SAND2021-13645C

Exceptional service in the national interest

Sandia National Laboratories

OGC | Open Grid Computing, Austin, TX

OGC

LDMS Version 4.3.8 Advanced Tutorial: Part 2

https://github.com/ovis-hpc/ovis

Jim Brandt, Ann Gentile

Sandia National Laboratories

Tom Tucker

Open Grid Computing, Inc.

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Exercise 2.1: Re-configuration

Manually Modify Verbosity of a Running ldmsd

OGC Sandia National Laboratories

This is useful if you don't want to fill up your log files but want to perform a more in-depth inspection of what is happening on a running ldmsd while troubleshooting

- In one terminal window start a sampler and an aggregator

  $ ~/ldmscon2021/advanced/exercises/ldms/scripts/E2.1/**start_ldms_e2.1_sampler.sh**
  $ ~/ldmscon2021/advanced/exercises/ldms/scripts/E2.1/**start_ldms_e2.1_agg.sh**

  Since the daemons are running without verbosity specified it is the default of "ERROR". The associated **log files should be empty**

- Verify that both daemons are running

  $ ps auxw | grep ldmsd should return something like:

  ldmsd -x sock 10001 –l /home/…/ldms/logs/e2.1_sampler.log

  ldmsd -x sock 20001 –l /home/…/ldms/logs/e2.1_agg.log

Modify Verbosity of a Running ldmsd (cont)

- In the second terminal window on the same host run:

```
$ tail –f ~/ldmscon2021/advanced/exercises/ldms/logs/e2.1_agg.log
```

- In the first window use the **ldmsd_controller** to change loglevel to DEBUG:

```
$ echo "loglevel level=DEBUG" | ldmsd_controller --host localhost --xprt sock --port  20001
```

- You now should see a continuous stream of information being logged at a variety of levels to the aggregator log file displayed in your second terminal window
- You can experiment with changing the levels between the states of: DEBUG, INFO, ERROR, CRITICAL, and QUIET to see how the information flowing to the log file can be changed dynamically

Use a Script to Modify Verbosity of a ldmsd

A script to modify the verbosity of a running ldmsd can be found at:
~/ldmscon2021/advanced/exercises/ldms/scripts/**E2.1/change_verbosity_e2.1.sh**

```
=====================================
#!/bin/bash
VERBOSITY=$1
PORT=$2
if [ -z ${VERBOSITY} ] | [ -z ${PORT} ]
then
   echo "Usage: change_verbosity_e2.1.sh <verbosity level> <ldmsd port>"
   exit
fi
echo "loglevel level=${VERBOSITY}" | ldmsd_controller --host localhost --xprt sock --port ${PORT}
=====================================
```

Try it (Arguments are verbosity level and port number.)

$ ~/ldmscon2021/.../change_verbosity_e2.1.sh DEBUG 20001

Use a Script to Modify the Collection Interval of a Sampler ldmsd

A script to modify the verbosity of a running ldmsd can be found at:
~/ldmscon2021/advanced/exercises/ldms/scripts/**E2.1/change_sample_interval_e2.1_sampler.sh**

```
==========================
#!/bin/bash
PLUGIN=$1
INTERVAL=$2
if [ -z ${PLUGIN} ] | [ -z ${INTERVAL} ]
then
   echo "Usage: change_sample_interval_e2.1_sampler.sh <plugin name> <interval in usec>"
   exit
fi
echo "stop name=${PLUGIN}" | ldmsd_controller --host localhost --xprt sock --port 10001
echo "start name=${PLUGIN} interval=${INTERVAL} offset=0" | ldmsd_controller --host localhost --xprt
sock --port 10001
==========================
```

Try it (Arguments are plugin name and interval in usec)

$ ~/ldmscon2021/.../change_sample_interval_e2.1_sampler.sh meminfo 5000000

Use a Script to Remove a Sampler Plugin

A script to modify the verbosity of a running ldmsd can be found at:
~/ldmscon2021/advanced/exercises/ldms/scripts/**E2.1/remove_sampler_plugin_e2.1_sampler.sh**

```
==============================
#!/bin/bash
PLUGIN=$1
if [ -z ${PLUGIN} ]
then
   echo "Usage: remove_sampler_plugin_e2.1_sampler.sh <plugin name (e.g., meminfo)>"
   exit
fi
echo "stop name=${PLUGIN}" | ldmsd_controller --host localhost --xprt sock --port 10001
echo "term name=${PLUGIN}" | ldmsd_controller --host localhost --xprt sock --port 10001
==============================
```

Try it (Argument is a plugin name. Note: This script uses port 10001 only)

$ ~/ldmscon2021/.../remove_sampler_plugin_e2.1_sampler.sh vmstat

Use a Script to Load a Sampler Plugin

A script to modify the verbosity of a running ldmsd can be found at:
~/ldmscon2021/advanced/exercises/ldms/scripts/**E2.1/load_sampler_plugin_e2.1_sampler.sh**

```
==============================
#!/bin/bash
PLUGIN=$1
if [ -z ${PLUGIN} ]
then
   echo "Usage: load_sampler_plugin_e2.1_sampler.sh <plugin name (e.g., meminfo)>"
   exit
fi
CONFDIR=/home/brandt/ldmscon2020/advanced/exercises/ldms/conf/E2.1
cat ${CONFDIR}/${PLUGIN}.conf | ldmsd_controller --host localhost --xprt sock --port 10001
==============================
vmstat.conf:
load name=vmstat
config name=vmstat producer=node-2 instance=node-2/vmstat component_id=2 job_set=node-2/slurm
uid=1005 gid=1005 perm=0755
start name=vmstat interval=1000000 offset=0
==============================
```

Try it (Argument is a plugin name. Note: This script uses port 10001 only)

$ ~/ldmscon2021/.../load_sampler_plugin_e2.1_sampler.sh vmstat

Exercise 2.2: Authentication

Using LDMS Authentication

- “none” (no) authentication is the default and what you used in the basic and advanced part 1 exercises
- Examples:

```
$ldmsd -x sock:10001
$ldmsd -x sock:10001 -a none
```

- auth “none”
  - All **data is accessible to anyone** who has network access and wants to query for it
  - **Anyone** who has network access **can (re)configure** the running ldmsd
- auth “ovis”
  - Network access is available to anyone who knows the shared secret
  - **Anyone** who has network access **can (re)configure** the running ldmsd
- auth “munge”
  - Data is accessible to user.group who has been authenticated on this connection
  - Each metric set contains user.group and permissions
  - Only requests on a connection from user.group can access metric set data for which they have permissions
  - Configuration changes cannot be made on a daemon for which the connection user.group is not authorized

Using “ovis” Shared Secret Authentication

OGC Sandia National Laboratories

SYNOPSIS: ldms_app **-a ovis [-A conf=**PATH**]**

DESCRIPTION: ovis_auth uses a shared secret to authenticate the connection. The secret is a text file containing the line: **secretword=X where X is a string at least 8 characters long.**

The following four locations are checked, **in order**, for the secret:

1) **full file path** given on the **command line** via "-A conf=authfile",
2) **full file path** defined by the environment variable **“LDMS_AUTH_FILE”**,
3) **$HOME/.ldmsauth.conf,** and
4) **$SYSCONFDIR/ldmsauth.conf** (e.g. /etc/ldmsauth.conf).

$HOME is taken from /etc/password and **$SYSCONFDIR** is determined at ldms compile time.

- If one of these is not set, the search continues with the next location.
- **A failure in reading one, if the file exists, ends the search** and is a failure to authenticate.

The secret file **permissions** must be set to **600 or more restrictive.**

Run An ldmsd Sampler Daemon Using Shared Secret Authentication

- Check contents, ownership, and permissions of: ~/ldmscon2021/advanced/../ldms/conf/E2.2/my_secret

```
$ ls -l ~/ldmscon2021/advanced/../ldms/conf/E2.2/my_secret
$ -rw------- 1 uid gid   20 Oct  27  2021 my_secret
$ cat ~/ldmscon2021/advanced/../ldms/conf/E2.2/my_secret
$ secretword=shhhhhhh
```

Run a sampler ldmsd using “ovis” shared secret authentication: …/E2.2/start_sampler_shared_secret.sh

```
$ ldmsd -x sock:10001 \
-l ~/ldmscon2021/advanced/../ldms/logs/E2.2/sampler_auth.log \
-c ~/ldmscon2021/advanced/../ldms/conf/E2.2/simple_sampler.conf \
–a ovis –A conf=/home/<user>/ldmscon2021/advanced/…/ldms/conf/E2.2/my_secret
```

- `-x:` transport : listener port
- `-l:` Specify the log file path
- `-c:` Specify configuration file
- `-a:` Authentication method and associated file

ldms_ls Using “ovis” Shared Secret Authentication

- Try ldms_ls with no authentication

```
$ ldms_ls -h localhost -x sock -p 10001
```

```
Connection failed/rejected
```

- Now try ldms_ls with same authentication as sampler daemon

```
$ ldms_ls -h localhost -x sock -p 10001 -a ovis -A \
conf=/home/<user>/ldmscon2021/advanced/exercises/ldms/conf/E2.2/my_secret
```

```
node-64/vmstat
node-64/meminfo
```

- Now set the LDMS_AUTH_FILE environment variable to point to your shared secret

```
$ export LDMS_AUTH_FILE=/home/<user>/ldmscon2021/advanced/…/conf/E2.2/my_secret
```

- Use ldms_ls without specifying the location of your shared secret on the command line

```
$ ldms_ls -h localhost -x sock -p 10001 -a ovis
```

```
node-64/vmstat
node-64/meminfo
```

Aggregator -> sampler authentication

- Run an ldmsd aggregator using shared secret authentication that matches your sampler (…/conf/E2.2 my_secret.sh)

```
$ ldmsd -x sock:20001
-l /home/<user>/ldmscon2021/advanced/.. /ldms/logs/E2.2_aggd.log
-a ovis -A conf=/home/<user>/ldmscon2021/advanced/../ldms/conf/E2.2/my_secret
```

- `-x:` transport : listener port
- `-l:` Specify the log file path
- `-a:` Specify alternate shared secret file

Aggregator -> sampler authentication

- Kill your previous aggregator ldmsd
- Run an ldmsd aggregator using shared secret authentication **not matching** your sampler
  - **Note:** you will need to create your "other_secret" in the conf directory

```
$ ldmsd -x sock:20001
-l /home/<user>/ldmscon2021/advanced/../ldms/logs/E2.2_aggd.log
-a ovis -A conf=/home/<user>/ldmscon2021/advanced/ldms/conf/E2.2/other_secret
```

- -x: transport : listener port
- -l: Specify the log file path
- -a: Specify alternate shared secret file
- Kill all of your running ldmsd and verify they are gone

```
$killall ldmsd
```

Using Munge Authentication

- Run a daemon using munge authentication:

```
$ldmsd -x sock:10001 -v DEBUG -l \
~/ldmscon2021/advanced/../ldms/logs/E2.2/samplerd.log -c \
~/ldmscon2021/advanced/../ldms/conf/E2.2/sampler.conf -a munge
```

Note: This will also write out DEBUG-level information to the specified (-l) log.

- Run ldms_ls on that node to see sets, set meta-data, and set data:

```
$ ldms_ls -h localhost -x sock -p 10001 -a munge
$ ldms_ls -h localhost -x sock -p 10001 -v -a munge
$ ldms_ls -h localhost -x sock -p 10001 -l -a munge
```

Note: Users will not be able to authenticate to a daemon launched using "**-a munge**" if not querying using the "-a munge" option.

Note: Users will only be able to see sets as allowed by set permissions in response to ldms_ls.

Using Mixed Authentication Methods

- **Edit sampler configuration file for E2.2 and uncomment:**

```
#auth_add name=munge_port plugin=munge    # Defines a domain for munge authentication
#listen auth=munge_port xprt=sock port=10001    # use munge_port for listening on sock:10001

#auth_add name=ovis_port plugin=ovis \
#conf=/home/<user>/ldmscon2021/advanced/exercises/ldms/conf/E2.2/my_secret
#listen auth=ovis_port xprt=sock port=30001
```

- **Edit aggregator configuration file for E2.2 and uncomment:**

```
#auth_add name=munge_port plugin=munge # Defines a tag for munge authentication
…20000000 #auth=munge_port    # Use munge authentication for this producer
#listen auth=munge_port xprt=sock port=20001    # Use munge authenticaion on sock:20001
```

Using Mixed Authentication Methods

- Start sampler daemon using modified E2.2/simple_sampler.conf

```
$ldmsd -c /home/<user>/ldmscon2021/advanced/…/conf/E2.2/simple_sampler.conf
```

- Start agg daemon using modified E2.2/agg.conf

```
$ldmsd -c /home/<user>/ldmscon2021/advanced/…/conf/E2.2/agg.conf
```

Using Mixed Authentication Methods

- Try using ldms_ls to talk to sampler on each of ports 10001 (munge) and 30001 (ovis)

```
$ldms_ls -h localhost -x sock -p 10001 -a munge
node-64/vmstat
node-64/meminfo
```

```
$ldms_ls -h localhost -x sock -p 30001 -a ovis -A
conf=/home/user64/ldmscon2021/advanced/exercises/ldms/conf/E2.2/my_secret
node-64/vmstat
node-64/meminfo
```

- Try using ldms_ls to talk to aggregator on port 20001 (munge)

```
$ldms_ls -h localhost -x sock -p 20001 -a munge
node-64/vmstat
node-64/meminfo
```

- Using each of –a munge and –a ovis, and ports 10001 and 30001 respectively, query ldmsd on node-64 and see how **munge restricts access based on uid, gid, and perms**

Exercise 2.3: Scale Emulation

Run Many ldmsd Per Compute Node

- Differentiate between ldmsd and sampler configurations using port number
  - ldmsd –x sock:10001 –c /home/<user>/advanced/exercises/ldms/conf/scripts/E2.3/sampler_10001.conf
  - …
  - ldmsd –x sock:10100 –c /home/<user>/advanced/exercises/ldms/conf/scripts/E2.3 /sampler_10100.conf
- There is a sample script for this: /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/E2.3/**start_multi_samplers.sh**
  - Generates configuration files and writes them to: /home/<user>/ldmscon2021/advanced/exercises/ldms/conf/E2.3/multi_sampler_confs/multi_sampler_<port#>.conf
  - Starts a ldmsd per configuration file

Example Multi-ldmsd Sampler Configuration

OGC Sandia National Laboratories

multi_sampler_10001.conf

```
================
load name=meminfo
config name=meminfo producer=node-2_10001 instance=node-2_10001/meminfo component_id=210001
start name=meminfo interval=1000000 offset=0

load name=vmstat
config name=vmstat producer=node-2_10001 instance=node-2_10001/vmstat component_id=210001
start name=vmstat interval=1000000 offset=0
================
```

Example Multi-ldmsd Sampler Start Script

start_multi_samplers.sh

```
================
#!/bin/bash
ME=$(whoami)
CONFDIR=/home/${ME}/advanced/exercises/ldms/conf/E2.3/multi-samplers_conf
LDMSD_AUTH_PLUGIN=none
LOGLEVEL=QUIET
LDMSD_PLUGIN_LIBPATH=/opt/ovis/lib/ovis-ldms
ZAP_LIBPATH=/opt/ovis/lib/ovis-lib


ldmsd -x sock:10001 -v ${LOGLEVEL} -c ${CONFDIR}/sampler_10001.conf -r /home/${ME}/advanced/exercises/ldms/scripts/E2.3/sampler_10001.pid -a ${LDMSD_AUTH_PLUGIN}
ldmsd -x sock:10002 -v ${LOGLEVEL} -c ${CONFDIR}/sampler_10002.conf -r /home/${ME}/advanced/exercises/ldms/scripts/E2.3/sampler_10002.pid -a ${LDMSD_AUTH_PLUGIN}
.
.
.
================
```

Example Multi-ldmsd Aggregator Configuration

```
prdcr_add name=prdcr-grp1_10001 host=compute1 port=10001 xprt=sock type=active interval=20000000
prdcr_start name=prdcr-grp1_10001
prdcr_add name=prdcr-grp2_10002 host=compute1 port=10002 xprt=sock type=active interval=20000000
prdcr_start name=prdcr-grp2_10002
…
updtr_add name=updtr1 interval=1000000 offset=200000
updtr_prdcr_add name=updtr1 regex=prdcr-grp1_.*
updtr_start name=updtr1

updtr_add name=updtr2 interval=1000000 offset=200000
updtr_prdcr_add name=updtr2 regex=prdcr-grp2_.*
updtr_start name=updtr2

load name=store_csv
config name=store_csv path=/home/<user>/advanced/exercises/ldms/data/  buffer=0

strgp_add name=meminfo-store_csv plugin=store_csv container=meminfo_metrics schema=meminfo
strgp_start name=meminfo-store_csv

strgp_add name=vmstat-store_csv plugin=store_csv container=vmstat_metrics schema=vmstat
strgp_start name=vmstat-store_csv
```

Run Many ldmsd Per Compute Node

- Modify the E2.3 **start_multi_samplers.sh** script to generate the number of confs and samplers you want
- Run the E2.3 **start_agg.sh** and look at the resulting files
- Use ps auxw | grep ldmsd to see the many running ldmsd
- Use ldms_ls –h localhost –x sock –p 20001 to see what your aggregator is aggregating
- Run **pkill ldmsd** to kill your aggregator daemon

Note: Running in this containerized environment your compute and memory resources are much more limited than on a traditional compute node. Keep the number of sampler ldmsd to 10 or less for this exercise.

Use of the ”test_sampler” Plugin

- Generation of arbitrary metric sets whose component values change in an expected way
- Enables generation of sets that have the same content as those being collected from existing telemetry but without the backend overhead (e.g. collecting GPU metrics can take ~10ms).
  - Enables large scale infrastructure testing of large sets and/or large numbers of sets
- Example configuration:

```
load name=test_sampler

config name=test_sampler action=add_schema schema=set1 \
metrics=meta1:meta:U64:1234,component_id:meta:U64:${COMPONENT_ID},job_id:data:U64:1,scalar1:data:U64:1,scalar2:data:U64:100,array_100_element:data:U64_ARRAY:1:100

config name=test_sampler action=add_schema schema=set2 \
metrics=meta1:meta:U64:5678,component_id:meta:U64:${COMPONENT_ID},job_id:data:U64:1,scalar1:data:U64:1,scalar2:data:U64:100,array_10_element:data:U64_ARRAY:100:10

config name=test_sampler action=add_set schema=set1 instance=${MYHOST}_${COMPONENT_ID}/set1 \
producer=${MYHOST}_${COMPONENT_ID}

config name=test_sampler action=add_set schema=set2 instance=${MYHOST}_${COMPONENT_ID}/set2 \
producer=${MYHOST}_${COMPONENT_ID}

start name=test_sampler interval=1000000 offset=0
```

Output From ”test_sampler” Configuration

node-2_2/set2: consistent, last update: Wed Aug 05 21:16:21 2020 -0500 [1416us]

M u64	meta1	5678
M u64	component_id	2
D u64	job_id	1
D u64	scalar1	55
D u64	scalar2	154
D u64[]	array_10_element	154,154,154,154,154,154,154,154,154,154

Note: If there is interest we can use a LDMS User Group meeting to demonstrate how to use an actual metric set as a template to generate an emulation test set.

Exercise 2.4: CSV Store configs

Store Philosophy

- Support a few stores writing to well-known formats that could be easily converted from there. CSV is one such standard.
- Features:
  - One store plugin can be configured to handle multiple schema (e.g., meminfo vs vmstat) and multiple instances (e.g., all nodes' meminfo)
  - Defaults and overrides (e.g., handle different schema differently)
  - Can be written to a file or piped to downstream consumers (e.g., port for syslog)
  - File Rollover and closed file manipulations
  - Buffering and/or flushing
  - File permission setting
  - Optional separate header (to support loading into a database)
  - IETF 4180 quoting for header column names
  - Configuration file for options to support multiplicity of options

CSV Store: (1) configuration for L1 samplers

- ldmsd configuration files: ~/ldmscon2021/advanced/…/conf/E2.4
- 2 samplers with different hostnames and component_ids.

$ cat sampler_realnode.conf

```
load name=meminfo
config name=meminfo producer=${HOST} instance=${HOST}/meminfo component_id=${COMP_ID}
start name=meminfo interval=1000000 offset=0

load name=vmstat
config name=vmstat producer=${HOST} instance=${HOST}/vmstat component_id=${COMP_ID}
start name=vmstat interval=1000000 offset=0
```

$ cat sampler_fakenode.conf

Hardwired fake node-1000 and comp_id
Only has meminfo @ 5 sec intervals

```
load name=meminfo
config name=meminfo producer=node-1000 instance=node-1000/meminfo component_id=1000
start name=meminfo interval=5000000 offset=0
```

CSV Store: (2) configuration for L2 store

OGC Sandia National Laboratories

$ cat agg.conf

```
…
load name=store_csv
# store can take the configure lines in a configuration file\
(opt_file=XXX.conf) or called as multiple config lines
config name=store_csv path=XXX/data altheader=1  buffer=0 rolltype=1
rollover=60 \ # DEFAULT
# override for schema vmstat (container and schema uniquely identify)
config name=store_csv altheader=0 container=csv schema=vmstat

strgp_add name=policy_mem plugin=store_csv container=csv\
schema=meminfo
strgp_prdcr_add name=policy_mem regex=.*
strgp_start name=policy_mem
…
<<similar for vmstat>>
```

Config line without container and schema is the default:

- path = path to store. path/container/schema are the csv files
- altheader = 1 – write header to a separate file
- buffer = 0 – don't use system buffering, flush after every line
- Rolltype =1 – rollover the output file based on time
- Rollover = 60 – every 60 seconds

Others are overrides:

- Only for csv/vmstat altheader = 0

CSV Store: (3) Start both L1 and L2

- Run scripts: ~/ldmscon2021/advanced/…/scripts/E2.4

```
$ ./start_sampler_realnode.sh
$ ./start_sampler_fakenode.sh
$ ./start_agg.sh
$ ps aux | grep ldmsd   #shows all 3 ldmsd
$ ldms_ls -x sock -p 21001 –a munge   #query the aggregator
    node-63/vmstat
    node-63/meminfo
    node-1000/meminfo
```

CSV Store: (4) Examine the store

- store: /home/<user>/ldmscon2021/advanced/…/ exercises/ldms/data/E2.4/csv

$ ls

```
meminfo.1635307075  meminfo.HEADER.1635307075  vmstat.1635307075
meminfo.1635307134  meminfo.HEADER.1635307134  vmstat.1635307134
meminfo.1635307194  meminfo.HEADER.1635307194  vmstat.1635307194
```

- Naming convention with rollover = schema.*epochtime.* Rollover every 60 sec.

$ cat meminfo.HEADER.XXX

```
#Time,Time_usec,ProducerName,component_id,job_id,app_id,MemTo
Inactive(anon),Active(file),Inactive(file),Unevictable,Mlocke
nreclaim,KernelStack,PageTables,NFS_Unstable,Bounce,Writeback
```

Override: vmstat has header within its data file.

$ cat vmstat.XXX

```
#Time,Time_usec,ProducerName,component_id,job_id,app_id,nr_free_pages,…
1635307195.001986,1986,node-63,63,0,0,47260754,…
1635307196.001820,1820,node-63,63,0,0,47260891,…
1635307197.002230,2230,node-63,63,0,0,47260800,…
```

CSV Store: (5) Examine the store timings:

- store: /home/<user>/ldmscon2021/advanced/…/exercises/ldms/data/E2.4/csv

```
$ grep node-63 meminfo.1596679548          # every 1 sec
1635307538.001148,1148,node-63,63,0,0,263519336,189045864,249258376,14716,60933672,…
1635307539.003048,3048,node-63,63,0,0,263519336,189045996,249258536,14716,60933716,…
1635307540.001397,1397,node-63,63,0,0,263519336,189045748,249258300,14716,60933720,…
$ grep node-1000 meminfo.1596679548        # every 5 sec
1635307540.006179,6179,node-1000,1000,0,0,263519336,189045748,249258300,14716,60933728,…
1635307545.005514,5514,node-1000,1000,0,0,263519336,189045992,249258756,14716,60933960,…
1635307550.005870,5870,node-1000,1000,0,0,263519336,189046068,249258916,14716,60934120,…
```

Section 2.5: Systemd Configuration

sampler.conf

```
env SAMPLE_INTERVAL=1000000
env SAMPLE_OFFSET=0
env MYHOST=$(eval hostname)
env COMPONENT_ID=$(echo ${MYHOST} | sed 's/compute//g')

load name=slurm_sampler
config name=slurm_sampler component_id=${COMPONENT_ID} producer=${MYHOST} \
    instance=${MYHOST}/slurm_sampler task_count=8 job_count=8
start name=slurm_sampler interval=1000000 offset=0

load name=meminfo
config name=meminfo producer=${MYHOST} instance=${MYHOST}/meminfo component_id=${COMPONENT_ID}
start name=meminfo interval=${SAMPLE_INTERVAL} offset=${SAMPLE_OFFSET}

load name=vmstat
config name=vmstat producer=${MYHOST} instance=${MYHOST}/vmstat component_id=${COMPONENT_ID}
start name=vmstat interval=${SAMPLE_INTERVAL} offset=${SAMPLE_OFFSET}
```

ldmsd.sampler.env

```
# This file contains environment variables for ldmsd.sampler, which will affect
# ldmsd initial configuration (e.g. transport, named socket path)

# LDMS transport option (sock, rdma, or ugni)
LDMSD_XPRT=sock
# LDMS Daemon service port
LDMSD_PORT=411
# LDMS spank transport option (sock, rdma, or ugni)
LDMSD_SPANKD_XPRT=sock
# LDMS spankd port
LDMSD_SPANKD_PORT=10000
# LDMS memory allocation
LDMSD_MEM=512K
LDMSD_VERBOSE=QUIET
# Log file control. The default is to log to syslog.
# LDMSD_LOG_OPTION="-l /var/log/ldmsd.log"

# Authentication method
LDMSD_AUTH_PLUGIN=munge

# Authentication options
#LDMSD_AUTH_OPTION="-A conf=/opt/ovis/etc/ldms/ldmsauth.conf"

# LDMS plugin configuration file, see /opt/ovis/etc/ldms/sampler.conf for an example
LDMSD_PLUGIN_CONFIG_FILE=/opt/ovis/etc/ldms/sampler.conf

# These are configured by configure script, no need to change.
LDMSD_PLUGIN_LIBPATH=/opt/ovis/lib64/ovis-ldms
ZAP_LIBPATH=/opt/ovis/lib64/ovis-lib
```

aggregator.conf

```
# LDMS_XPRT is set in ldmsd.aggregator.env


# Adding 1 producer per ldmsd.sampler, if you have thousands of nodes, feel free
# to use a script to generate the configuration file. Producers take care only of
# the LDMS connection aspect. Updater will take care of the data updating logic.
prdcr_add name=nid00012 host=nid00012 type=active xprt=ugni port=411 interval=20000000
prdcr_add name=nid00013 host=nid00013 type=active xprt=ugni port=411 interval=20000000
prdcr_add name=nid00014 host=nid00014 type=active xprt=ugni port=411 interval=20000000
prdcr_start_regex regex=.*
# Create an updater for all producers and all sets.
updtr_add name=update_all interval=1000000 offset=500
# Add all producers.
updtr_prdcr_add name=update_all regex=.*
# By default, all sets in a producer will be updated.
# Start the updater
updtr_start name=update_all
```

ldmsd.aggregator.env

```
# This file contains environment variables for ldmsd.sampler, which will affect
# ldmsd initial configuration (e.g. transport, named socket path)
# LDMS transport option (sock, rdma, or ugni)
LDMSD_XPRT=sock
# LDMS Daemon service port
LDMSD_PORT=412
# LDMS memory allocation
LDMSD_MEM=2G
# Number of event threads
LDMSD_NUM_THREADS=8
LDMSD_ULIMIT_NOFILE=100000
LDMSD_VERBOSE=CRITICAL
# Log file control. The default is to log to syslog.
LDMSD_LOG_OPTION="-l /var/log/ldmsd.log"
```

```
# Authentication method
LDMSD_AUTH_PLUGIN=munge

# Authentication options
LDMSD_AUTH_OPTION="-A conf=/opt/ovis/etc/ldms/ldmsauth.conf"

LDMSD_PLUGIN_CONFIG_FILE=/opt/ovis/etc/ldms/aggregator.conf

# These are configured by configure script, no need to change.
LDMSD_PLUGIN_LIBPATH=/opt/ovis/lib64/ovis-ldms
ZAP_LIBPATH=/opt/ovis/lib64/ovis-lib
```

ldmsd.sampler.service

```
[Unit]
Description = LDMS Sampler Daemon
Documentation = http://ovis.ca.sandia.gov

[Service]
Type = forking
EnvironmentFile = /opt/ovis/etc/ldms/ldmsd.sampler.env
Environment = HOSTNAME=%H
ExecStartPre = /bin/mkdir -p /opt/ovis/var/run/ldmsd
ExecStartPre = -/bin/bash -c "test -n
\"${LDMS_JOBINFO_DATA_FILE}\" && touch
${LDMS_JOBINFO_DATA_FILE} || touch
/var/run/ldms_jobinfo.data"
ExecStart = /opt/ovis/sbin/ldmsd \
        -x ${LDMSD_XPRT}:${LDMSD_PORT} \
        -x ${LDMSD_SPANKD_XPRT}:${LDMSD_SPANKD_PORT} \
        -c ${LDMSD_PLUGIN_CONFIG_FILE} \
        -a ${LDMSD_AUTH_PLUGIN} \
        -v ${LDMSD_VERBOSE} \
        -m ${LDMSD_MEM} \
        $LDMSD_LOG_OPTION \
        -r /opt/ovis/var/run/ldmsd/sampler.pid

[Install]
WantedBy = default.target
```

Note: On our production cray machines we set up the following link: /etc/systemd/system/ldmsd.sampler.service -> /opt/ovis/etc/systemd/system/ldmsd.sampler.service
We then install the service file, via rpm install, to /opt/ovis/etc/systemd/system/ldmsd.sampler.service

ldmsd.aggregator.service

```
[Unit]
Description = LDMS Daemon
Documentation = http://ovis.ca.sandia.gov

[Service]
Type = forking
LimitNOFILE = ${LDMSD_ULIMIT_NOFILE}
EnvironmentFile =
/opt/ovis/etc/ldms/ldmsd.aggregator.env
Environment = HOSTNAME=%H
ExecStartPre = /bin/mkdir -p /opt/ovis/var/run/ldmsd
ExecStart = /opt/ovis/sbin/ldmsd \
        -x ${LDMSD_XPRT}:${LDMSD_PORT} \
        -c ${LDMSD_PLUGIN_CONFIG_FILE} \
        -a ${LDMSD_AUTH_PLUGIN} \
        -v ${LDMSD_VERBOSE} \
        -m ${LDMSD_MEM} \
        $LDMSD_LOG_OPTION \
        -P ${LDMSD_NUM_THREADS} \
        -r /opt/ovis/var/run/ldmsd/aggregator.pid

[Install]
WantedBy = default.target
```

Note: On our production cray machines we set up the following link: /etc/systemd/system/ldmsd.aggregator.service -> /opt/ovis/etc/systemd/system/ldmsd.aggregator.service
We then install the service file, via rpm install, to /opt/ovis/etc/systemd/system/ldmsd.aggregator.service

Run Sampler and Aggregator ldmsd using systemctl

- Run sampler and aggregator

```
$ systemctl start ldmsd.sampler
$ systemctl start ldmsd.aggregator
```

- Use ldms_ls to query aggregator for sets
- Modify the ldmsd configuration file specified in the ldmsd.sampler.env file to load and start an additional set
- Restart sampler and aggregator

```
$ system restart ldmsd.sampler
$ system restart ldmsd.aggregator
```

- Use ldms_ls to query aggregator for sets

END Advanced Part 2