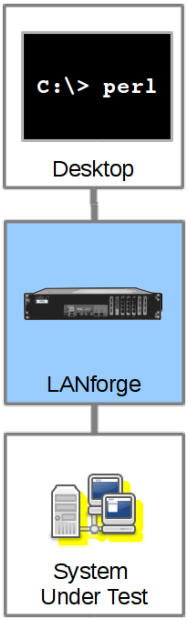
sales@candelatech.com
support@candelatech.com
+1 (360) 380-1618 [PST, GMT -8]

Finding LANforge Report Data

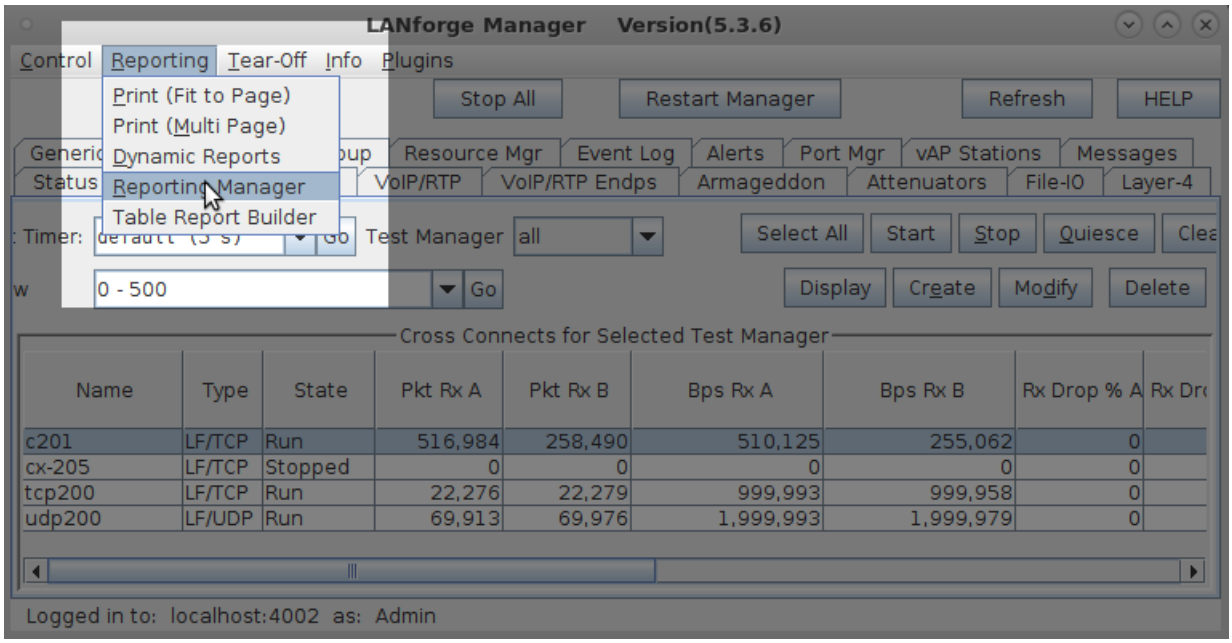
Goal: Properly configured, the LANforge server or the LANforge GUI can collect connection performance information in CSV format.
By default, your LANforge server and your LANforge client do not save the data on connection and port performance. When you configure the save destination for this data, you can use it with any other tool that can read a CSV file.



Finding LANforge Report Data

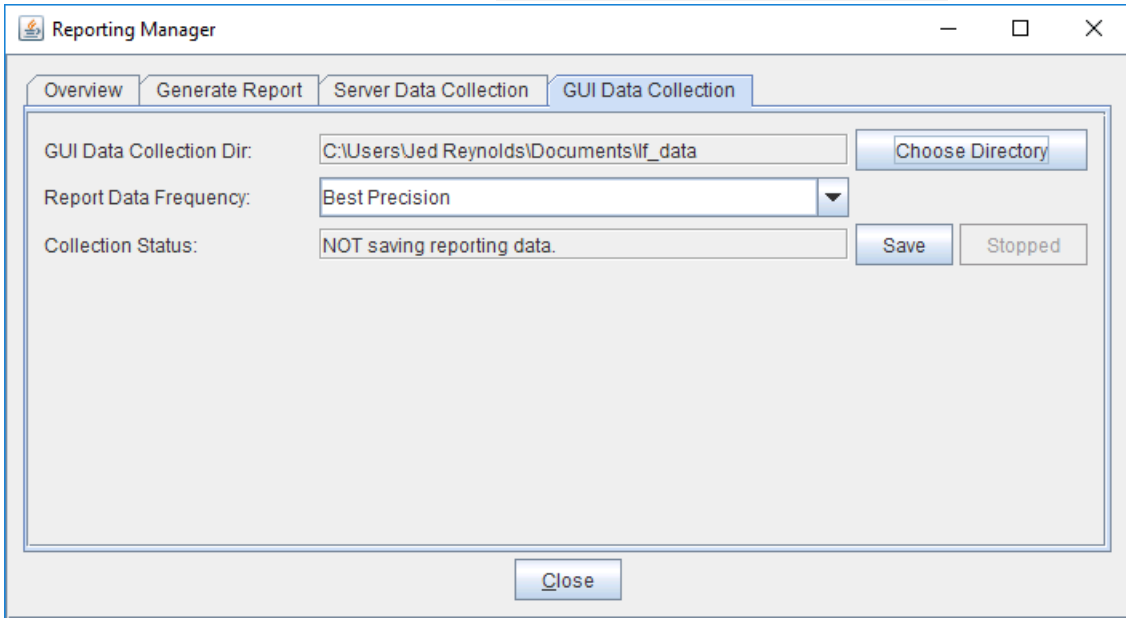
Select your Save Location

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.



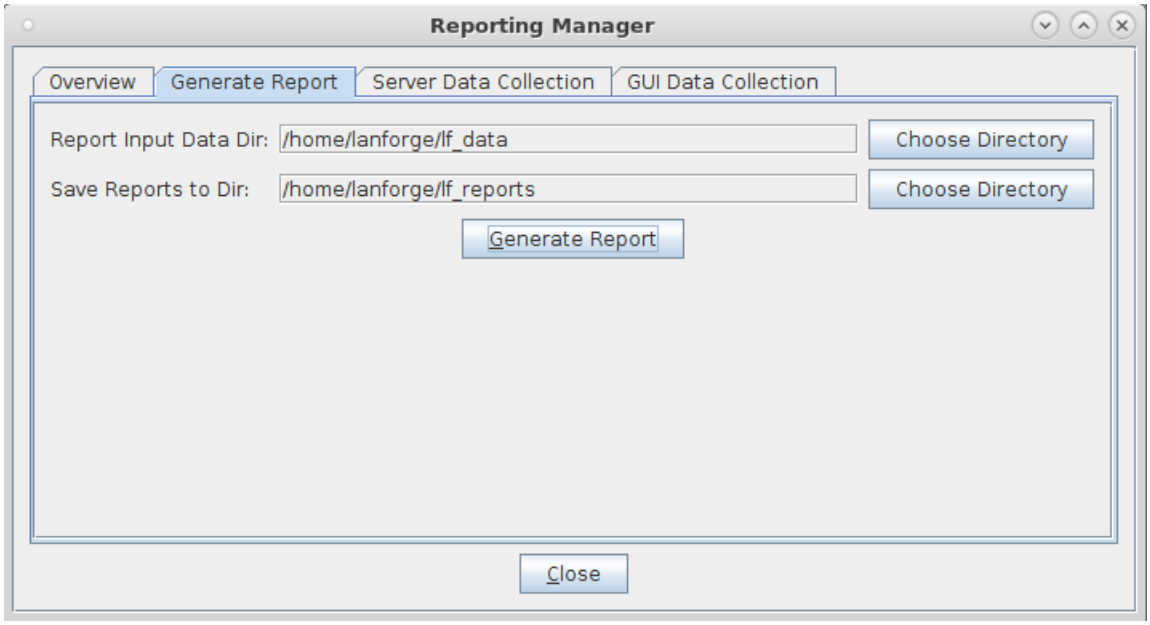
GUI Data Collection (Desktop Folder)

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 `lf_data` that probably means `C:\Users\mumble\AppData\Local\LANforge-GUI\lf_data`. You probably want to put in a fully qualified path thats more intuitive, like `C:\Users\mumble\Documents\lf_data`.



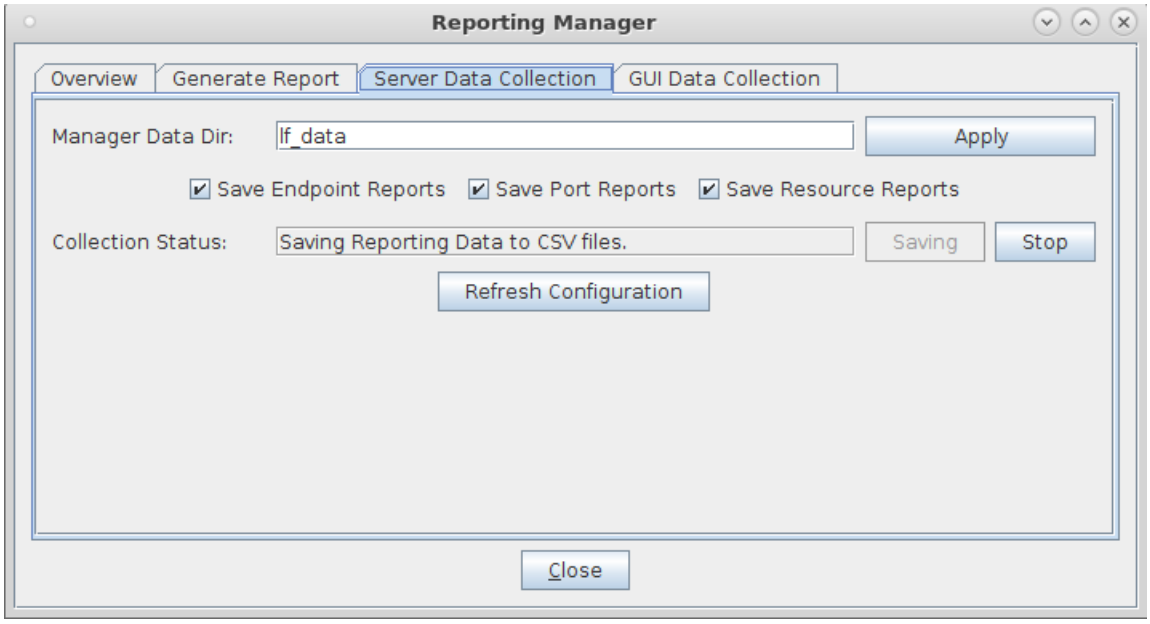
Generate Report

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)

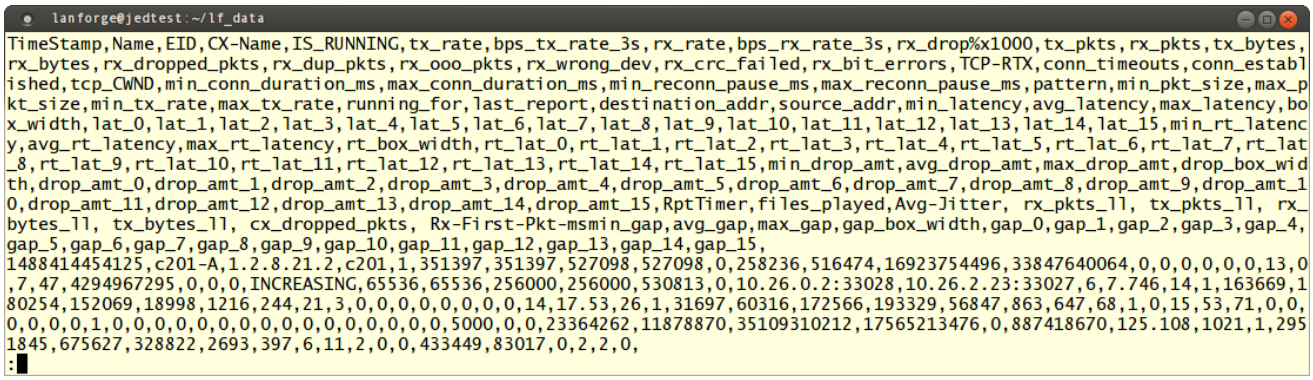
If your test scenario runs longer than your GUI can be up, you can configure the LANforge server to collect the data. The directory is relative to the `/home/lanforge` directory, so if you enter `lf_data`, you would find the CSV files in `/home/lanforge/lf_data`.



You can take a look at the data files easily. Here is a server data collection directory:

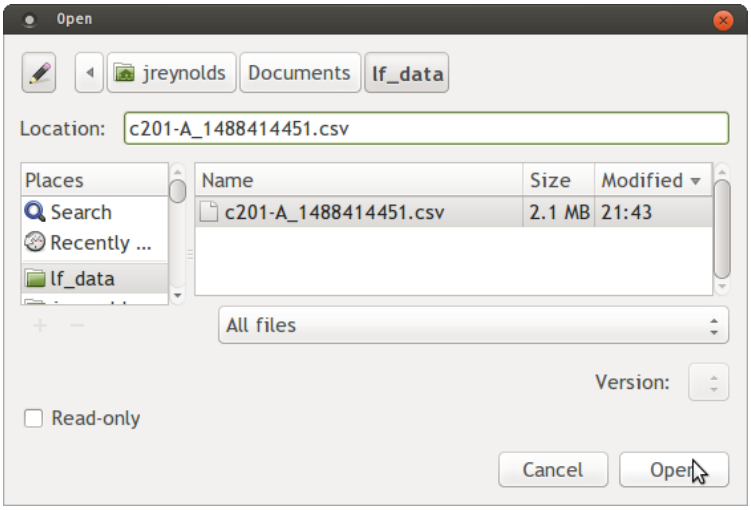
```
lanforge@jedtest ~/lf_data
 > r
total 2628
-rw-r--r-- 1 root root  31465 Mar  1 16:52 wlan2_1.1.7_1488414451.csv
-rw-r--r-- 1 root root  31465 Mar  1 16:52 wlan1_1.1.6_1488414451.csv
-rw-r--r-- 1 root root  31465 Mar  1 16:52 wlan0_1.1.5_1488414451.csv
-rw-r--r-- 1 root root  90889 Mar  1 16:52 wiphy2_1.1.4_1488414451.csv
-rw-r--r-- 1 root root 125299 Mar  1 16:52 wiphy1_1.1.3_1488414451.csv
-rw-r--r-- 1 root root 101801 Mar  1 16:52 wiphy0_1.1.2_1488414451.csv
-rw-r--r-- 1 root root 138049 Mar  1 16:52 udp200-B_1488414451.csv
-rw-r--r-- 1 root root 137626 Mar  1 16:52 udp200-A_1488414451.csv
-rw-r--r-- 1 root root 160328 Mar  1 16:52 tcp200-B_1488414451.csv
-rw-r--r-- 1 root root 158351 Mar  1 16:52 tcp200-A_1488414451.csv
-rw-r--r-- 1 root root  26376 Mar  1 16:52 resource_jedtestcandelatechcom_1.1_1488414451.csv
-rw-r--r-- 1 root root 114505 Mar  1 16:52 eth1_1.1.1_1488414451.csv
-rw-r--r-- 1 root root  43937 Mar  1 16:52 eth0_1.1.0_1488414451.csv
-rw-r--r-- 1 root root 168161 Mar  1 16:52 c201-B_1488414451.csv
-rw-r--r-- 1 root root 169329 Mar  1 16:52 c201-A_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar  1 16:52 wlan2_1.2.7_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar  1 16:52 wlan1_1.2.6_1488414451.csv
-rw-r--r-- 1 root root  27937 Mar  1 16:52 wlan0_1.2.5_1488414451.csv
-rw-r--r-- 1 root root  83629 Mar  1 16:52 wiphy2_1.2.4_1488414451.csv
-rw-r--r-- 1 root root 114613 Mar  1 16:52 wiphy1_1.2.3_1488414451.csv
-rw-r--r-- 1 root root 100485 Mar  1 16:52 wiphy0_1.2.2_1488414451.csv
-rw-r--r-- 1 root root 327241 Mar  1 16:52 vap0_1.1.8_1488414451.csv
-rw-r--r-- 1 root root  40057 Mar  1 16:52 sta205_1.2.10_1488414451.csv
-rw-r--r-- 1 root root  48097 Mar  1 16:52 sta200_1.2.8_1488414451.csv
-rw-r--r-- 1 root root  39289 Mar  1 16:52 sta100_1.2.9_1488414451.csv
-rw-r--r-- 1 root root  21691 Mar  1 16:52 resource_kedtest_1.2_1488414451.csv
-rw-r--r-- 1 root root 105065 Mar  1 16:52 eth1_1.2.1_1488414451.csv
-rw-r--r-- 1 root root  40789 Mar  1 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:

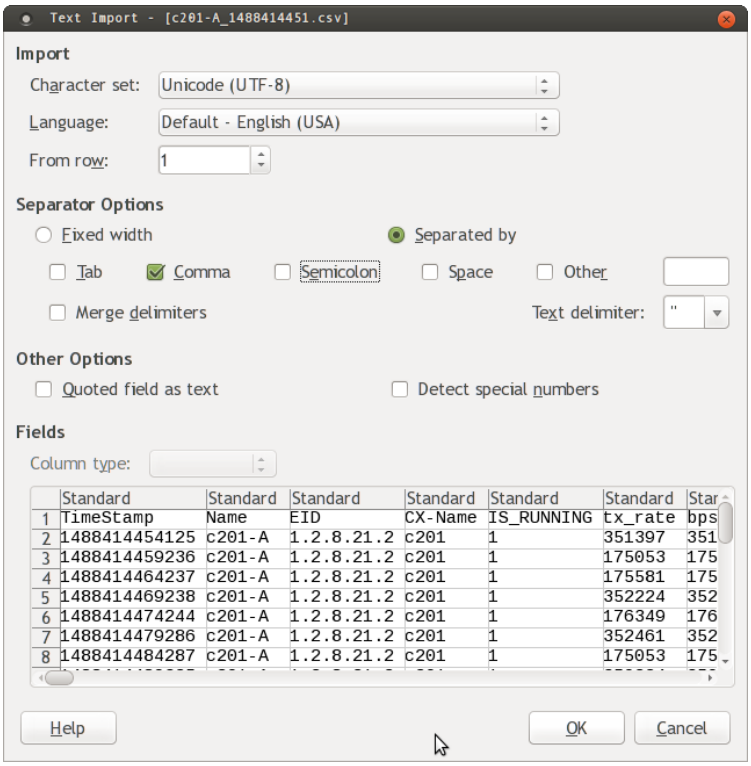


Using Libre Office

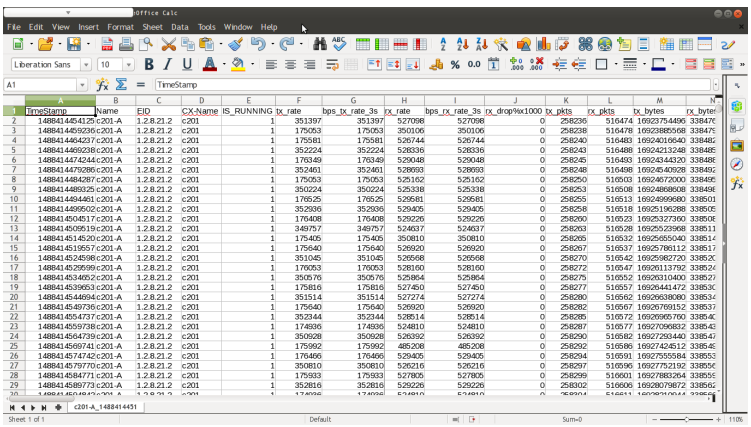
Importing the file into a spreadsheet like LibreOffice Calc is simple:



You only need to separate on comma (,)



	Standard	Standard	Standard	Standard	Standard	Standard	Star
1	TimeStamp	Name	EID	CX-Name	IS_RUNNING	tx_rate	bps
2	1488414454125	c201-A	1.2.8.21.2	c201	1	351397	351
3	1488414459236	c201-A	1.2.8.21.2	c201	1	175053	175
4	1488414464237	c201-A	1.2.8.21.2	c201	1	175581	175
5	1488414469238	c201-A	1.2.8.21.2	c201	1	352224	352
6	1488414474244	c201-A	1.2.8.21.2	c201	1	176349	176
7	1488414479286	c201-A	1.2.8.21.2	c201	1	352461	352
8	1488414484287	c201-A	1.2.8.21.2	c201	1	175053	175



The timestamp column

Libre Office does not have a builtin 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 an dort 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
TimeStamp
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` switches as an input file.

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

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

Not everthing you do in perl is going to be a one-liner. Here's an example of the same script as a 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";
```