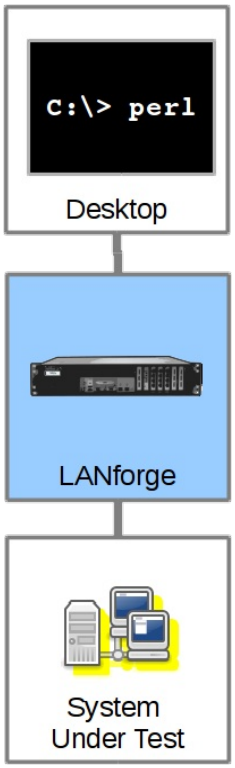
http://www.candelatech.com
sales@candelatech.com
+1 360 380 1618 [PST, GMT -8]

Querying the LANforge GUI for JSON Data

Goal: Learn how to configure and query the LANforge GUI for JSON formatted data.



The LANforge GUI (as of release 5.3.6) can be configured to start a lightweight web service listener that can provide data about ports and layer-3 connections. This service can be queried with with any browser or AJAX connection. This feature promises to provide a number of 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, can be represented in JSON format.
- GUI client synthesized data: many columns in the GUI tabs are synthesized during traffic generation and are not available through the CLI scripting API.
- Reduced effort when integrating with third party test libraries: many other testing libraries expect JSON formatted input.

Present and potential drawbacks of the GUI JSON data feature:

- Introductory state: the JSON views/schema of the objects is at a demonstation state. It is likely to change with the next release of LANforge, for example, the JSON objects in 5.3.7 probaby won't match 5.3.6.
- JSON Features are compiled into the LANforge GUI from Java sources. Adding or extending JSON features is not as simple as adding another Groovy script. We're concerned that Groovy plugins will not perform as well. *If you are interested in Groovy plugins that provide JSON data, please discuss your needs with us.*
- Initially limited data views: in 5.3.6, we are providing this as a proof of concept and have chosen to display only data from the Port Manager, Layer-3 Connections, and Layer-3 Endpoints tabs. As requirements develop, we'll develop more data views.

GUI 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:

Starting the GUI with web socket on port 8080.

```
$ cd /home/lanforge
$  ./lfclient.bash -httpd
```

You can specify the port to listen on:

Starting the GUI with web socket on port 3210.

```
$ cd /home/lanforge
$  ./lfclient.bash -httpd 3210
```

There is not a graphical gui preference to set for the feature at this time.

Example Query

Status

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

```
 $ curl -sqv http://localhost:8080/
*   Trying ::1...
* Connected to localhost (::1) port 8080 (#0)
> GET / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Tue, 2 May 2017 21:22:57 GMT
< Connection: keep-alive
< Content-Length: 461
<
* Connection #0 to host localhost left intact
"text":"These urls are presently available",
"mappings":{"":"(This page) Provides status and configuration.","/Port/Index":"List of ports
```

Parsing Results

JSON formatted text is pretty difficult to read, so you might like to know two different utilities that can help you look at it: `jq` and `jsonlint`.

Installing and using `jq`

On Fedora, install `jq`:

```
 $ sudo dnf install -y jq
```

On Ubuntu, install `jq`:

```
$ sudo apt install -y jq
```

Now we can perform a query:

```
$ curl -sq http://localhost:8080/ | jq
[
  {
    "handler": "candela.lanforge.HttpStatus"
  },
  {
    "uri": ""
  },
  {
    "text": "These urls are presently available"
  },
  {
    "mappings": {
      "": "(This page) Provides status and configuration.",
      "/Port/Index": "List of ports in the PortManager tab.",
      "/PortTab": "List of ports in the PortManager tab.",
      "/Layer3/Calc": "Show Layer3 Endpoint calculations",
      "/Port/Calc": "Show port calculations.",
      "/Layer3/Index": "List of ports in the Layer3 tab",
      "/": "(This page) Provides status and configuration."
    }
  }
]
```

Validating with jsonlint

On Fedora, install `php-jsonlint`:

```
$ sudo dnf install -y php-jsonlint
```

On Ubuntu, install `jsonlint`:

```
$ sudo apt install -y jsonlint
```

Running `jsonlint-php` is not nearly as exciting:

```
$ curl -sq http://localhost:8080/ | jsonlint-php
Valid JSON
```

Data Views

Ports

/Port/Index

Provides a digest of ports and their state:

```
$ curl -sq http://localhost:8080/Port/Index
[{"handler":"candela.lanforge.HttpPortIndex"},{"uri":"Port/Index"},
{"header":["eid","name","phantom","down"]},
```

```
{"data":[
          ["1.0.0","eth0","false","false"],
          ["1.1.0","eth0","false","false"],
          ["1.2.0","eth0","false","false"],
          ["1.1.1","eth1","false","false"],
          ["1.2.1","eth1","false","false"],
          ["1.1.2","wiphy0","false","false"],
          ["1.2.2","wiphy0","false","false"],
          ["1.1.3","wiphy1","false","false"],
          ["1.2.3","wiphy1","false","false"],
          ["1.1.4","wiphy2","false","false"],
          ["1.2.4","wiphy2","false","false"],
          ["1.1.5","wlan0","false","true"],
          ["1.2.5","wlan0","false","false"],
          ["1.1.6","wlan1","false","true"],
          ["1.2.6","wlan1","false","false"],
          ["1.1.7","wlan2","false","true"],
          ["1.2.7","wlan2","false","false"],
          ["1.1.8","vap100","false","false"],
          ["1.1.9","vap110","false","false"],
          ["1.1.10","vap120","false","false"],
          ["1.1.11","b100","false","false"],
          ["1.1.12","r100a","false","false"],
          ["1.1.13","r100b","false","false"],
          ["1.1.14","r101a","false","false"],
          ["1.1.15","r101b","false","false"]
]}]
```


/PortTab

PortTab provides data found in Port Mgr tab. The configuration of the Port Mgr tab (Add-Remove Table Columns) will show in this response.

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


```
[{"handler":"candela.lanforge.HttpPortTab"},
{"uri":"PortTab"},
{"header":["Port","Phantom","Down","Parent Dev","Device","IP","RX Bytes","AP","Connections","SS
{"data":[["1.1.00","false","false","","eth0","192.168.100.26","4,097,857,554","","0","","192.16
  ["1.1.01","false","false","","eth1","10.26.1.1","1,651,636,119,416","","0","","0.0.0.0","00:9
  ["1.1.02","false","false","","wiphy0","0.0.0.0","3,217,297,499","","0","","0.0.0.0","00:0e:8e
  ["1.1.03","false","false","","wiphy1","0.0.0.0","4,462,247,651","","0","","0.0.0.0","00:0e:8e
  ["1.1.04","false","false","","wiphy2","0.0.0.0","4,294,994,299","","0","","0.0.0.0","00:0e:8e
  ["1.1.05","false","true","wiphy0","wlan0","0.0.0.0","0","Not-Associated","0","","0.0.0.0","00
  ["1.1.06","false","true","wiphy1","wlan1","0.0.0.0","0","Not-Associated","0","","0.0.0.0","00
  ["1.1.11","false","false","","b100","10.26.4.1","9,128,638,764","","0","","0.0.0.0","00:0e:8e
  ["1.1.12","false","false","r100b","r100a","0.0.0.0","850,541,375,484","","0","","0.0.0.0","a6
  ["1.1.13","false","false","r100a","r100b","10.26.4.13","845,854,312,922","","0","","10.26.4.1
  ["1.1.14","false","false","r101b","r101a","0.0.0.0","845,854,004,368","","0","","0.0.0.0","62
  ["1.1.15","false","false","r101a","r101b","10.26.4.14","850,541,680,988","","0","","10.26.4.1
```

```
  ["1.2.0","false","false","","eth0","192.168.100.42","1,285,507,529","","0","","192.168.100.1"
  ["1.2.1","false","false","","eth1","10.26.1.2","11,741,736,214","","0","","10.26.1.1","00:90:
  ["1.2.2","false","false","","wiphy0","0.0.0.0","656,227,422,653","","0","","0.0.0.0","00:0e:8
  ["1.2.3","false","false","","wiphy1","0.0.0.0","591,417,884,328","","0","","0.0.0.0","00:0e:8
  ["1.2.4","false","false","","wiphy2","0.0.0.0","564,127,688,199","","0","","0.0.0.0","00:0e:8
  ["1.1.07","false","true","wiphy2","wlan2","0.0.0.0","0","Not-Associated","0","","0.0.0.0","00
  ["1.1.08","false","false","wiphy0","vap100","0.0.0.0","3,071,654,503","","0","jedtest-40","0.
  ["1.1.09","false","false","wiphy1","vap110","0.0.0.0","3,071,656,850","","0","jedtest-36","0.
  ["1.1.10","false","false","wiphy2","vap120","0.0.0.0","3,071,657,075","","0","jedtest-44","0.
  ["1.2.5","false","false","wiphy0","wlan0","10.26.4.12","548,247,399,423","00:0E:8E:EF:49:56",
  ["1.2.6","false","false","wiphy1","wlan1","10.26.4.11","548,246,490,732","00:0E:8E:D1:B1:91",
  ["1.2.7","false","false","wiphy2","wlan2","10.26.4.10","548,245,803,601","00:0E:8E:C9:6F:5B",
]}]
```

/Port/Calc

The path provides port calculations similar to the Port Calculations window. It requires one or more EID parameters. You can specify the eids as a comma separated list: `?eid=1.2.3,1.2.4` or a series of query parameters: `?eid=1.2.3&eid=1.2.4`

```
curl -sq "http://localhost:8080/Port/Calc?eid=1.1.15"
[{"handler":"candela.lanforge.HttpPortTabSsData"},
{"uri":"Port/Calc"},
{"1.1.15":
  {"TimeStamp":"1493767726537",
  "Name":"r101b",
  "EID":"1.1.15",
  "Resource":"jedtest",
  "tx_pkts":"48631077",
  "rx_pkts":"48697662",
  "tx_bytes":"845854005290",
  "rx_bytes":"850541798136",
  "tx_errors":"0",
  "rx_errors":"0",
  "tx_dropped":"0",
  "rx_dropped_pkts":"0",
  "rx_multicast":"0",
  "rx_collisions":"0",
  "rx_length":"0",
  "rx_overrun":"0",
  "rx_crc":"0",
  "rx_frame":"0",
  "rx_fifo":"0",
  "rx_missed":"0",
  "tx_aborted":"0",
  "tx_carrier":"0",
  "tx_fifo":"0",
  "tx_heartbeat":"0",
  "tx_window":"0",
```

```
  "rx_signal":"0",
  "link_speed":"10000000000",
  "rx_link_speed":"10000000000",
  "frequency":"0",
  "portal_login_ok":"0",
  "portal_login_fail":"0",
  "portal_logout_ok":"0",
  "portal_logout_fail":"0",
  "dhcp_neg_ms":"1699",
  "conn_count":"0",
  "last_conn_ms":"0",
  "connect_duration_us":"0",
  "disconn_duration_us":"0",
  "anqp_duration_us":"0",
  "4way_duration_us":"01493767729321"}
}]
```


Connections

Layer-3 information is organized in a similar way.

/Layer3/Index

This path provides a digest of layer 3 connections and their status.

```
 $ curl -sq 'http://localhost:8080/Layer3/Index'
[{"handler":"candela.lanforge.HttpLayer3Index"},
{"uri":"Layer3/Index"},
{"header":["Eid","Name","State","EidA","EidB"]},
{"data":[
  ["2.1","u-e1-w0","Stopped","1.2.5.2","1.2.1.1"],
  ["3.1","u-e1-w1","Stopped","1.2.6.4","1.2.1.3"],
  ["4.1","u-e1-w2","Stopped","1.2.7.6","1.2.1.5"],
  ["7.1","test-rr-10G","Stopped","1.1.15.12","1.1.13.11"]
]}]
```

/Layer3/Calc

This path provides data similar to the Layer-3 Calculations window. Notice that the pattern of the EID for connections is ordered as `[CX_INDEX].[TYPE]`, unlike ports that are identified like `[SHELF].[RESOURCE].[PORT_INDEX]` This `[TYPE]` identifier should correspond to the end of the EID for endpoints, which are `[SHELF].[RESOURCE].[PORT_INDEX].[ENDPOINT_INDEX].[TYPE]`. The following example queries endpoint data for two connections, 2.1 and 4.1:


```
$ curl -sq "http://localhost:8080/Layer3/Calc?eid=2.1,4.1"
[{"handler":"candela.lanforge.HttpLayer3TabSsData"},
{"uri":"Layer3/Calc"},
{"header":[
  "TimeStamp","Name","EID","CX-Name","IS_RUNNING","tx_rate","bps_tx_rate_3s","rx_rate","bps_rx_
```

```
]},
{
  "2.1":{
    "tx_data":[
      "1493770429164","u-e1-w0-A","1.2.1.1.1","u-e1-w0","false","9990794","10010814","52973","5
    ],
    "rx_data":[
      "1493770429164","u-e1-w0-B","1.2.5.2.1","u-e1-w0","false","52973","53077","9985497","1000
    ]
  },
  {
    "4.1":{
      "tx_data":[
        "1493748925591","u-e1-w2-A","1.2.1.5.1","u-e1-w2","false","9999315","10008210","55994",
      ],
      "rx_data":[
        "1493748925591","u-e1-w2-B","1.2.7.6.1","u-e1-w2","false","55999","59453","9999263","10
      ]
    }}]
```