MicroPMU Quickstart Kit

Administrator Manual

Revision 13

Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

Table of Contents

Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA

Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

Preface

Please take the time to read any bodies of text that begin with **IMPORTANT:** or **NOTE:** because they will contain either critical information to the task being explained or helpful information that may pertain to a particular situation or aid in troubleshooting.

IMPORTANT: It is **HIGHLY** recommended that the Quickstart Kit (server, MicroPMUs, network switch) is backed up by and uninterruptable power supply (UPS) in the event of a power surge or failure. This will greatly reduce the likelihood of damage caused to any of the hardware or potential data corruption as a result of a power outage or fluctuation. A recommended UPS model is the **APC-SMT750**.

I. Conventions Used in This Manual and Usage Notes

Command Line input and code samples will be displayed in a bold, monospaced (fixed width) font, and will be indented like so.

```
echo hello
```

Any command or example that can be executed on the command line by pressing the "Enter" key on the keyboard will end with a carriage return character like so.

```
echo hello ↵
```

In this example, the command to type is **`echo hello`**, then to execute the command, press the "Enter" key on the keyboard. If a series of commands are listed together, execute them in order from top to bottom. Command line keywords and file paths such as **`/home/manager`** may also be shown in-line.

Comments on command usage and results will be de-noted to the right of the command with either a '#' character for command line entries or two forward slashes, `//`, for comments on source code examples. Any text to the right of these characters is part of the comment and should not be included with the displayed command. Comments will also be in monospace font and not bolded. For example.

```
sudo supervisorctl start all ↵ # starts daemon services necessary for
operation of the Quickstart Server
```

The only part of the above command to actually type in the command line is **`sudo supervisorctl start all`**, followed by a press of the "Enter" key on the keyboard.

Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

II. List of Common Commands and Operations

sudo – Stands for "super user do". Any command prepended with **sudo** will be executed as the root user. Do not execute a command with **sudo** unless using a command from the manual that requires it. When using **sudo** to execute a command, a prompt for the **manager** user's password (**Company*2016!**) will appear.

Take special care when executing any commands that begin with **sudo**. The **sudo** keyword executes any command as the root user in UNIX based operating systems including Linux and Mac OSX. When a command is run as root, the operating system will most often execute it without question or hesitation which could lead to results anywhere from undesirable to disastrous if a command is entered incorrectly. In this manual, whenever asked to execute a command with **sudo**, triple check that it is typed exactly as seen in the manual before executing the command.

Logging in – Whenever faced with a login prompt, login with a username of **manager** and the default password of **Company*2016!** Unless the password has been manually changed.

logout – Use this command to logout of the current terminal session whether it is remotely over SSH or on the server itself.

ssh – Stands for "secure shell". The **ssh** commands allows for an encrypted remote terminal connection from a client computer to a remote server. For example, this command can be used to open a terminal session from a laptop to the Quickstart Server, so long as the laptop is on the same network as the server. The command format is as follows.

```
ssh user@host ↵
```

Where in the case of connecting from a laptop to the Quickstart Server, user is **manager** and host is **172.16.1.100**. A prompt to enter the password for the user account on the remote server will also appear.

cd – Stands for "change directory". This command is used to change the current working directory. For example, upon logging into the manager account the working directory will be **/home/manager**. The following command would change the working directory to **/home/manager/mr-plotter**.

```
cd mr-plotter/ ↵
```

If the **cd** command is executed without any parameters, the working directory will be changed to the home directory of the current user.

```
cd ↵ # changes current working directory to /home/manager
```

ls – The **ls** command will list the contents of the current working directory if no parameters are passed to the command. Otherwise, **ls** will list the contents of any valid file path passed to it.

```
ls ↵ # lists contents of the current working directory
ls /var/log ↵ # lists contents of the /var/log directory
```

mkdir – Use this command to create a new directory. To make more than one new directory where each new directory is a child to the previous directory, use the **–p** option. For example.

```
mkdir –p parent/child/grandchild ↵ # this command would create the
parent/child/grandchild directory in the present working directory
and the –p option will create the parents of grandchild directory if
they do not exist
```

rm – Use this command to delete files. The **–r** option will recursively delete all files in the given path and the **–f** option will force the deletion without confirming for certain files such as write protected files. **Be extremely cautious when executing this command because there is no going back once it is done.**

```
rm test.txt ↵ # removes the file names 'test.txt'
rm –rf test/data ↵ # removes all files under the data directory
including the data directory
```

shutdown – Use this command to safely shut down the Quickstart Server like in the example below.

```
shutdown now ↵
```

reboot – Use this command to safely reboot the Quickstart Server like in the example below.

```
reboot now ↵
```

Backup and Restore Database – The following is a summary of the database backup and restore procedure for the Quickstart Server.

IMPORTANT: If not completely familiar with the database backup and restore procedure and its necessary precautions, please take the time to review and understand the details of the database backup and restore procedure in chapter 6.

Backup – If there is an external drive, with enough space to hold the backup data, mounted to the **/media/external** directory, a backup can be done with the following commands.

```
sudo supervisorctl stop all ↵ # stop services to avoid losing incoming
data during the backup
sudo cp –r /ssd/data /media/external ↵
```

```
sudo cp –r /ssd/srv /media/external ↲
```

Restore – If there is an external drive with a copy of the contents of the **/ssd/data** and **/ssd/srv** directories, and it is mounted to the **/media/external** directory, a restore can be done with the following commands.

```
sudo supervisorctl stop all ↲
sudo cp –r /media/external/data /ssd ↲
sudo cp –r /media/external/srv /ssd ↲
```


Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

1 Quickstart Server Startup/Shutdown

IMPORTANT: It is **HIGHLY** recommended that the Quickstart Kit (server, MicroPMUs, network switch) is backed up by and uninterruptable power supply (UPS) in the event of a power surge or failure. This will greatly reduce the likelihood of damage caused to any of the hardware or potential data corruption as a result of a power outage or fluctuation. A recommended UPS model is the **APC-SMT750**.

Before getting started with the MicroPMU setup, the Quickstart Server needs to be powered up. To power your server, first make sure it is plugged into a suitable power supply and then press the power button. It is located at the top left corner of the server's front panel, directly to the right of the Dell logo.

IMPORTANT: Please note that this button should not be used to force a server shutdown, nor should the power cable be unplugged while the server is powered on. Please refer to the next section on how to properly shutdown your server.

Upon pressing the power button, there should be audible and visible cues that the server began the boot process, including the power button turning green. While the server is booting, some options may be available for the F1, F2, F3 keys, etc. Do not press any of these keys, and let the server continue to boot until the login prompt is displayed.

1.1 Quickstart Server Login

Once the server is ready the prompt will show…

```
quickstart login:
```

The default Quickstart Server user is: **manager**

Type manager for the user and press the "Enter" key ↵ on the keyboard, then the prompt will display the following.

```
Password:
```

The default Quickstart Server password is: **Company*2016!**

Type the password and press the "Enter" key ↵ on the keyboard. Upon a successful login the prompt will display the following.

```
manager@quickstart:~$
```

1.2 Quickstart Server Logout

To log out of a user account or ssh session, simply type logout in the command prompt and press the enter key.

```
logout ↲
```

1.3 Quickstart Server Shutdown

It's important to try to gracefully shutdown the Quickstart Server whenever a shutdown is required. To do this, first make sure manager is logged in, then type shutdown now into the command prompt and press the "Enter" key.

```
shutdown now ↲
```

2 Quickstart Server Remote Login Using SSH

SSH allows logins to a remote computer/server over an encrypted connection. To SSH into the Quickstart Server, either an SSH client for Windows (we recommend PuTTY) is required, or the terminal can be used on a Mac or Linux computer.

2.1 Finding Your Quickstart Server IP

Before attempting to establish an SSH connection to your Quickstart Server, the server IP address must be noted.

By default, the Quickstart Server IP address is: 172.16.1.100

To find or double check the IP, login as **`manager`** on the server (using a monitor and keyboard plugged into the server). Enter the following command to display the server IP.

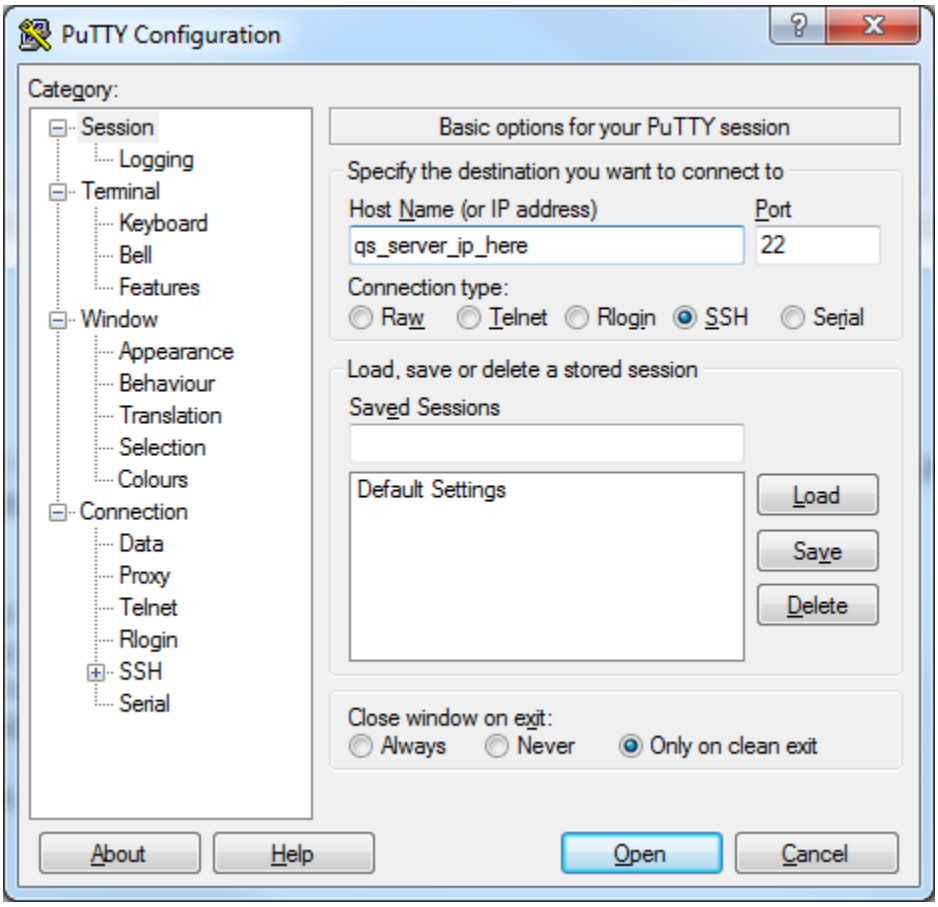
```
ifconfig | grep "inet addr" ↲
```

Make sure to type the command exactly as seen and at least two lines of text that start with "inet addr" should be printed on the screen. Take note of the "inet addr" on the line that also displays "Bcast" and "Mask", this is the server IP address.

2.2 SSH Using Windows

To SSH from a Windows computer, a third party SSH client will need to be installed. We recommend using PuTTY as it is reliable, safe, easy to use and free! If PuTTY is not downloaded already, the program may be downloaded from www.putty.org.

Open PuTTY and the PuTTY Configuration window should be displayed. To start an SSH connection enter the Quickstart Server's IP address as well as make sure the port is set to 22 and the connection type is set to SSH.



This configuration can be saved by typing in a configuration name under "Saved Sessions" and click the save button. From there, just highlight the configuration name and click the load button then click the open button to start a connection.

A prompt may ask if host to connect to is trusted. Type and enter "yes" to finish the connection process. Once connected, proceed to login the same way as on the server itself (Login as: manager Password: Company*2016!).

2.3 SSH Using Mac OSX

To SSH from a Mac, there is no need to use third party software. Simply open a terminal window which can be found by using the spotlight search (press "command" + "space" or click magnifying glass icon

toward the top right corner of your screen) and typing “terminal”. Enter the following command in the terminal to establish an SSH connection with the Quickstart Server.

ssh manager@quick-start-server-ip ↵

Example:

ssh manager@172.16.1.100 ↵

A prompt may ask if host to connect to is trusted. Type and enter “yes” to finish the connection process. Once connected, proceed to login the same way as on the server itself (Password: Company*2016!).

3 Managing MicroPMUs with upmu-adm

Each Quickstart Server ships preconfigured to communicate and accept data streams from the MicroPMUs that come with the Quickstart Kit. In the event that these MicroPMUs need to be added again, removed, edited or perhaps additional MicroPMUs need to be added to the server, the **upmu-adm** script may be used to fulfill these needs.

3.1 Adding MicroPMUs with upmu-adm

To add a MicroPMU, enter the following commands:

```
cd ↵ # to go to the home directory if not already there

upmu-adm ↵ # run upmu-administration panel

add ipaddr serial alias [ext] ↵ # upmu ip address, upmu serial number,
upmu alias name, optional extended mode
```

Note that any name can be used for upmu alias. If the **ext** parameter is not included, .dat files will default to base mode.

To add a MicroPMU in base mode, simply enter the **add** command without the **ext** parameter.

Example:

add 192.168.1.24 P3112233 uPMU1 ↵

To add a MicroPMU in extended mode, enter **true** at the end of the **add** command.

Example:

add 192.168.1.24 P3112233 uPMU1 true ↵

Then use the following command to double check the new entries.

```
list ↲ # view list of MicroPMUs, their IP address, extended mode
status and aliases to confirm changes
```

After confirming the entries are correct, the save command must be used to apply the changes to the MicroPMU configuration file.

```
save ↲  # this command must be to write to the upmuconfig.ini file or
else any changes will not be saved

Ctrl+C # press the "Control" and "c" keys on the keyboard at the same
time to exit upmu-adm
```

Upon a successful exit, the command prompt should print a message like "clean exit – no unsaved changes".

Next, the metadata database must be updated:

```
cd ↲ # redirects to the home directory

python manager2lite.ipy ↲ # adds metadata or updates existing plotter
metadata
```

Status messages will confirm that existing metadata have been added, removed, or updated.

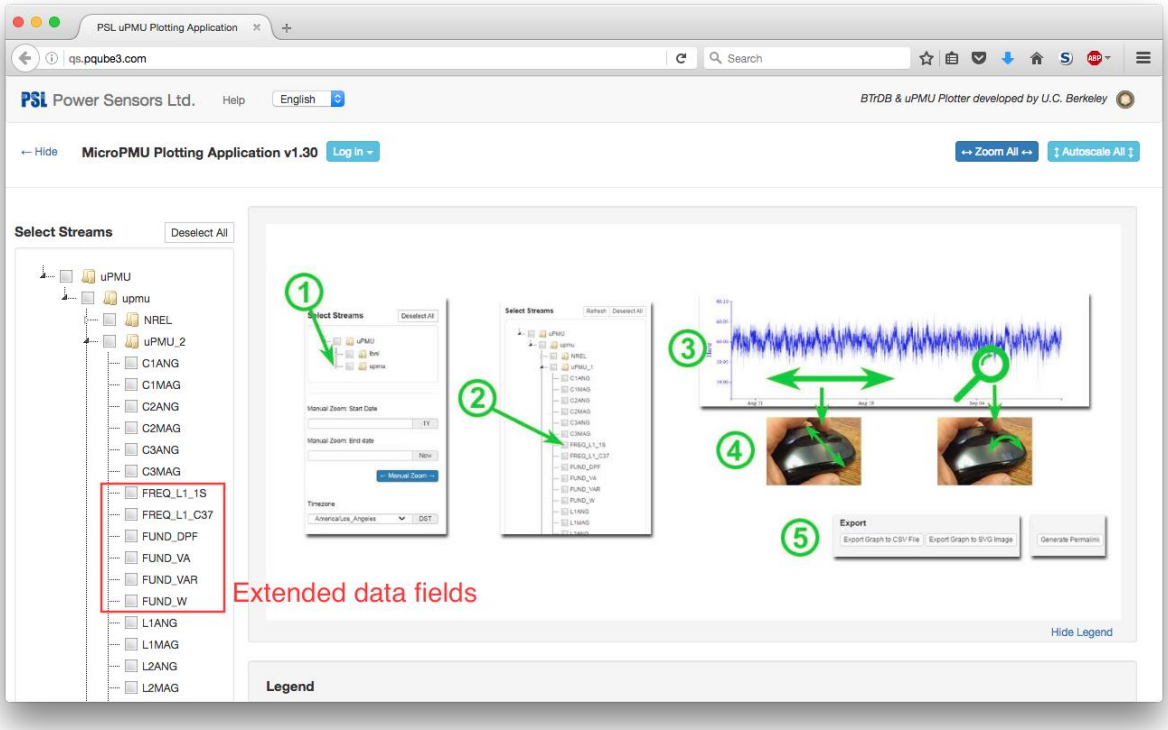
Finally, restart services:

```
sudo supervisorctl restart all ↲ # enter manager password when
prompted
```

Note the services will take up to 30 seconds or less to stop and start again. Any "spawn error" messages from this command may be ignored.

3.2 Switching Between Base and Extended Mode

MicroPMUs have two .dat file modes: base and extended. The difference between these two modes is the amount of data fields that are recorded. The extended .dat file mode will present frequency, apparent power, real power, and power factor data in addition to the base data fields as can be seen in the screenshot below of the MicroPMU Plotter website. Base mode will generate files around 379KB in size and extended mode will generate files around 552KB in size.



To switch between base and extended mode, first make sure the **upmu-adm** shell is running. Enter the following command to switch the .dat file mode (true=extended, false=base).

setext ipadrr [true | false] ↲

Example:

setext 192.168.1.101 true ↲

3.3 Removing MicroPMUs and MicroPMU Data

To remove an existing MicroPMU, the metadata database will also need to be updated.

First, login to the Quickstart Server as manager, then enter the following commands.

cd ↲ # switches to home directory if not already there

upmu-adm ↲ # starts MicroPMU administration panel

list ↲ # view list of MicroPMUs, their IP address, serial number, extended mode status and aliases

```
rem [IP | Serial | Alias] ↲ # remove the MicroPMU by specifying one of
the following: IP Address, Serial Number, Alias
```

Example:

```
rem 192.168.1.24 ↲ # remove MicroPMU with the entered IP address
rem P3001234 ↲ # remove MicroPMU with the entered serial number
rem uPMU1 ↲ # remove MicroPMU with the entered alias
```

```
save ↲ # save changes
Ctrl+C # exit
```

Upon a successful exit, the command prompt should print a message like “clean exit – no unsaved changes”.

```
cd ↲ # redirects to the home directory
python manager2lite.ipy ↲ # removes metadata or updates existing
plotter metadata
```

A message will print stating that metadata is being removed for the MicroPMU chosen to remove along with processing messages for any other MicroPMUs available.

Finally, restart services:

```
sudo supervisorctl restart all ↲ # enter manager password when
prompted
```

Note the services will take up to 30 seconds or less to stop and start again. Any “spawn error” messages from this command may be ignored.

4 Importing Existing .dat Files

Existing data files can be imported to **plotter** by using **`upmu-adm`** command-line interface.

NOTE: The MicroPMU that the .dat files belong to must be added to the plotter instance on the server the data will be imported to. For example, say data files will be imported for a MicroPMU with a serial number of P3001234. If a MicroPMU with this serial number has not been added by using the **`upmu-adm`** program, then it will need to be added first. Please refer to the “Adding MicroPMUs” section for instructions on adding a MicroPMU to the plotter.

4.1 Copying Data Files to The Server

First the data files need to be present on the Quickstart Server. If there are already data files on the server then proceed to the importing data step.

4.1.1 Copy Data Files via SCP

If data files are stored on a remote computer then the **scp** command can be used to move them to the server (Linux and Mac OSX only). Windows can use a similar tool called pscp.exe.

First create a directory on the server to copy the data files into.

```
cd ↲ # redirects to the current user's home directory
mkdir data-import ↲
```

Now, from the remote computer, identify the full path of the directory you want to import and then use the following command on the command line of the remote computer. Note that this example is all one command with a space in between the "**scp -r /path**" section and the "**manager@IPAddress:/path**" section of the command.

```
scp -r /path/to/folder/containing/data/files
manager@172.16.1.100:/home/manager/data-import/ ↲ # replace example
IP with the Quickstart Server IP and use a space between the local
path and the Quickstart Server path
```

4.1.2 Copy Data Files via USB/External Drive

Alternatively, a FAT32 formatted USB or external drive may be plugged directly into the server to copy the .dat files.

Plug the external drive into the Quickstart Server via USB port.

NOTE: After plugging in the external drive, a message may appear displaying something to the effect of "No Caching mode page found…Assuming drive cache: write through". Ignore this message and use **Ctrl+c** in order to make the command prompt appear again.

Now use the following command to list the available drives that the server sees.

```
sudo fdisk -l | grep sd ↲
```

In the list should be some entries for "Disk" that will list a path such as **/dev/sda/** or **/dev/sdb/**. Those are the two drives installed in the server. If there are no other drives installed in the server then the external drive will likely be listed as **/dev/sdc/**. Confirm which drive is the external drive by

checking the size listed as well as the type, which should include FAT32. Underneath the “Disk” line should be a list of partitions for the drive such as **/dev/sdc1**, **/dev/sdc2**, etc. depending on how many partitions (independent file systems) are on the drive which will most likely just be one partition unless explicitly formatted otherwise.

Once the path of the external drive is found, mount the drive with the following commands (assuming the drive is located at **/dev/sdc/** and only has one partition).

Example:

```
sudo mkdir /media/external ↲ # if the directory does not already exist

mount -t vfat /dev/sdc1 /media/external ↲
```

4.2 Importing Data

Now everything is staged to begin the data import.

NOTE: This process is not recursive, so all files to import must in the same directory.

To access **importdat**, type the following commands:

```
cd ↲ # go to home directory

upmu-adm ↲ # to run upmu-administration panel

importdat FULL-PATH-TO-DIRECTORY ↲ # where "FULL_PATH_TO_DIRECTORY"
is where the .dat files are located. Copy ALL .dat files to import
into the directory. The process is not recursive, so all files must be
in the directory. For help, type importdat with no arguments for usage
instructions or help to show all commands
```

Example (if data is stored in a “data” folder on the external drive):

```
importdat /media/external/data ↲
```

Then:

```
Ctrl+C # exit
```

Upon a successful exit, the command prompt should print a message like “clean exit – no unsaved changes”.

Once finished with a data import, unmount the external drive with the following command.

```
sudo umount /media/external ↲
```

Afterwards, wait up to three minutes to let the **sync2_quasar** service automatically add the data to the plotter instance. Remember that files that have been imported are either in "base" or "extended" mode.

5 Backing Up And Restoring MicroPMU Databases

The **/ssd/data/** and **/ssd/srv/** directories may become quite large, and may be backed up periodically either remotely or by physically backing up the storage drive. Databases can be manually backed up by copying the entire **/ssd/data/** and **/ssd/srv/** directories to a safe location, but first, please read the tips and instructions below to ensure the backup will be done safely.

5.1 Important Notes for A Database Backup/Restore

It is best practice to save the directories while there are no services running. This way, all data leading up to the backup date will be preserved when the files are copied back onto the new drive, and queued data will begin streaming in from MicroPMUs.

IMPORTANT: Do not remove the first drive, which contains the operating system and plotter settings. Remove the secondary storage drive from the server and mirror its contents to the new drive.

IMPORTANT: never hot unplug the Quickstart Server while data is being synchronized or written.

IMPORTANT: make sure to stop all services during backup so new data isn't being written:

```
sudo supervisorctl stop all ↲
```

NOTE: Hooks to the operating system exist to safely shut down scripts on graceful reboot or shutdown. The following commands are **SAFE** for powering down the server to do a physical drive backup.

```
sudo reboot now ↲
sudo shutdown now ↲
```

5.2 Example Backup/Restore

As mentioned earlier, the databases can be backed up by copying the contents of **/ssd/data/** and **/ssd/srv/** to a safe location, by mirroring the physical drives, or by copying the drive contents over a network connection to a remote drive.

The most reliable, safest and simplest means of backing up the data is to directly connect/plug in an external drive to the Quickstart Server and copy the data to that drive. To do this, start by acquiring an

external drive formatted to FAT32 with enough space to fit the contents of the data drive (**/ssd/data/** and **/ssd/srv/**). Refer to the plotter website to see how much disk space is currently occupied on the server.

Plug the external drive into the Quickstart Server via USB port.

NOTE: After plugging in the external drive, if a message appears displaying something to the effect of "No Caching mode page found…Assuming drive cache: write through", ignore this message and use **Ctrl+c** in order to make the command prompt appear again.

Now use the following command to list the available drives that the server sees.

```
sudo fdisk –l | grep sd ↲
```

In the list should be some entries for "Disk" that will list a path such as **/dev/sda/** or **/dev/sdb/**. Those are the two drives installed in the server. If there are no other drives installed in the server then the external drive will likely be listed as **/dev/sdc/**. Confirm which drive is the external drive by checking the size listed as well as the type, which should include FAT32. Underneath the "Disk" line should be a list of partitions for the drive such as /dev/sdc1, /dev/sdc2, etc. depending on how many partitions (independent file systems) are on the drive which will most likely just be one partition unless explicitly formatted otherwise.

Once the path of the external drive is found, mount the drive with the following commands (assuming the drive is located at **/dev/sdc/** and only has one partition).

```
sudo mkdir /media/external ↲
sudo mount –t vfat /dev/sdc1 /media/external ↲
```

Now everything is staged to begin the backup, but first remember to stop all services!

```
sudo supervisorctl stop all ↲
```

Now backup the data with the following commands.

```
sudo cp –r /ssd/data /media/external ↲
sudo cp –r /ssd/srv /media/external ↲
```

Note that data backups can be restored to the server by switching the placement in the paths like so. Make sure the ssd directory is empty before copying files to it.

```
sudo cp –r /media/external/data /ssd ↲
sudo cp –r /media/external/srv /ssd ↲
```

Once finished with a backup or restore, unmount the external drive with the following command.

```
sudo umount /media/external ↲
```

Now services may be started back up.

```
sudo supervisorctl start all ↲
```

Note the services will take up to 30 seconds or less to start again. Any “spawn error” messages from this command may be ignored.

6 Deleting Data and Starting Fresh with a New Instance

Take note before deleting database data:

IMPORTANT: Remember to copy the entire **/ssd/data/** and **/ssd/srv/** directories and **`upmuconfig.ini`** to a safe location in case you need to restore it to an old state.

IMPORTANT: never hot unplug the Quickstart Server while data is being synchronized.

IMPORTANT: make sure to stop all services with the following command:

```
sudo supervisorctl stop all ↲
```

The file-based database can be deleted to start a new instance by executing the following command.

```
sudo newdb
```

A prompt will appear confirming whether to continue with deletion and recreation of the Quickstart Server database. Type “y” or “Y” on the keyboard to continue. Any other key will cancel the operation.

New MicroPMUs may now be added. Refer to the section that covers the process for adding MicroPMUs and go through all the steps to properly add the MicroPMUs.

7 Setting a New Quickstart Server IP Address

Open the network interfaces file for editing with the following command.

```
sudo nano /etc/network/interfaces ↲ # Opens the file in the nano text
editor. Alternatively, the vi/vim text editor may be used. Nano is
recommend as a more user friendly text editor if it is unclear which
```

```
one to use.
```

Edit the highlighted sections with the desired static settings:

```
auto eno1
iface eno1 inet static
      address 172.16.1.100
      netmask 255.255.255.0
      gateway 172.16.1.1
      #dns-nameservers XXX.XXX.XXX.XXX
      #dns-nameservers XXX.XXX.XXX.XXX
```

(DNS nameservers are optional, so they are by default commented out. Nameservers are used for assigning domain names to IP Addresses).

Save changes in **nano** with **Ctrl+x** then type in the letter “y”.

Restart the network interface with the following command.

```
sudo ifdown eno1 && sudo ifup eno1 ↲
```

NOTE: Automatic dhcp settings should not be used because MicroPMUs need a static IP to send the data to. The dynamic settings are available in the network configuration file if a dynamic IP address needs to be assigned to the server.

8 Routing MicroPMU data to the Quickstart Server

The default IP Address for the Quickstart Server is 172.16.1.100
If this has been changed, the MicroPMU setup.ini files will need to be updated with the new IP address.
Change this variable in the setup.ini file of each MicroPMU that will send data to the Quickstart Server:

```
;-----------------------------------------------------
[microPMU_Database_Server_Settings]
;-----------------------------------------------------
; ------ If you have a microPMU database server, enable this setting
; ------ to automatically push data from your microPMU to the server.
; ------ Valid values: ON, OFF
Enable_Push_Data_to_Database_Server=ON
Database_Server_IP_Address=172.16.1.100
```

Once complete, restart the MicroPMU for changes to take effect. Make sure that the MicroPMUs are on the same domain as the Quickstart Server if they are on a closed network with the provided Ethernet switch.

```
;-----------------------------------------------------
```

```
[Network_Setup]
;-----------------------------------------------------
; ------ Valid Values: Use_DHCP, Use_Fixed_IP
IP_Address_Method=Use_Fixed_IP
Publish_IP_Address=ON

;-----------------------------------------------------
[Fixed_IP]
;-----------------------------------------------------
; ------ This section is ignored if the IP_Address_Method is set to
Use_DHCP
IP_Address=172.16.1.101
IP_Gateway=172.16.1.1
IP_Mask=255.255.255.0
IP_DNS1=8.8.8.8
IP_DNS2=8.8.4.4
```

NOTE: The serial number of each MicroPMU **MUST MATCH** the serial number entered in the **upmuconfig.ini** file (this can be checked with **upmu-adm's list command** or view the contents of **upmuconfig.ini** in a text editor). If the serial number ever changes, update this in **upmuconfig.ini**. The **receiver** process validates and sorts data by serial number, so it will reject data from unknown serial numbers.

9 Changing the Root User Password

By default, the username for the MicroPMU Quickstart Server is manager. You are also provided with a default manager/root user password, although it is recommended to change it for increased security. To change it, type:

```
sudo passwd ↵
```

From there, follow the instructions to change the root password. This should mean entering the existing password, followed by the new password twice. Also note that the password will not be displayed as it is entered so be sure to type with precision. Be sure to take note of the new password and record it in a safe place!

10 Changing MicroPMU Domains for Access Control

Edit **/home/manager/upmuconfig.ini** and change the domain of the MicroPMUs to a separate domain (for example from **upmu** to **lbnl**). In **upmuconfig.ini**, find the **Path** fields for the MicroPMUs to assign to new domain(s). Each **Path** field for each unit will need to be changed.

```
Path=/upmu/upmu_name/stream_name
```

becomes:

```
Path=/lbnl/upmu_name/stream_name
```

After saving the changes to **/home/manager/upmuconfig.ini**, re-run **manager2lite.ipy**

Changes will not show until made public. The domain needs to be made public by editing **/home/manager/mr-plotter/tools/tagconfig.json** and assigning the domain name (ex. **lbnl**).

All units in a domain can be made public by using an open ended domain like **/upmu** or **/lbnl** or a combination of devices from different domains. The existing file has **"public": ["/upmu"]**, so all devices in **upmu** domain are open to the public by default. To make all domains public, make **"public": ["/"].**

Syntax for singular units made public:

```
{
      "public": ["/lbnl/alameda_conference1","/upmu/san_diego"],
}
```

Syntax for multiple units in a domain made public:

```
{
      "public": ["/lbnl", "/upmu"],
}
```

Syntax for making everything public:

```
{
      "public": ["/"],
}
```

Next, restart the metadata process with:

```
sudo supervisorctl restart metadata
```

Two domains should be seen on the Quickstart Server's "Select Streams" when you access its IP address in a browser. The next section explains how to add user accounts and assign domain tags to them.

Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

11 Add User Accounts/Assign Them "Domain" Tags

Add user accounts by running **`python accounts.py`**. The **`accounts.py`** file is located in the **`/mr-plotter/tools`** directory. The Mr. Plotter account access control console will open. Type **`help`** for additional clarification.

```
cd /home/manager/mr-plotter/tools && python accounts.py ↲
```

Usage:

```
adduser username password tags…
```

ex:

```
adduser peter psladmin psl lbnl pge ↲
exit ↲
```

The user "peter" has password "psladmin" and can access "psl", "lbnl", and "pge" domains. At least one tag is required when using **`adduser`** command.

Add additional "domain" tags to the end of the statement if user requires multiple access. All users by default can view what's tagged "public" in **tagconfig.json** (**`cd /home/manager/mr-plotter/tools`**).

The tags are the same domain tags that were updated in **upmuconfig.ini** (in the previous section), so "upmu", "lbnl" or "psl" for instance. Next, edit **tagconfig.json** to add the tag that is assigned to the user and remove the access-controlled streams from public domain. Public can be an empty array if no MicroPMUs are to be made public.

The following example makes the **`/upmu`** domain public while only users with **`/lbnl`** domain access (**`adduser`** command) can view **`/lbnl`** MicroPMUs.

```
{
      "public": ["/upmu"],
      "lbnl": ["/lbnl"]
}
```

Run **`manager2lite.ipy`** and then restart services:

```
cd ↲
pythonmanager2lite.ipy ↲ # adds metadata or updates existing plotter metadata
sudo supervisorctl restart all ↲
```

Example:

1. Change the path domains for unit “uPMU_1” to **`/lbnl`** (from **`/upmu`**) in **`upmuconfig.ini`**
2. Run **`python accounts.py`** from **`/home/manager/mr-plotter/tools`** directory and **`adduser`** command.
3. Update **`tagconfig.json`** domains to reflect new “domain tags” and remove tags from public domain
4. Run **`python manager2lite.ipy`** and restart services with **`sudo supervisorctl restart all`**.
5. Domains now appear in "Select Streams" section. When logged in as a user with **`/lbnl`** domain permissions, the user will be able to view **`/lbnl`** domain streams. It is no longer publicly available as specified in **`tagconfig.json`**, unless streams are left public for demo purposes.

12 Clear Up Space by Deleting Synchronized Data

Delete files that have been synchronized to free up disk space using:

```
cd ↵
clean_synced ↵
```

Run this file, which will tell you how many files can be deleted. Type **`delete`** to verify data removal.

13 BTrDB Firewall (Shorewall) Reference

This guide covers step-by-step procedures for securing an Ubuntu 16.04 server running BTrDB using Shorewall following the BTrDB Firewall Reference. Shorewall comes installed on PSL Quickstart Server, so installation instructions are for reference.

Double check that Shorewall is installed with the following command.

```
sudo shorewall status ↵
```

If there is a line text in the returned message that shows “Shorewall is running”, then the installation section may be skipped.

13.1 Shorewall Installation

Please ensure that the system software is up-to-date as security vulnerabilities to exposed services cannot be prevented by a firewall.

```
sudo apt-get update ↲
sudo apt-get dist-upgrade –y ↲
```

Then install the shorewall package.

```
sudo apt-get install -y shorewall ↲
```

Shorewall is disabled and unconfigured by default, copy the default configuration files and modify them.

```
cd /usr/share/doc/shorewall/examples/one-interface/ ↲
sudo cp rules interfaces zones policy /etc/shorewall/ ↲
```

13.2 Interfaces

The first step is to ensure that the interface described in **/etc/shorewall/interfaces** is correct. To see interfaces, run the following command.

```
Ifconfig ↲
```

The interface will have a name like **eno1** or **eth0**. Edit **/etc/shorewall/interfaces** so that the name matches. If the interface is configured statically, also remove the 'dhcp' from the interface flags. For reference, our file looks like this.

```
#ZONE       INTERFACE          OPTIONS
net         eno1               tcpflags
```

13.3 Rules

The default policy (deny all) and zones files are acceptable. It is necessary to modify the rules file to permit traffic to the BTrDB server. Add the following lines to **/etc/shorewall/rules**

```
# Allow access to the plotter
ACCEPT          net         $FW         tcp         80,443
# Allow uPMU data
```

```
ACCEPT          net         $FW         tcp         1883
# Allow access to the BTrDB API
ACCEPT          net         $FW         tcp         4410,9000
```

Note that the default shorewall rules file denies ICMP Pings to the server. You may want to remove this line.

```
Ping(DROP)      net         $FW
```

And replace it with

```
Ping(ACCEPT)    net         $FW
```

In addition, you may want to permit SSH access with

```
ACCEPT          net         $FW         tcp         22
```

13.4 Auto start

To ensure the firewall starts on boot, enable it in **`/etc/default/shorewall`**. Open that file and change the following line from

```
startup=0
```

To

```
startup=1
```

so that the firewall will start on boot.

Finally, add shorewall to systemctl with the following command.

```
sudo systemctl enable shorewall ↲
```

NOTE: If any key service updates were installed in the dist-upgrade step earlier, or a kernel update was installed, it is best to reboot the server so that these take effect.

```
sudo reboot ↲
```

If there were no key updates, simply start the firewall without rebooting the server with the following command.

```
sudo shorewall safe-restart ↲
```

14 Using the Quickstart Server Package Manager: qss-get

The Quickstart Server is designed to run on a closed local network with no connection to the internet. Because of this, installations and updates for the Quickstart Server dependencies must be done with a local install or update archive downloaded from the PSL website.

Install archive names have a prefix of “qss-install” followed by its version number and the file extension, e.g. “qss-install-1.4.0.tar.gz”. Update archives have the same naming convention, but with a prefix of “qss-update”.

In order to make use of these archives, the **qss-get** script must be executed. The following is an outline of how to use the **qss-get** script.

usage : qss-get command [args...] [options...]

Quickstart Server package manager that is used to install and update Quickstart Server dependencies via local resource archives. For the install, update and packages command, the archive path must be the first argument passed if the local archive is not in the present working directory.

-i --install [archive_path] [-a] [--ups]	Performs a fresh install of the Quickstart Server. Defaults to present working directory if path to qss-install ARCHIVE is not given. Use the -a option to include all optional packages. View the package list below for required and optional packages. Use the -ups option to include the apcupsd package if using using an uninterupptable power supply.
-u --update [archive_path]	Updates Quickstart Server repositories. Defaults to present

	working directory if path to qss-update ARCHIVE is not given.
-p --packages [archive_path] packages... [-a]	Install or update package dependencies for the Quickstart Server. Defaults to present working directory if path to qss-packages ARCHIVE is not given. It is required to pass at least one valid package name from the package list below in a list separated by spaces, unless the -a option is used, which will install all available packages.
-h –help	Displays this usage outline. Will also appear in the event of an invalid command execution.

Quickstart Server Package Dependencies

Required:

golang – The "Go" open source programming language developed by Google and open source contributors. Visit golang.org for more information.

librados-dev – Provides API for creating a ceph storage cluster interface. Required to build the Quickstart Server binaries.

mongodb – Mongo database server used to store raw data streaming from the MicroPMU as well as account and permalink data for MR-Plotter.

python – programming language needed to run several utility scripts on the Quickstart Server.

ipython – Command shell for interactive computing.

python-pip – Python package/module installer.

configobj – Python module used for reading configuration files.

pymongo – Python interface to mongodb.

requests – Python module for network requests.

meld3 – Python module for markup templating.

supervisor – Python module for controlling a set of processes. In the case of the Quickstart Server, supervisor is used to control the btrdb, mongo, mr-plotter, receiver, and sync2q processes.

six – Python module used to help write python2 and python3 compatible code.

shorewall – User friendly interface to iptables configuration.

Optional:

apcupsd – Daemon program used for management of uninterruptible power supplies.

14.1 Quickstart Server Fresh Installation

Each Quickstart Server ships with a fresh installation of the latest version of the Quickstart Server software. Although it is unlikely to have to perform another fresh installation on a Quickstart Server, the following is an example of how to use **qss-get** to perform a fresh installation on a vanilla Ubuntu Server 16.04 LTS with open SSH installed.

```
sudo qss-get –i /mnt ↲
```

In the above example, the **–i** flag is used to call the install command and the **/mnt** argument indicates that the **qss-install** archive file is located in the **/mnt** directory. Otherwise the script will search for the archive in the present working directory. Upon pressing the “Enter” key on the keyboard, the install will begin. Once the installation is finished, a prompt will appear to press any key to initiate a system reboot in order to finalize the installation.

14.2 Quickstart Server Update

The most common use for the **qss-get** script is to update the Quickstart Server packages and repositories. To perform an update, run the **qss-get** script with the update flag and pass the path to the **qss-update** archive file if it is not located in the present working directory, such as in the following example.

```
sudo qss-get –u /mnt ↲
```

In the above example, the **–u** flag is used to call the update command and the **/mnt** argument indicates that the **qss-update** archive file is located in the **/mnt** directory. Otherwise the script will search for the archive in the present working directory. Upon pressing the “Enter” key on the keyboard,

the update will begin and no further action is required. A message will indicate when the update is complete.

14.3 Quickstart Server Packages

The **qss-get** script may also be used to grab packages from a **qss-packages** archive file. This is useful to install any optional packages that may not already be installed. For example, if the server is first used without a UPS backup, and a UPS backup is introduced at a later date then the **apcupsd** package may be individually installed by passing the update flag to the **qss-get** script as well as by passing "apcupsd" as an argument to the update command. The following is an example of such a scenario.

```
sudo qss-get -p /mnt apcupsd ↵
```

In the above example, the **-p** flag is used to call the packages command and the **/mnt** argument indicates that the **qss-packages** archive file is located in the **/mnt** directory. Otherwise the script will search for the archive in the present working directory. Upon pressing the "Enter" key on the keyboard, the install or update of the specified package(s) will begin and no further action is required. A message will indicate when the task is complete.

The packages command may also be useful in the event that specific packages need to be reinstalled or updated.

15 Using a UPS Backup

IMPORTANT: It is **HIGHLY** recommended that the Quickstart Kit (server, MicroPMUs, network switch) is backed up by and uninterruptable power supply (UPS) in the event of a power surge or failure. This will greatly reduce the likelihood of damage caused to any of the hardware or potential data corruption as a result of a power outage or fluctuation. A recommended UPS model is the **APC-SMT750**.

Hopefully the above message has become very clear by now because using a UPS backup is critical to maintaining the stability and health of any running server. Support is included for the recommended **APC-SMT750** uninterruptable power supply and the following instructions refer only to this model UPS.

Each Quickstart Server comes with APC's UPS management software for Linux based systems, called **apcupsd**. However the software configuration is not enabled by default, but is easily enabled with an execution of the **qss-get** script. First, please ensure the UPS is properly installed.

IMPORTANT: The following steps outline the steps for proper installation of the **APC-SMT750** UPS. Please read through the official installation guide included with the UPS if not familiar with installation of uninterruptable power supplies.

---

1) Before plugging the UPS into a power supply, make sure to insert the battery connector into the battery jack.
2) Plug the power cords of the Quickstart Server and network switch into the UPS
3) Plug the UPS into a grounded outlet
4) Connect the UPS to the Quickstart Server via the USB cable provided with the UPS. The type B (square looking) connector goes into the UPS and the type A (thinner rectangular looking) connector goes into a USB port on the Quickstart Server.
5) Power on the UPS by holding the power button. There will be a first beep, and shortly after, a second beep. Let go of the power button after the second beep.

With the UPS installed, run the following command to configure the **apcupsd** software.

```
sudo qss-get –configure apcupsd ↵
```

Once the above command is executed, the UPS will be configured to shut down the Quickstart Server after 30 seconds of running on battery.

16 Troubleshooting

Symptom:

No data is shown on the plotter.

Solution #1:

Data must be sent from MicroPMUs to **receiver**, which saves them to a queue for **sync2_quasar** services. Quasar is the database system. To check if **sync2_quasar** and **receiver** are running as expected, use the following commands.

```
sudo supervisorctl tail -f receiver ↲
Ctrl+c # exit tail program
sudo supervisorctl tail -f sync2_quasar ↲
Ctrl+c # exit tail program
```

Symptom:

Data is not being sent to the plotter.

Solution #1:

Check if receiver is running on the server. By default, it should constantly be receiving data from MicroPMUs.

```
sudo supervisorctl tail -f receiver ↲
Ctrl+C # exit tail program
```

Solution #2:

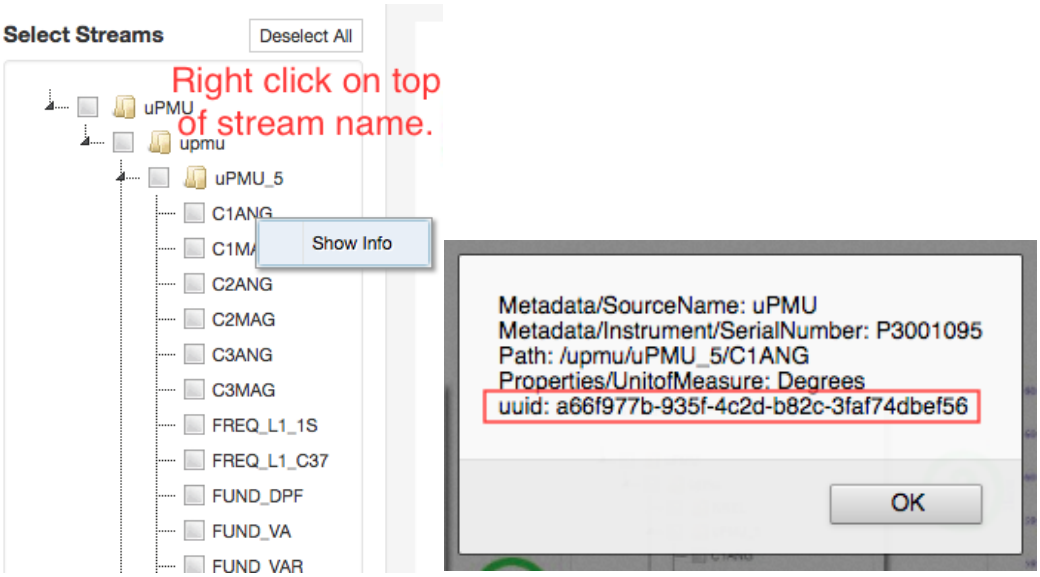
The transfer agent may have stopped running due to prolonged server outage. It can be restarted by restarting the MicroPMU. Also make sure the MicroPMUs are sending data to the Quick Start server. Refer to “Routing MicroPMU data to the Quickstart Server” for data routing instructions.

17 Appendices

17.1 BTrDB REST API Examples

BTrDB includes a REST api that can be used from any language supporting HTTP. This guide will explain how to do some common operations using curl which is a command line program, usually available by default on MacOSX (unless using a very old version of OSX), but can also be installed on Windows. Bear in mind that the binary API is faster, and has golang and python support.

BTrDB stores data in streams identified by a UUID (Universally Unique Identifier) which can be found on the plotter website by right clicking on a stream name and clicking “show info”.



17.1.1 jq Installation (optional)

The BTrDB REST API returns data in JSON (javascript object notation) format. While not necessary, it is recommended to install jq which will parse the JSON data into a more readable format when API requests are made. The jq binaries can be downloaded from https://stedolan.github.io/jq/ for Windows, Linux or Mac OSX.

For Windows, simply download the .exe file and save it to the C: drive and add the path to the .exe to the system’s “PATH” environment variable. Refer to the curl installation in the next section for instructions on editing the system environment variables.

For Mac OSX, download the appropriate binary file and move it into your **`/usr/bin/`** directory with the **`mv`** command. Note that execute permissions may need to be added to the binary with the following command.

```
sudo chmod +x /path-to-jq-binary ↲
```

If any permission errors are raised, prepend the commands with **sudo** and enter the administrator password (password used to log on to the Mac).

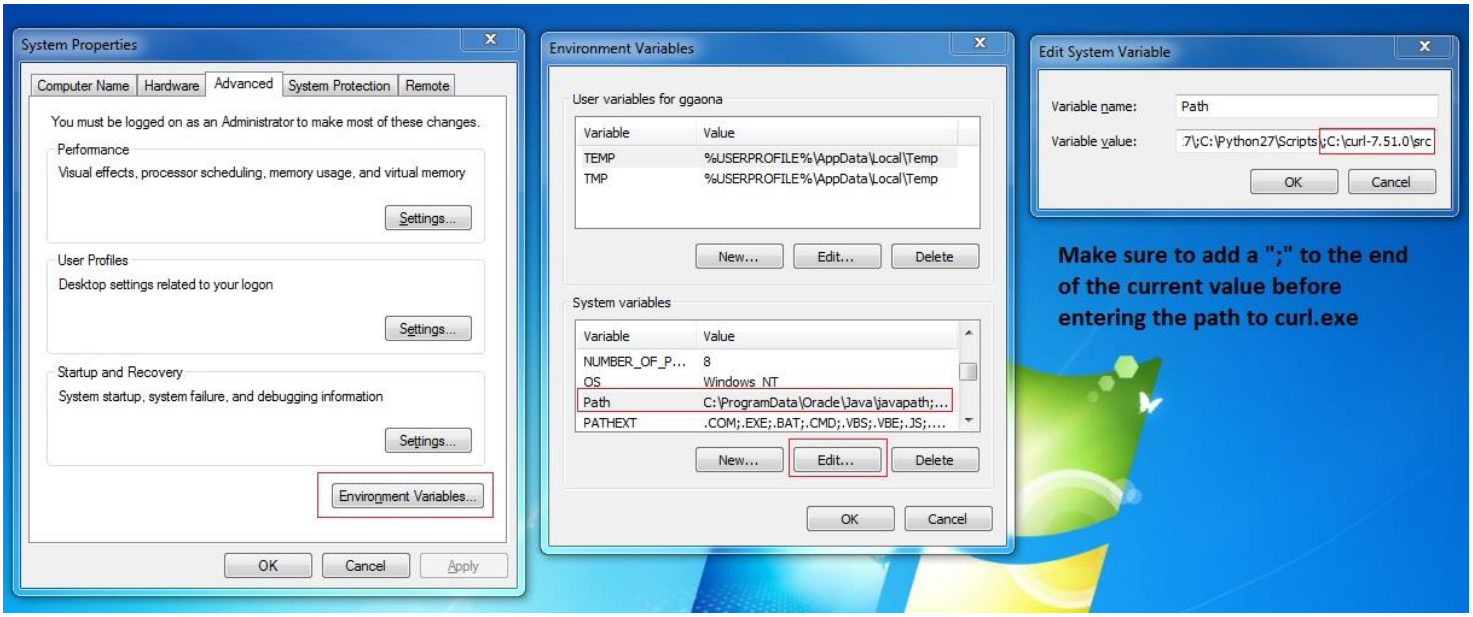
17.1.2 Using Curl

17.1.2.1 Windows Install

Curl can be installed on Windows by downloading the source from http://curl.haxx.se/download.html. Get either the generic win32 or win64 package depending on whether the Windows OS being used is a 32-bit or 64-bit OS. Note that the latest version(s) should be highlighted in yellow.

1. Create a "curl" folder in the "C:" drive and extract the contents of the source zip file to that folder.
2. Now open the start menu and in the search bar type "edit the system environment variables" and press the enter key.
3. In the window that appears click on the "Environment Variables" button toward the bottom right corner.
4. In the next window find the "Path" variable and click the "Edit" button.
5. Finally add the path of the curl.exe to the end of the variable value which should be C:\curl\src\ if the source zip was extracted into a curl folder on the C drive.

See the following image for clarification.



Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

Now curl can be run from the Windows command line. Note that variables can be set and accessed in the Windows command line like so.

```
set foo=bar ↲ # set value of "bar" to variable "foo"

echo %foo% ↲ # prints value of variable "foo"

set "foo=" ↲ # removes value for variable "foo"
```

17.1.2.2 API Examples

17.1.2.2.1 Checking Status

To verify that the server is up, and the firewall is correctly configured, use the status endpoint.

```
SERVER=http://127.0.0.1:9000 ↲ # replace 127.0.0.1 with the
Quickstart Server IP

curl $SERVER/status ↲ # prints OK to the command line if the server
is not experiencing any issues
```

If an error message is returned, first double check that the correct IP is being used. Otherwise, the server is either on a different port or the firewall is incorrectly configured. At that point, please contact support for further assistance.

17.1.2.2.2 Querying Data

With a stream UUID, either a raw query or statistical query may be performed. Raw queries retrieve data at its native resolution and statistical queries which obtains coarser resolution but is substantially faster.

To get an idea of how much data is available, a bracket query may be used and is formatted like so.

```
curl -X POST $SERVER/q/bracket -d '{"UUIDS": ["UUID-Goes-Inside-
Quotes"]}' ↲
```

Example:

```
SERVER=http://127.0.0.1:9000 ↲ # replace with Quickstart Server IP

UUID=f0896731-0e43-4ed5-a97f-19266d8e732c ↲

curl -X POST $SERVER/q/bracket -d '{"UUIDS": ["'$UUID'"]}' | jq ↲ #
jq command is optional and used to print returned data in more
readable format. See the    "Optional jq Installation" section for
more info. Also note that the $UUID variable must be wrapped in "'"
double and single quotes
```

Windows Example (minor differences in how variables are accessed and escaped characters):

```
set SERVER=172.17.6.122:9000 ↲
```

```
set UUID=f0896731-0e43-4ed5-a97f-19266d8e732c ↲
curl –X POST %SERVER%/q/bracket –d "{\"UUIDS": [\"%UUID%\"]}" | jq ↲
```

The server will return data in JSON format if it recognizes the request as a valid request.

```
{
      "Brackets": [
            [
                  1477100329040881700,
                  1477186729034743600
            ]
      ],
      "Merged": [
            1477100329040881700,
            1477186729034743600
      ]
}
```

The data in brackets is the start and end time of the stream's current recording period in Unix time (measured in nanoseconds). The data in merged is the start and end time of all queried streams' current recording time period.

A point width parameter can also be used in conjunction with the start and end time of the stream to perform a statistical query. The point width sets the resolution of the data returned, meaning that it will return statistics in the interval set by the point width value. Point width values are equal to log base 2 of a time interval in nanoseconds. For example, pw=23 is 2^23 nanoseconds or 8.388 ms. Below is a table of point width reference values.

PW	Rough Equivalent	Actual Value
23	8.388 ms	16,777,216
30	1 second	1,073,741,824
36	1 minute	68,719,476,736
42	1 hour (73 minutes)	4,398,046,511,104
46	1 day (19.5 hours)	70,368,744,177,664
49	1 week (6.5 days)	562,949,953,421,312

51	1 month (26 days)	2,251,799,813,685,248
54	1 year (208 days)	18,014,398,509,481,984

BTrDB does support exact windowing, although not through the REST API. Point widths are useful because they allow for significantly faster query responses than exact windowing through the binary interface.

A statistical query of the example stream will be structured as follows.

```
STIME=1477100329040881700 ↲

ETIME=1477186729034743600 ↲

PW=46 ↲ # returns data in 1 day intervals

PARAMS="starttime=$STIME&endtime=$ETIME&unitoftime=ns&pw=$PW" ↲

UUID=f0896731-0e43-4ed5-a97f-19266dd8e732c ↲

curl -X GET "$SERVER/data/uuid/$UUID?$PARAMS" | jq ↲ # pipe to jq
command is optional
```

Windows Example:

```
set STIME=1477100329040881700 ↲

set ETIME=1477186729034743600 ↲

set PW=46 ↲ # returns data in 1 day intervals

set PARAMS="starttime=%STIME%&endtime=%ETIME%&unitoftime=ns&pw=%PW%"↲

set UUID=f0896731-0e43-4ed5-a97f-19266dd8e732c ↲

curl -X GET %SERVER%/data/uuid/%UUID%?%PARAMS% | jq ↲ # pipe to jq
command is optional. Also note the path is not encased in quotes for
the Windows version of this command
```

The response will look like this.

```
[
      {
            "uuid": "f0896731-0e43-4ed5-a97f-19266dd8e732c",
            "XReadings": [
                  [
                        1477039940289,
```


Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com

```
                    167360,
                    95915605781875420,
                    95915929807604350,
                    95916253833333300,
                    1197648
                ],
                [
                    1477110309033,
                    345024,
                    95916253833874380,
                    95918538533067200,
                    95920823232259970,
                    8444587
                ]
            ],
            "version": 2
        }
    ]
```

Since there are only two recordings, this indicates the total recording period for the stream is roughly two days since our point width interval was about one day. Each recording contains the following six fields.

- Timestamp in nanoseconds since the epoch
- The remainder of the timestamp in nanoseconds
- The minimum recorded value of the stream
- The average recorded value of the stream
- The maximum recorded value of the stream
- The number of values

The `jq` command can be used to extract parts of this data. For example, retrieve the number of values with the same command, then pipe to `jq` like so.

```
curl … | jq .[0].XReadings[][5] ↲
```

Which will return only the sixth field for every reading (in this case, the number of values).

Statistical queries can also return results in CSV format. The endpoint for the CSV response accepts JSON parameters rather than URL parameters. Below is an example.

```
SERVER=http://127.0.0.1:9000 ↲ # replace with Quickstart Server IP
curl –X POST $SERVER/directcsv –d
'{
      "UUIDS":["'$UUID'"], # make sure to wrap in "'" double quote, single quote
      "Labels": ["C1ANG"],
      "StartTime": '"$STIME"', # make sure to wrap in '"' single quote, double quote
      "EndTime": '"$ETIME"',
      "UnitofTime": "ns",
      "PointWidth": 46
}' > file.csv ↲ # where file.csv is the name of the output file to write to
```

Widows Example (Note the command must be on one line in Windows, although the example is split in multiple lines for readability):

```
curl –X POST %SERVER%/directcsv –d
"{
      \"UUIDS\":[\"%UUID%\"],
      \"Labels\": [\"C1ANG\"],
      \"StartTime\": \"%STIME%\",
      \"EndTime\": \"%ETIME%\",
      \"UnitofTime\": \"ns\",
      \"PointWidth\": 46
}" > file.csv ↲ # where file.csv is the name of the output file to write to
```

Finally, over small periods of time, raw data can be queried. This looks the same as the statistical query, but does not include the point width parameter. Below is an example.

```
STIME=1474321947545300000 ↲ # 9/19/16 2:52:28 PM
```

```
ETIME=1474326345591810000 ↲ # 9/19/16 4:05:46 PM
PARAMS="starttime=$STIME&endtime=$ETIME&unitoftime=ns" ↲
UUID=f0896731-0e43-4ed5-a97f-19266d8e732c ↲
curl -X GET "$SERVER/data/uuid/$UUID?$PARAMS" > data.json ↲
```

Note that even for this small time period (~1 hour), the file is quite large (10MB). Take care not to perform raw queries over too much data. Example of file information output via **ls** command.

```
$ ls -hal data.json
$ -rw-rw-r-- 1 userA userA 10M Oct 23 13:05 data.json
```

Windows Example:

```
set STIME=1474321947545300000 ↲ # 9/19/16 2:52:28 PM
set ETIME=1474326345591810000 ↲ # 9/19/16 4:05:46 PM
set PARAMS="starttime=%STIME%&endtime=%ETIME%&unitoftime=ns" ↲
set UUID=f0896731-0e43-4ed5-a97f-19266dd8e732c ↲
curl -X GET %SERVER%/data/uuid/%UUID%?%PARAMS% > data.json ↲
```

17.2 BTrDB Binary API Examples

17.2.1 Binary API via Python

NOTE: It is assumed that users of the python interface to the binary API are familiar with the python programming language. Otherwise it is recommended to use the REST API instead.

The python interface for the binary API is supported on Linux and Mac OSX, but not Windows. Before installing the **btrdbcapnp** module, make sure **pip** (python package manager) is installed. Installation instructions for **pip** can be found at https://pip.pypa.io/en/stable/installing/.

To install the **btrdbcapnp** module, first download the module from https://github.com/SoftwareDefinedBuildings/btrdb-python/archive/master.zip. Once downloaded, extract the contents of the zip file then move into the extracted folder from the command line and run the setup.py file.

```
cd path-to-extracted-zip/btrdb-python-master/ ↲
python setup.py install ↲
```

Once installed, the binary API can be queried with python such as in the following example.

```
>>> import uuid ↲
```

```
>>> import btrdbcapnp ↲
>>>
>>> # This is the UUID of the stream we are going to interact with
>>> u = uuid.UUID('6390e9df-dfcb-4084-8080-8c719ce751ed') ↲
>>>
>>> # Set up the BTrDB Connection and Context
>>> connection = btrdbcapnp.BTrDBConnection('localhost', 4410) ↲
>>> context = connection.newContext() ↲
>>>
>>> # Insert some data
>>> # We have to use the "sync" flag because we want the insert to be
committed immediately, so that we can query it right away
>>> statuscode = context.insertValues(u, ((1, 10), (3, 14), (5, 19),
(9, 13)), sync = True) ↲
>>> print statuscode ↲
ok
>>> # Query some data
>>> result = context.queryStandardValues(u, 0, 7) ↲
>>> print result ↲
([(1, 10.0), (3, 14.0), (5, 19.0)], 2L)
>>>
>>> # Close the context and connection
>>> context.destroy() ↲
>>> connection.close() ↲
```

Documentation of the module can also be viewed with the following command.

```
pydoc btrdbcapnp ↲
```

17.2.2 Binary API via Go

NOTE: It is assumed that users of the Go interface to the binary API are familiar with the Go programming language. Otherwise it is recommended to use the REST API instead.

The Go interface to the binary API can be installed with the following command.

```
go get github.com/go-btrdb/btrdb ↵
```

The following is an example program for interacting with the binary API.

```
import (
      "gopkg.in/btrdb.v3"
      "github.com/pborman/uuid"
)

func main() {
      var myuuid uuid.UUID
      var err error
      var bc *btrdb.BTrDBConnection
      var version uint64
      var versionchan chan uint64
      var svchan chan btrdb.StandardValue
      var sv btrdb.StandardValue
      var points []btrdb.StandardValue
      var statcode chan string
      var strstatcode string
      var asyncerr chan string

      bc, err = btrdb.NewBTrDBConnection("localhost:4410")
      if err != nil {
            /* Fatal error */
      }

      /* UUID of the stream into which to insert/query */
```

```
myuuid = uuid.NewRandom()

/* Points to insert */
points = []btrdb.StandardValue {
      btrdb.StandardValue{Time: 1, Value: 2.0},
      btrdb.StandardValue{Time: 4, Value: 7.5},
      btrdb.StandardValue{Time: 6, Value: 2.5},
      btrdb.StandardValue{Time: 13, Value: 8.0},
      btrdb.StandardValue{Time: 15, Value: 6.0},
}

/* Insert */
statcode, err = bc.InsertValues(myuuid, points, true)
strstatcode = <- statcode
if err != nil || "ok" != strstatcode {
      /* Error */
}

/* Standard Values Query */
svchan, versionchan, asyncerr, err =
bc.QueryStandardValues(myuuid, 0, 16, 0)
if err != nil {
      /* Error */
}

for sv = range svchan {
      /* Handle Point */
}

/* Get the version used to satisfy the query */
```

```
        version = <- versionchan
}
```

For technical assistance or the latest version of this document, please call 1-510-522-4455 or email us at: support@powersensorsltd.com if you have any further questions.

Power Sensors Ltd. 980 Atlantic Ave, Alameda CA 94501, USA
Tel ***++1-510-522-4400*** Fax ***++1-510-522-4455*** www.PowerSensorsLtd.com