Finding LANforge Report Data

**Goal:** Properly configured, the LANforge server or the LANforge GUI can collect connection performance information in CSV format.

By default, your LANforge server and your LANforge client do not save the data on connection and port performance. When you configure the save destination for this data, you can use it with any other tool that can read a CSV file.

Finding LANforge Report Data

**Select your Save Location**

You can tell the LANforge server to save data to a directory locally on the management machine, and you can configure your workstation running the the LANforge GUI to save data to a local desktop folder. First, find the Reporting Manager dialog by in the Reporting menu, and select Report Manager the client.
GUI Data Collection (Desktop Folder)

Collecting data on your local workstation is very convenient if you can leave the GUI running for the duration of your test scenario. The format of the data here should be similar to the format of the data saved to the server directory. The folders for collecting data are relative to the folder you start your GUI from. If you type in `1f_data` that probably means `C:\Users\mumble\AppData\Local\LANforge-GUI\1f_data`. You probably want to put in a fully qualified path that’s more intuitive, like `C:\Users\mumble\Documents\1f_data`.

**Generate Report**

The Report Generator uses the local data files. In that dialog shows the Report Input Directory field is a local folder where the CSV files collect. The Save Reports to Directory field is where HTML and PDF files should collect.

**Server Data Collection (Server Directory)**

If your test scenario runs longer than your GUI can be up, you can configure the LANforge server to collect the data. The directory is relative to the `/home/lanforge` directory, so if you enter `1f_data`, you would find the CSV files in `/home/lanforge/1f_data`. 
You can take a look at the data files easily. Here is a server data collection directory:

```bash
TamForgeObjectTest ~/lf_data
```

```
total 2628
-rw-r--r-- 1 root root  31465 Mar 16:52 wlan2_1.1.7_1488414451.csv
-rw-r--r-- 1 root root  31465 Mar 16:52 wlan1_1.1.6_1488414451.csv
-rw-r--r-- 1 root root  31465 Mar 16:52 wlan0_1.1.5_1488414451.csv
-rw-r--r-- 1 root root  90389 Mar 16:52 wiphy2_1.1.4_1488414451.csv
-rw-r--r-- 1 root root 125299 Mar 16:52 wiphy1_1.1.3_1488414451.csv
-rw-r--r-- 1 root root 101801 Mar 16:52 wiphy0_1.1.2_1488414451.csv
-rw-r--r-- 1 root root 138049 Mar 16:52 udp200-B_1488414451.csv
-rw-r--r-- 1 root root 137626 Mar 16:52 udp200-A_1488414451.csv
-rw-r--r-- 1 root root 160328 Mar 16:52 tcp200-B_1488414451.csv
-rw-r--r-- 1 root root 153351 Mar 16:52 tcp200-A_1488414451.csv
-rw-r--r-- 1 root root 26376 Mar 16:52 resource_testcandelatechcom_1.1_1488414451.csv
-rw-r--r-- 1 root root 114505 Mar 16:52 eth1_1.1.1_1488414451.csv
-rw-r--r-- 1 root root  49337 Mar 16:52 eth0_1.1.1_0_1488414451.csv
-rw-r--r-- 1 root root 168161 Mar 16:52 c201-B_1488414451.csv
-rw-r--r-- 1 root root 169329 Mar 16:52 c201-A_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar 16:52 wlan2_1.2.7_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar 16:52 wlan1_1.2.6_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar 16:52 wlan0_1.2.5_1488414451.csv
-rw-r--r-- 1 root root  38629 Mar 16:52 wiphy2_1.1.4_1488414451.csv
-rw-r--r-- 1 root root  38629 Mar 16:52 wiphy1_1.1.3_1488414451.csv
-rw-r--r-- 1 root root  38629 Mar 16:52 wiphy0_1.1.2_3_1488414451.csv
-rw-r--r-- 1 root root 114613 Mar 16:52 wiphy0_1.1.2_3_1488414451.csv
-rw-r--r-- 1 root root  32724 Mar 16:52 vadv0_1.1.8_1488414451.csv
-rw-r--r-- 1 root root  40057 Mar 16:52 st405_1.2.10_1488414451.csv
-rw-r--r-- 1 root root  48097 Mar 16:52 st2000_1.2.8_1488414451.csv
-rw-r--r-- 1 root root  39299 Mar 16:52 st2000_1.2.9_1488414451.csv
-rw-r--r-- 1 root root  21691 Mar 16:52 resource_testcandelatechcom_1.1_1488414451.csv
-rw-r--r-- 1 root root 105065 Mar 16:52 eth1_1.2.1_1488414451.csv
-rw-r--r-- 1 root root  40789 Mar 16:52 eth0_1.2.0_1488414451.csv
```

And using a utility like `notepad`, `vi`, `more` or `less` you can look at the file contents:

```
<table>
<thead>
<tr>
<th>TimeStamp</th>
<th>Name</th>
<th>EID</th>
<th>Cx-Name</th>
<th>IS_RUNNING</th>
<th>tx_rate</th>
<th>bps_tx_rate_3s</th>
<th>rx_rate</th>
<th>bps_rx_rate_3s</th>
<th>rx_droppkts1000</th>
<th>tx_pkts</th>
<th>rx_pkts</th>
<th>tx_bytes</th>
<th>rx_bytes</th>
<th>rx_dropped_pkts</th>
<th>rx_dup_pkts</th>
<th>rx_coo_pkts</th>
<th>rx_wrong_dev</th>
<th>rx_crc_failed</th>
<th>rx_bit_errors</th>
<th>tcp_Rx</th>
<th>conn_timedout</th>
<th>conn_estab</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Using Libre Office**
Importing the file into a spreadsheet like LibreOffice Calc is simple:

You only need to separate on comma (,)

Fields

The timestamp column
Libre Office does not have a built-in formula to do this, but it has been discussed here. And the solution is a formula that looks like this:

\[(A2/86400)+25569\]

and then you format the column as Date.

**Scripting with Bash**

There are a number of ways to collect and sort the data with shell utilities. The first utility to consider is `cut`, then `awk`. The first column of the endpoint file we are going to read is the timestamp, the 14th is the rx bytes.

**Reading the Data and RX Bytes**

**Converting Unix Date**

```
$ head -n2 c201-A_1488414451.csv | cut -d, -f1
Time Stamp
1488414454125
$ date -d @1488414454125
Mon Dec 23 19:28:45 PST 49135
```

**Using bash**

```
$ head -n2 c201-A_1488414451.csv | (while IFS=, read -a L; do echo ${L[13]}; done)
rx_bytes
33847640064
```

**Using cut**

```
$ head -n2 c201-A_1488414451.csv | cut -d, -f14
rx_bytes
33847640064
```

**Using awk**

```
$ head -n2 c201-A_1488414451.csv | awk -F, '{print $14}'
rx_bytes
33847640064

head -n2 c201-A_1488414451.csv | awk -F, '{print $1 "\t" $14}'
Timestamp rx_bytes
1488414454125 33847640064
```

**Scripting with Perl**

It is a lot easier to do math with a perl script than a bash or an awk script. You can pipe things into perl or perl will read the last argument of the `-ne` switch as an input file.

```
#!/usr/bin/perl

my $tt=0;
```

```
$ head -n2 c201-A_1488414451.csv \
   perl -ne '@v=split(/,/,$_); print "$v[0]\t$v[13]\n";' \
Time Stamp
1488414454125 33847640064
```

```
perl -ne 'BEGIN{$tt=0;@tstamps=();@rxb=();} \
   @{v=split(/,/,$_); push(@tstamps,$v[0]); push(@rxb,$v[13]);} \nEND{$dt=tstamps[#tstamps] - $tstamps[1]; $db=rxb[$#rxb] - $rxb[1]; \n   print "Time: $dt, Total: $db\n";}' \
c201-A_1488414451.csv
Time: 18959363, Total:1213399040
```

Not everything you do in perl is going to be a one-liner. Here’s an example of the same script as more properly formatted perl file:

```
#!/usr/bin/perl
my $tt=0;
```
my @tstamps=();
my @rxb=();
while(<>) {
    @v = split(/,/,$_);
    push(@tstamps, $v[0]);
    push(@rxb, $v[13]);
}
$dt = $tstamps[$#tstamps] - $tstamps[1];
$db = $rxb[$#rxb] - $rxb[1];
print "Time: $dt, Total:$db\n";