Sasquatch: How To Access The EFD

Silvia Pietroni
INAF OACN
Vera Rubin - LSST

February 2024

Abstract

Sasquatch is the Rubin Observatory's service for metrics and telemetry data. Built on Kafka and InfluxDB, it offers a comprehensive solution for collecting, storing, and querying time-series data. Sasquatch is currently deployed at the Summit, USDF and test stands through Phalanx (a GitOps repository for Rubin Observatory's Kubernetes environments, notably including Rubin Science Platform deployments). Rubin Observatory generates a large amount of engineering data, which is stored in the Engineering and Facilities Database (EFD).

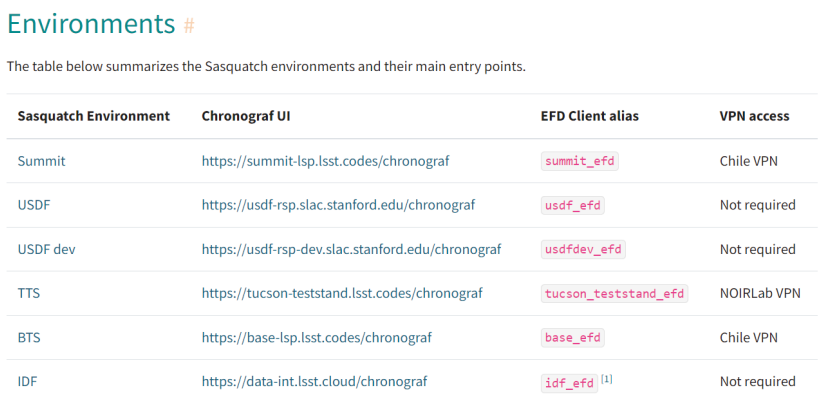
In this paper there is a brief guide on how to access the EFD from the different environments available by focusing on the USDF Chronograf which is accessible without Chile VPN, and a huge part is dedicated to the EdfClient Python client on how we can get the list of all available Telemetry Topics in SAL scripts, all the related fields in each topic and the list of dictionaries describing these fields. Several example are given on specific mechanical parts of the telescope.

1 Introduction

Sasquatch is the Rubin Observatory's service for metrics and telemetry data, it provides a detailed User Guide for the Observatory Telemetry (EFD - Engineering and Facilities Database), Analysis Tools metrics and Data exploration and visualization with Chronograf. Sasquatch provides also different Environments, a Developer Guide and technical docs and software in Rubin Docs. In Sec. 2 we make a brief overview on the available environments, in Secs. 3-4 we give a brief explanation of the Summit and BTS environments, in Sec. 5 we explain how to access and how to use the USDF (US Data Facility), in Sec. 6 we show how to access the USDF Rubin Science Platform with some example notebooks and finally in Sec. 7 we find some useful links to the LSST (Large Synoptic Survey Telescope) instrumentation.

2 Environments

In each Sasquatch Environment we have a Chronograf according to the following Table 1:



Environments #

The table below summarizes the Sasquatch environments and their main entry points.

Sasquatch Environment	Chronograf UI	EFD Client alias	VPN access
Summit	https://summit-lsp.lsst.codes/chronograf	summit_efd	Chile VPN
USDF	https://usdf-rsp.slac.stanford.edu/chronograf	usdf_efd	Not required
USDF dev	https://usdf-rsp-dev.slac.stanford.edu/chronograf	usdfdev_efd	Not required
TTS	https://tucson-teststand.lsst.codes/chronograf	tucson_teststand_efd	NOIRLab VPN
BTS	https://base-lsp.lsst.codes/chronograf	base_efd	Chile VPN
IDF	https://data-int.lsst.cloud/chronograf	idf_efd [1]	Not required

Figure 1: Sasquatch Environments.

For our purpose we need the Summit, the Base Test Stand (BTS) and the USDF (US Data Facility) Environments with their respective Chronographs. The Summit and the BTS Chronographs requires the Chile VPN and other authorizations, while it is possible to access the USDF Chronographs by logging with the institutional LSST account (S3DF or SLAC).

For any technical issue with these environments it is possible to drop a line at #com-square-support on LSSTC Channel on Slack.

3 Summit

Summit is the main production operational environment. Systems running here will be directly controlling hardware or communicating with components that control actual hardware.

The EFD is used at the Summit for real-time monitoring of the observatory systems using Chronograf dashboards and alerts. This instance collects engineering data from the Summit and is the primary source of EFD data.

Intended audience: Observers and Commissioning team at the Summit.

4 Base Test Stand - BTS

The Base Test Stand (BTS) is a simulation operational environment running at the base facility in La Serena. Some systems will run using hardware simulators. Hardware simulators are higher fidelity than those of pure software, for that,

BTS will be used mostly for higher fidelity testing, development and debugging. It will also be particularly useful for testing lower-level hardware interfaces.

Intended audience: Telescope & Site team.

5 USDF - US Data Facility

The EFD data is also replicated to the US Data Facility (USDF) for long-term storage and analysis.

Intended audience: Project staff.

Once into the USDF Chronograf, we have this screen with a lateral vertical bar as shown in Fig.2:

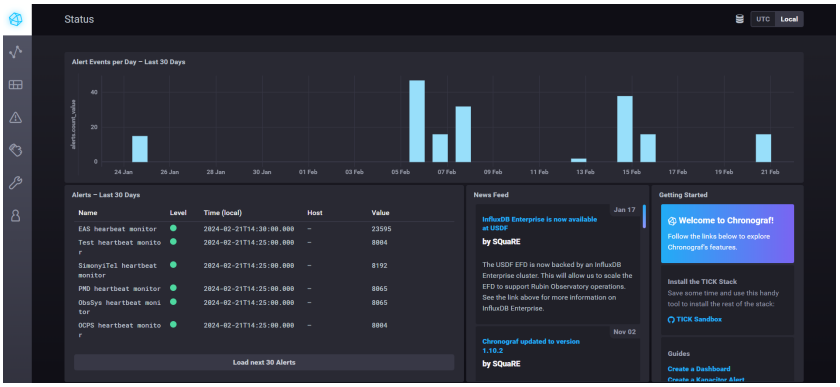


Figure 2: USDF Chronograf Status.

then we can move down to EXPLORE on the left bar as in Fig.3:

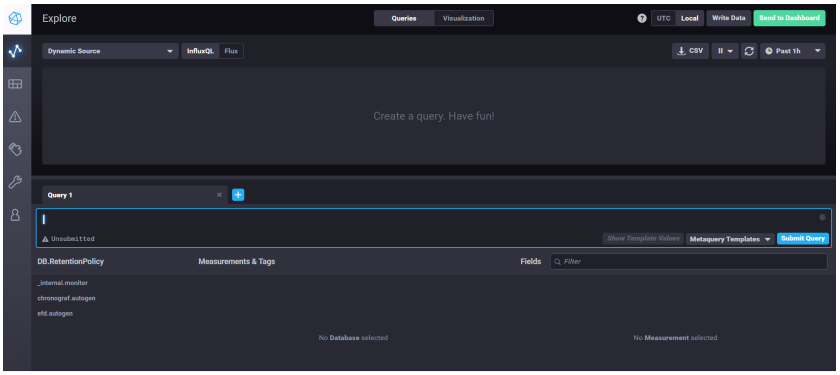


Figure 3: USDF Chronograf Explore.

From the Dashboard, see Fig.4,

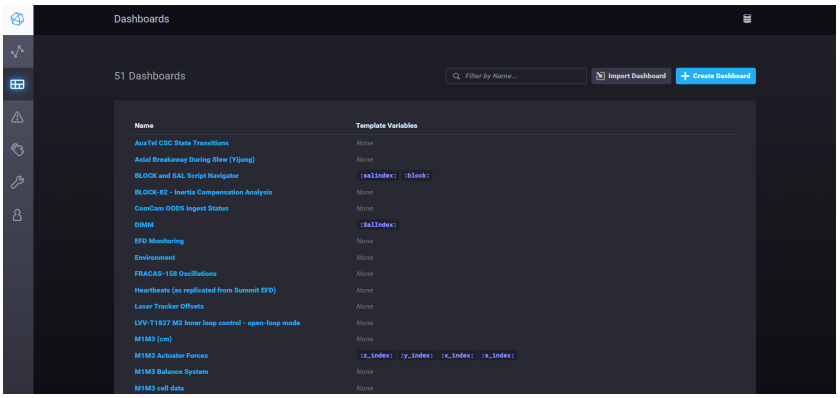


Figure 4: USDF Chronograf Dashboard.

we can select a component from LSST (please see Rubin Technology), for example the MTMount (the Telescope Mount), where we can see all the variables such as the azimuth or the elevation (see Fig.5) at different samples and different epochs (minutes, hours, or, with the Data Picker, see Fig.6, we can choose a specific date, at the top right of the Dashboard).

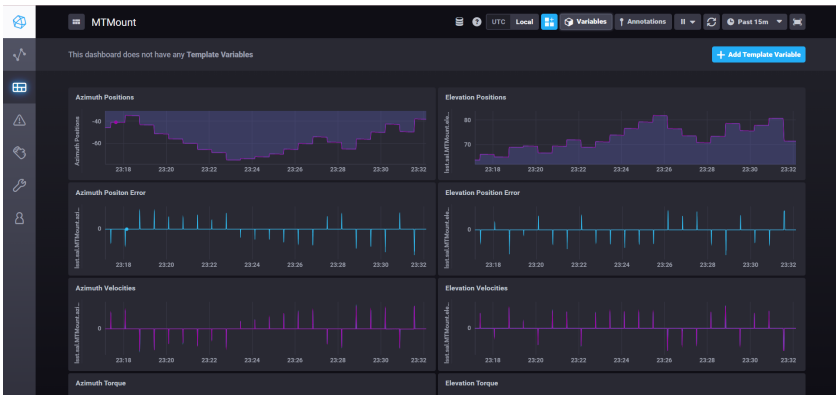


Figure 5: USDF Chronograf: MT Mount Variables.

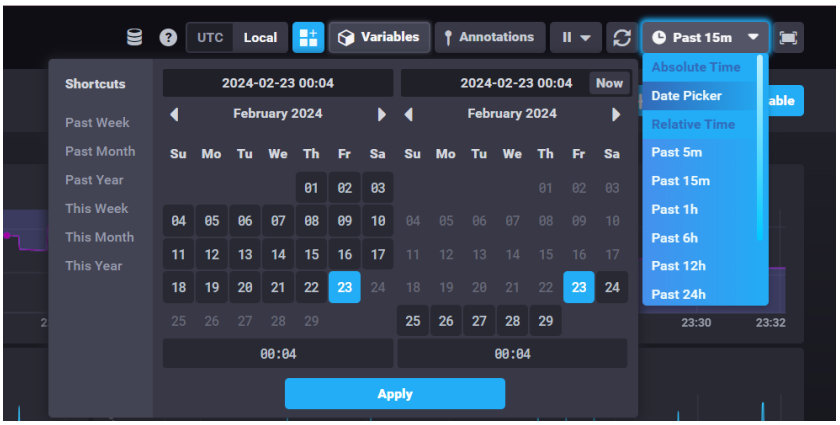


Figure 6: USDF Chronograf: Data Picker.

It is also possible, on each single variable, to download the respective CSV file, as shown in the example in Fig.7.

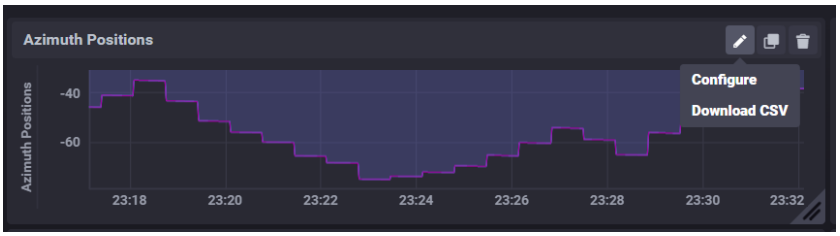


Figure 7: USDF Chronograf: how to download a CSV file from a single variable.

In Fig.8 there is another example for MTM2, one of the Mirrors:

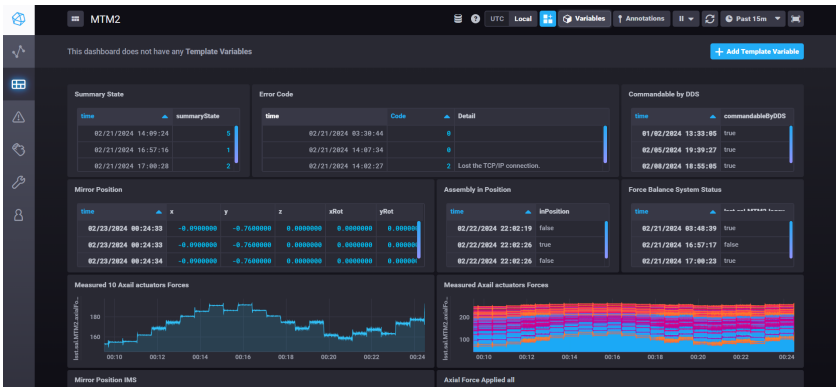


Figure 8: MTM2 variables from the Dashboard.

6 The RSP and the EFD Python Client

It is possible also to access the Chronograf through a Python script.

It is necessary to go on the USDF RSP (Rubin Science Platform) (see Fig.9)

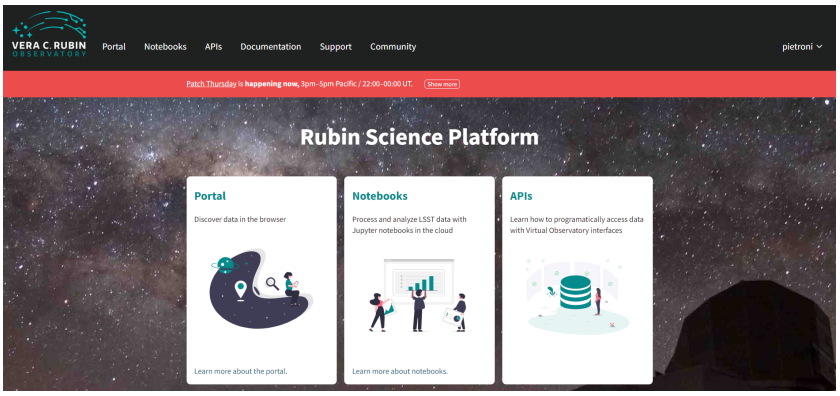


Figure 9: USDF RSP (Rubin Science Platform)

and to click on Notebooks where some tutorial-notebooks are available. Then it is possible to read the User Guide at the The EFD Python client where there is the explanation on how to instantiate the EFD client with a Python script and where it is possible to create a notebook on the USDF Rubin Science Platform in order to get the EFD data.

There are Python functions and classes in the Python API reference documentation. Among the classes to handle connections and basic queries asynchronously we can refer to EfdClient.

We give a detailed explanation of an example notebook able to access the EFD on the USDF Environment which is available on GitHub - Silvia Pietroni.

6.1 Available Topics and Variables in SAL Scripts

First we initiate our code with:

```
from lsst_efd_client import EfdClient
efd_client = EfdClient("usdf_efd")
```

then, in order to get all the available topics, we can use SAL scripts which are programs that perform coordinated telescope and instrument control operations; in our case we have a list of Telemetry Topics or we are able to get a list through this command:

```
topics = await efd_client.get_topics()
```

which gives more than 2320 topics, in Fig.10 a fraction of the list of topics available for the MTM2 and for the MTMount:

```
1778 lsst.sal.MTM2.mirrorPositionMeasured
1779 lsst.sal.MTM2.netForcesTotal
1780 lsst.sal.MTM2.netMomentsTotal
1781 lsst.sal.MTM2.position
1782 lsst.sal.MTM2.positionIMS
1783 lsst.sal.MTM2.powerStatus
1784 lsst.sal.MTM2.powerStatusRaw
1785 lsst.sal.MTM2.rawDisplacement
1786 lsst.sal.MTM2.rawTelemetry
1787 lsst.sal.MTM2.stepVectorUpdate
1788 lsst.sal.MTM2.systemStatus
1789 lsst.sal.MTM2.tangentActuatorAbsolutePositionSteps
1790 lsst.sal.MTM2.tangentActuatorPositionAbsoluteEncoderPositionMeasured
1791 lsst.sal.MTM2.tangentActuatorSteps
1792 lsst.sal.MTM2.tangentEncoderPositions
1793 lsst.sal.MTM2.tangentForce
1794 lsst.sal.MTM2.tangentForcesMeasured
1795 lsst.sal.MTM2.targetForces
1796 lsst.sal.MTM2.temperature
1797 lsst.sal.MTM2.temperaturesMeasured
1798 lsst.sal.MTM2.zenithAngle
1799 lsst.sal.MTM2.zenithAngleMeasured
1800 lsst.sal.MTMount.Camera_Cable_Wrap
1801 lsst.sal.MTMount.Safety_System
1802 lsst.sal.MTMount.ackcmd
1803 lsst.sal.MTMount.auxiliaryBoxes
1804 lsst.sal.MTMount.auxiliaryCabinetsThermal
1805 lsst.sal.MTMount.azimuth
1806 lsst.sal.MTMount.azimuthCableWrap
1807 lsst.sal.MTMount.azimuthDrives
1808 lsst.sal.MTMount.azimuthDrivesThermal
1809 lsst.sal.MTMount.balancing
1810 lsst.sal.MTMount.cabinet0101
```

Figure 10: Some of the SAL commands available for the MTM2 and for the MTMount.

We can get all the available variables (fields) in a single topic with the **get_fields(topic_name)** method (where we fill *topic_name* with one of the SAL commands):

```
1 await efd_client.get_fields("lsst.sal.MTM2.position")
```

and we can get, from a given topic, a list of dictionaries describing the fields with the **get_schema(topic)** method (where we fill *topic* again with one of the SAL commands):

```
1 await efd_client.get_schema("lsst.sal.MTM2.position")
```

in Fig.11 an example of a list of dictionaries for the MTM2 position:

	name	description	units	aunits	is_array
0	private_efdStamp	UTC time for EFD timestamp. An integer (the nu...	second	s	False
1	private_kafkaStamp	TAI time at which the Kafka message was created.	second	s	False
2	private_revCode	Revision hashcode	unitless		False
3	private_sndStamp	Time of instance publication	second	s	False
4	private_rcvStamp	Time of instance reception	second	s	False
5	private_seqNum	Sequence number	unitless		False
6	private_identity	Identity of publisher	unitless		False
7	private_origin	PID of publisher	unitless		False
8	x	Position x.	micron	micron	False
9	y	Position y.	micron	micron	False
10	z	Position z.	micron	micron	False
11	xRot	Rotation about x.	arcsec	arcsec	False
12	yRot	Rotation about y.	arcsec	arcsec	False
13	zRot	Rotation about z.	arcsec	arcsec	False

Figure 11: An example of a list of dictionaries for the MTM2 position.

6.2 Time Series: Samples in a Specific Time Interval

If we define a specific time interval, for example:

```
t_start = Time("2024-02-23T19:20:00", scale="utc", format="isot")
t_end = Time("2024-02-23T19:25:00", scale="utc", format="isot")
```

we are able to get, from a topic, all the variable values in that specific time interval with the **select_time_series(topic_name, fields=[ ], start, end)** method; here an example for the MTM2 position (we fill *fields* with the variables obtained (as in the example in Fig. 11), and we fill *start* and *end* with a time interval):

```
await efd_client.select_time_series("lsst.sal.MTM2.position",
fields=["x","xRot","y","yRot","z","zRot"],
start=t_start, end=t_end)
```

and we get a table as in Fig.12:

	x	xRot	y	yRot	z	zRot
2024-02-23 19:20:00.005327+00:00	-0.09	0	-0.76	0	0	0.01
2024-02-23 19:20:00.202694+00:00	-0.09	0	-0.76	0	0	0.01
2024-02-23 19:20:00.400641+00:00	-0.09	0	-0.76	0	0	0.01
2024-02-23 19:20:00.600401+00:00	-0.09	0	-0.76	0	0	0.01
2024-02-23 19:20:00.801134+00:00	-0.09	0	-0.76	0	0	0.01

Figure 12: Table for a specific time interval and for specific variables.

Other examples on more SAL Scripts and variables of different instruments are listed in the notebook (MTM2, MTM1M3, AT Camera, ATMCS (Auxiliary Telescope Mount Control System), ATPtg), please go to Sec. 7 for useful links, but in order do get data it is necessary to know a specific time interval in which measurements are available.

It is also possible to select fields that are time samples and unpack them into a dataframe with the **select_packed_time_series(topic_name, base_fields, start, end)** method as in the following example:

```
ATMCS_mount_AzEl_Encoder = await
efd_client.select_packed_time_series("lsst.sal.ATMCS.
    mount_AzEl_Encoders", base_fields=["azimuthCalculatedAngle",
 "elevationCalculatedAngle"], start=t_start, end=t_end)
```

where we fill *base_fields* with the fields obtained with the *get_schema* method shown before; we get a table as in Fig. 13 for the Mount AzEl Encorders (telemetry for elevation and azimuth encoders):

	azimuthCalculatedAngle	elevationCalculatedAngle	times
2024-02-23 19:19:59.509643078	188.698957	38.996416	1.708716e+09
2024-02-23 19:19:59.519643307	188.698950	38.996417	1.708716e+09
2024-02-23 19:19:59.529643297	188.698948	38.996421	1.708716e+09
2024-02-23 19:19:59.539643288	188.698943	38.996419	1.708716e+09
2024-02-23 19:19:59.549643517	188.698947	38.996419	1.708716e+09

Figure 13: Table for time samples unpacked into a dataframe for the Mount AzEl Encorders (telemetry for elevation and azimuth encoders).

If we want to select the most recent N samples from a set of topics in a single subsystem we can use the **select_top_n(topic_name, fields, num)** method, where *num* is the number of rows to return, as in the following example for the net moment of force in X, Y, Z directions:

```
moments_n_samples=
await efd_client.select_top_n("lsst.sal.MTM2.netMomentsTotal",
num=5, fields=["mx","my","mz"])
```

where we get the following table as in Fig. 14:

	mx	my	mz
2024-03-17 08:30:00.070788+00:00	1261.04	25.24	-231.67
2024-03-17 08:29:59.871360+00:00	1262.08	22.41	-223.98
2024-03-17 08:29:59.671503+00:00	1264.03	22.26	-235.10
2024-03-17 08:29:59.474388+00:00	1259.58	24.09	-226.98
2024-03-17 08:29:59.278734+00:00	1263.55	23.22	-237.32

Figure 14: Table for the most recent N=5 samples from a set of topic for the MTM2 net moment of force in X, Y, Z directions.

6.3 Queries with InfluxQL

To perform queries directly using InfluxQL (an SQL-like query language for interacting with data in InfluxDB) we can make use of the **influx_client.query()** method in conjunction with the EFD client.

For a comprehensive understanding of the InfluxQL query syntax, we recommend referring to the InfluxQL documentation.

With the **build_time_range_query(topic_name, fields, start, end)** method we can build a query based on a time range as we can see in the following example for the net moment of force in the X, Y, Z directions:

```
query=efd_client.build_time_range_query("lsst.sal.MTM2.
    netMomentsTotal", fields=["mx","my","mz"], start=t_start,
end=t_end)

moments=await efd_client.influx_client.query(query)
```

where the query we get is:

```
'SELECT mx, my, mz FROM "efd"."autogen"."lsst.sal.MTM2.netMomentsTotal"
    WHERE time >= \'2024-02-23T19:20:00.000Z\'
    AND time <= \'2024-02-23T19:25:00.000Z\''
```

and the plots in that specific time interval is in Figs. 15:

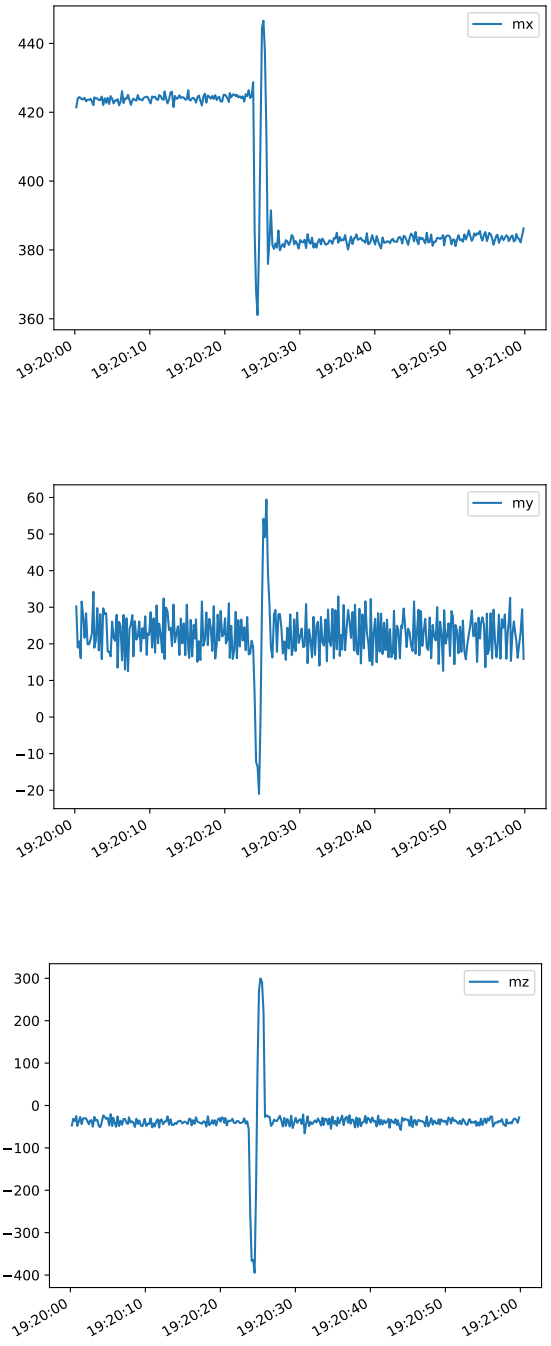


Figure 15: Net Moment Force in the X, Y, Z Directions.

Other examples on querying the EFD with InfluxQL can be found on Github - Silvia Pietroni.
If you use `lsst.sal.ATPtg.mountPositions` please remember to put `ref_timestamp_col="cRioTimestamp"` in the **select_packed_time_series()** method.

7 LSST Instrumentation

- Camera
- Camera (second link)
- Mirrors
- Telescope Mount
- ATCS (Auxiliary Telescope Control System) which comprehends ATPtg, ATMCS, ATAOS, ATDomeTrajectory, ATDome, ATPneumatics, ATHexapod)
- Observatory Facility