bi echo curl echo curl	n/bash 'alias -sqv - 'alias -sqv -	=udp1000 H 'Accep =udp1000 H 'Accep	-A&shelf=14 t: applica -B&shelf=14 t: applica	wresource=2&port=b2000&type=lf_udp&min_rate=1000000&payload_pattern=increasing' > /tmp/curl_data ion/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/add_endp wresource=7&port=sta7000&type=lf_udp&min_rate=54000&payload_pattern=increasing' > /tmp/curl_data ion/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/add_endp	
4							۱.

We'll save this file as a shell script: ~/create-endp.sh We can then run it from our terminal like so: bash -x create-endp.sh

jreynolds@atlas:~/btbits/x64_btbits/html/i 💥 jreynolds@atlas:~	ireynolds@atlas:/mnt/d2/pub/system-images	jreynolds@fs1:/mnt/d1/pub/sysimage	💥 jreynolds@atlas:~/btbits/x64_btbits/client 🔺
\$ cd			
jreynolds@atlas:~;#			
* Trying 192.168.100.51			
* TCP_NODELAY set * Connected to atlas (192,168,100,51) port 8080 (#0)			
> POST /cli-form/add_endp HTTP/1.1			
> User-Agent: curl/7.55.1			
<pre>> Accept: application/json > Content-Length: 101</pre>			
> Content-Type: application/x-www-form-urlencoded			
* upload completely sent off: 101 out of 101 bytes			
< Content-Type: application/json			
< Date: Wed, 1 Aug 2018 00:54:53 GMT < Connection: keen-alive			-
4			EN

We should see the endpoints we've created in the LANforge GUI**Endps** tab:

\$			LANforge Ma	anager Version(5.3.8)						
<u>C</u> ontrol <u>R</u> eporting <u>T</u> ear-Off <u>I</u> nfo <u>P</u> lugins										
Stop All Restart Manager Refresh HELP										
Layer-4 Generic Test Mgr Test Group Resource Mgr Event Log Alerts Port Mgr VAP Stations Messages										
Status Layer-3 L3 Endps VolP/RTP VolP/RTP Endps WanLinks Attenuators RF-Generator File-IO									-10	
		_			1					
Min PDU Size AUTO	- 0	Go N	1ax PDU Size Same	-	Go	Start Stop	Oujesce Cle	ear		
MIN TV Bate New Hodem (56	6 Khps) 🗶 (-	60					
					Display	Croate Medif	Rotch Modify	Delete		
View 0-1000	- 0	50			Display		Batch Modily	Delere		
	. I	1		All Endpoints —				1	_	
Name	FID BUI	Mng	Script	Tx Bate	Tx Rate	Tx Bate (last)	Tx Bate II	By Bate	₹x	
					(1 min)			(1	
udp2.b2000-08.sta8261-B	1.2.8.440 📃	~	None	0	0	0	0	0		
udp2.b2000-08.sta8262-A	1.8.272 📃	~	None	6,082	4,908	0	0	62,573		
udp2.b2000-08.sta8262-B	1.2.8.446	~	None	112,799	92,467	0	0	6,055		
udp1000-A	1.2.8.596		None	0	0	0	0	0		
udp1000-B	1.7.10		None	0	0	0	0	47.100		
udp:r3r2:3000-A	1.3.13.28		None	55,806	56,155	54,602	60,802	47,109	-51	
udp:r3r2:3000-B	1 2 1 4 20		None	55,082	55,423	24,772	50,749	50,082		
									Þ	
Logged in to: idtest:4002 as:	Admin									
Logged In to. Idtest.4002 ds.	-sammer									

Create L3 Connection

With the creation of two endpoints, we can proceed with creating a Layer 3 cross-connect. This is much simpler, it really only takes the names of the two endpoints we created above. We'll choose default_tm for the test manager.

	G	۵	01	ocalhosta	90%		♥ ☆		Search		¥									
		Co	mm	nand	Со	mpo	oser	[ad	ld_c	x]										
This is the curl command:																				
5 echo '' > 5 curl -sav	> /tmp,	/curl_da Accept:	ta applicat	ion/ison'	-X POST	-d '@/tm	p/curl da	ita' http	o://atlas:	8080/c	:li-fo	rm/add	сх							
This is the	CLI	comman	nd:									\$ curl -sqv -H 'Accept: application/json' -X POSI -d '@/tmp/curl_data' http://atlas:8080/cli-form/add_cx								
Parse Comn	nand	nandwill	undate wi	hen vou cha	ince them:	1														
Parse Comm Fields for th 01: alias	nand e comr	nand will	update wi	hen you cha	inge them:															
Parse Comm Fields for th 01: alias 02: test_mgr	nand e comr	nand will	update wi	hen you cha	inge them:															
Parse Comm Fields for th 01: alias 02: test_mgr 03: tx_endp	nand e comr NA NA	nand will	update wl	hen you cha	nge them:															

alias	udp1000
test_mgr	default_tm
tx_endp	udp1000-A
rx_endp	udp1000-B

Click the **Parse Command** button and copy the resulting **cur1** command into your editor with the shell script. Run the script again. It doesn't hurt to re-create the endpoints.

				LANforge /	lanager Version	(5.3.8)					
<u>C</u> ontrol <u>R</u> eporting <u>T</u> ear-Off <u>I</u> nfo <u>P</u> lugins											
Stop All Restart Manager Refresh HELP											
Layer-4 Generic Test Mgr Test Group Resource Mgr Event Log Alerts Port Mgr VAP Stations Messages Status Layer-3 L3 Endps VolP/RTP VolP/RTP Endps WanLinks Attenuators RF-Generator File-I0											
Rpt Tin	Rpt Timer: fast (1 s) 🔻 Go Test Manager all 💌 Select All Start Stop Quiesce Clear										
View	0 - 500			▼ Go			Display	Cr <u>e</u> ate	Mo <u>d</u> ify D	ele <u>t</u> e	
			Cr	oss Connec	ts for Select	ed Test Mai	nager ——				
Name	Туре	State	Pkt Rx A	Pkt Rx B	Bps Rx A	Bps Rx B	Rx Drop % A	Rx Drop % B	Drop Pkts A	Drop Pkts B	Avg RTT
udp2.b2000-08.s	LF/UDP	Run	2,250,983	211,913	64,536	6,075	42.966	0.357	1,695,749	759	1,01 🔺
udp2.b2000-08.s	LF/UDP	Stopped	0	0	0	0	0	0	0	0	
udp2.b2000-08.s	LF/UDP	Stopped	1,576,746	152,583	62,573	6,055	44.527	0.452	1,265,599	693	67
udp1000	LF/UDP	Stopped	0	0	0	0	0	0	0	0	
udp:r3r2:3000	LF/UDP	Stopped	68	74	50,623	55,090	8.108	0	6	0	1,74
udp:r3r2:3001	LF/UDP	Stopped	40	77	28,802	55,445	48.052	0	37	0	1,71
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		IStoppod	62	77	15 202	55.460	10102	^	14		<u> </u>
Logged in to: idtest	:4002 as:	Admin									

Toggling the Connection

Cross connects have three good state: STOPPED, RUNNING, and QUIESCE. The command to change them is set_cx_state. You will have no trouble creating the command:

test_mgr	default_tm
cx_name	udp1000
cx_state	RUNNING

er [set_cx_state]
data' http://atlas:8080/cli-form/set_cx_state
will be computed when you select them, but you might need to lues into some fields (when you see token values like [string] or
or signed 64 bit values, so not all flags can be calculated as positive unsigned integers. If you

Click **Parse Command** and then you can paste the resulting command into your editor.

Command Composer [set_cx_state]											
This is the curl command:											
\$ echo 'test_mgr=default_tm&cx_name=udp1000&cx_state=RUNNING' > /tmp/curl_data \$ curl -sqv -H 'Accept: application/json' -X POST -d '@/tmp/curl_data' http://atlas:8080/cli-form/set_cx_state											
This is the CLI command:											
default_tm udp1000 RUNNING Parse Command	default_tm udp1000 RUNNING										
Fields for the command will update when you Fichange them:	lag Fields for command will be computed when you select them, but you might need to ctually write modified values into some fields (when you see token values like [string] or name]).										
01: test_mgr default_tm	x_state_DELETED x_state_QUIESCE										
02: cx_name udp1000	x_state.STOPPED x_state.STOPPED x_state.SWITCH										
03: cx_state RUNNING	te following numbers are only valid for signed 64 bit values, so not all flags can be calculated as positive unsigned integers. If you e a negative number, first check that the Java flag was not entered as an int.										

Advanced Techniques

You can make JSON submissions and you can also submit Base64 encoded values in both form an and JSON submission URLs.

Submitting Base64

Field names that end in -64 are interpreted as base64 encoded values. From a linux terminal, you can convert text to base64 encoded value using the **base64** command:

\$ echo "RUNNING" | base 64
UlVOTklORwo=

Below is a CLI command example. You typically would not care to spend the effort doing this unless the data you need to express is difficult to URL encode.

\$ echo 'test_mgr-64=YW55Cg==&cx_name-64=dWRwMTAwMAo=&cx_state-64=U1VOTklORwo=' > /tmp/curl_data
\$ curl -A 'Accept: application/json' -X POST -d @/tmp/curl_data http://host/cli-form?set_cx_state

Submitting JSON

Instead of posting to /cli-form, you can post to /cli-json and your submission will be parsed as a json object. The parameter names stay the same. The base64 name extensions are also available! You **need** to specify that your **Content-type** in the POST is **application**/json.

```
$ echo '{"test_mgr":"default_tm","cx_name":"udp1000","cx_state":"RUNNING"}' > /tmp/curl_data
$ curl -sq -H 'Content-type: application-json' -H 'Accept: application/json' \
    -X POST -d@/tmp/curl_data http://localhost:8080/cli-json/set_cx_state
```

Handling Mismatched Column Errors

(*This should be fixed as of 2018/08/14*) When the LANforge cliet is in GUI mode, the **columns** of data that are returned match the GUI **table columns** displayed. You can use the *Right-click* \rightarrow *Add/Remove Table Columns* menu item to change this. **We do not recommend doing this** for querying JSON data though, because the table columns definitions will not match up to the data the webserver expects to return.

<u></u>	2 LANforge Manager Version(5.3.8)												
<u>C</u> ontrol <u>R</u> eporting <u>T</u> ear-Off <u>I</u> nfo <u>P</u> lugins													
								9	Stop All	Rest	art Manage	er	F
Generic Status	Test	t Mgr Layer	Test -3	Group R L3 Endps	esource Mg VolP	r Eve /RTP	ent Log 🛛 A VolP/F	lerts RTP E	Port Mgr M Endps W	vAP Sta /anLink	ations Me s At	essages tenuators	Fil
Disp	p: 192.	168.1	00.51:0	0.0	Sniff Packet	s		î	Clear Counte	ers	Reset Poi	t Dele	<u>t</u> e
Rpt	Timer:	mediu	1m (8	s) 🔻	Apply		VRF	î	⊻iew Detail	s	Cr <u>e</u> ate	Mo <u>d</u> i	ify
					A	ll Ether	net Interfac	es (I	Ports) for all Re	source	s. ———		
Port	Pha	Down	Parent Dev	Channel	Device		SSID		AP		IP	Activity	
1.1.00					eth0					192.1	68.100.41	0	
1.1.01					ethl					0.0.0.	0	0	
1.1.02		~		36	wiphy0					0.0.0.	0	0	
1.1.03		V		44	wiphyl					0.0.0.	0	0	
1.1.04				48	wiphy2					0.0.0.	0	0	
1.1.05		V.	wiphy0	36	wlan0			N	ot-Associated	0.0.0.	0	0	
								1			-		

\$	curl -sq http://localhost:8080/port json_pp	
{		
	"error_list" : [
	"names_to_col_ids map is not going to work:\n[tx abort][25 > 4]\n[cx time (us)][48 > 4]\:	n
],	
	"status" : "INTERNAL_ERROR"	
}		
4		•

The terminal you started the LANforge client on will also give a similar error:

```
1532480073953: names_to_col_ids size:71
java.lang.IllegalArgumentException: names_to_col_ids map is not going to work:
1532480073953: lfj table columns:10
```

Reset the Table Layout

1. Right-clicking the Port Mgr and selecting Add/Remove Table Columns will allow you to change this.

lerts P RTP Endps	ort Mgr V s Wa	AP St anLinl	ations ks	Me At	ssaq	ges ators	F	
1 CI	ear Counter	rs	Res	et Por	t	De	le <u>t</u> e	
I	⊻iew Details		Cr <u>e</u> ate			Mo	Mo <u>d</u> ify	
es (Ports	;) for all Res	ource	es. —					
	AP		IP		A	ctivity		
		192.1	168.10	0.41	0			
		0.0.0	.0		0		_	
		0.0.0	0		0		_	
		0.0.0	0		0		_	
Not-4 C Not-4 P en F a2 C a2 C C Not-4 C C Not-4 C Not-4 C	Clear Select Aodify Select Reset Select Display Select Display Select Display Select Display Select Count Select Count Select Calculations Create Layer	ed ted ted (L cted (L cted t ted	.ogout t)		D		
Not-4	dd/Remove	Tabl	e Colu	mns			-	
en /	Add/ <u>R</u> emove	fabl	e Repo	ort Coli	umns	5		
a2	<u>s</u> ave Table L	.ayou	t					
a2 F	a2 Reset Table Layout						_	
· 4	auto-Resi <u>z</u> e	colun	nns					

2. Clicking the **Select All/None** button and then **Apply** will get all the columns displayed, and returned in your queries.

_	🛃 Add or Remove Table Columns 🔲 🗵						
🖌 Port	🖌 Phantom	🖌 Down	🖌 Parent Dev				
🗹 Channel	🖌 Device	🗾 SSID	🖌 AP				
🗾 IP	🖌 Activity	🗾 SEC	🗾 Alias				
🗾 RX Bytes	🗾 RX Pkts	🗹 Pps RX	🗾 bps RX				
🗾 TX Bytes	🗹 TX Pkts	🗾 Pps TX	🗾 bps TX				
🗹 Collisions	🗾 RX Errors	🗹 TX Errors	🗹 RX Drop				
🗾 RX Length	🖌 RX Over	RX CRC	🗾 RX Frame				
🗾 RX Fifo	🗾 RX Miss	🗾 TX Abort	🗹 TX Crr				
🗾 TX Fifo	🗾 TX HB	🗾 TX Wind	🗾 bps TX LL				
🗾 Bytes TX LL	🗾 bps RX LL	🗾 Bytes RX LL	🗾 Reset				
🗹 TX-Rate	🗾 RX-Rate	🖌 Status	🗾 Signal				
🗾 Noise	🗹 Connections	🗾 DHCP (ms)	🖌 CX Ago				
🗾 No CX (us)	🗹 CX Time (us)	🖌 ANQP Time (us)	🗹 4Way Time (us)				
🗹 Crypt	🗾 Retry	🗾 Misc	🗾 Beacon				
🗹 Key/Phrase	🗾 Login-OK	🗾 Login-Fail	🖌 Logout-OK				
🗹 Logout-Fail	🗾 QLEN	MTU	🖌 Mask				
🗹 Gateway IP	MAC	🗹 IPv6 Address	🗹 IPv6 Gateway				
	Select None N	Apply	<u>C</u> ancel				

3. Make sure to **Right-Click** \rightarrow **Save Table Layout** so that your next session will show all the data.

s	s Port Mgr VAP Stations Messages Armageddon WanLinks Attenuato										
t	C	lear C	ounters	Reset Port		De	Dele <u>t</u> e				
ĩ	I <u>V</u> iew Details			Cr <u>e</u> ate		Mo	Mo <u>d</u> ify				
s (Ports) for all Resources.											
P	Parent Dev RX Bytes		RX Pkts	Pps RX			b				
1			610.611	669.028		2.53	36	20,			
W W W	iphyi iphy: iphy: iphy: iphy:	Clear Modi Rese Displ • Dy Table Cour <u>C</u> alcu View Crea	Selected fy Selected t Selected t Selected (ay Selected namic Repoi e Report t Selected ulations Logs te Layer-3 C	Logout) rt X		D	8846000590006	16,			
w	Add/Remove Table Columns Add/Remove Table Report Columns Save Table Layout wiphy Reset Table Layout wiphy Auto-Resize Columns wiphy 0 0 0					S	0824000				
0'281'\1''' 2'282'0\8							59	-			

4. Restart the LANforge client

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