Vera C. Rubin Observatory
Data Management

# LDM-503-EFDb: Replication of Summit EFD to USDF Test Plan and Report

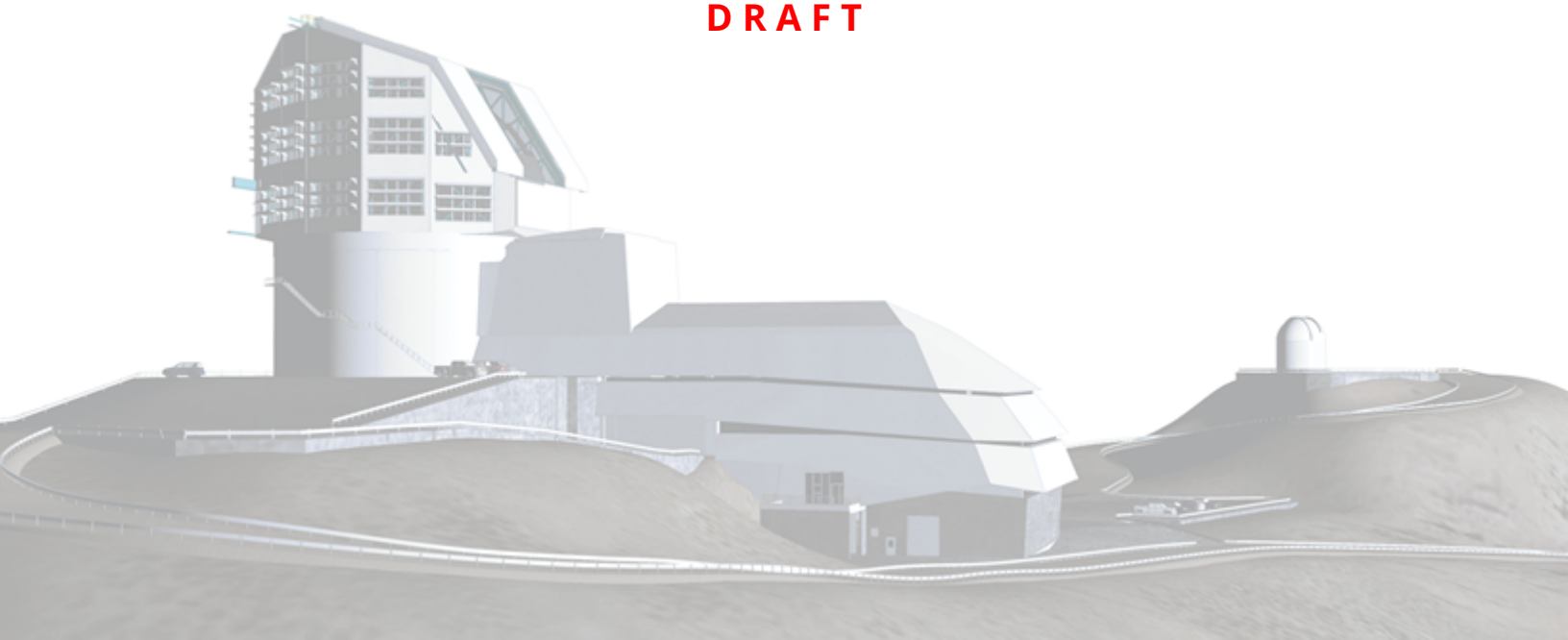
**William O'Mullane**

**DMTR-331**

**Latest Revision: 2025-03-04**



**DRAFT**



## Abstract

This is the test plan and report for **Replication of Summit EFD to USDF** (LDM-503-EFDb), an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2025-03-04 . The most recent change to the document repository was on 2025-03-04.

# Change Record

| Version | Date | Description | Owner name |
|---|---|---|---|
| 0.1 | 2021-10-19 | First draft | K. Simon Krughoff |
| 0.2 | 2023-06-16 | Update for move to USDF | WOM |
| 1.0 | 2024-02-08 | Surcessfully executed on summit and USDF | WOM |
| 1.1 | 2024-02-16 | With executions in summary (docsteady 2.5.5) | WOM |

*Document curator:* William O'Mullane
*Document source location:* `https://github.com/lsst-dm/DMTR-331`
*Version from source repository:* 0155f73-dirty

# Contents

# LDM-503-EFDb: Replication of Summit EFD to USDF Test Plan and Report

## 1 Introduction

### 1.1 Objectives

The purpose of this test plan is to describe all the necessary requirements and infrastructure for successfully testing the replication and archive of the Engineering Facility Database (EFD) as implemented with Kafka, InfluxDB and Chronograf from the summit to the USDF. This plan will describe the prerequisites for beginning a test campaign, step by step instructions for each test case and a description of the expected results and test artifacts.

NB: The use of the term reliability in this document is intended to indicate the number of messages produced relative to the number of messages recorded in the EFD. The system shall be considered reliable if at least 99.9% of produced messages are recorded.

At a high level, this test plan is intended to show that a nominally operating EFD at the summit is able to be replicated to the USDF and archived for future use either directly or via ingest into a secondary database management technology. We assume here that the archive technology will be parquet datasets stored on persistent/redundant disk at the USDF. There are no latency requirements in this test plan, but we will show that the replication and archiving are not falling behind relative to the summit instance in the aggregate. We choose a period of 6 days of continuous nominal operation in order to test the cases in this test plan.
Successful completion of the test campaign will show that:

1. users are able to access the same information at the USDF EFD that was originally ingested in the summit version
2. the reliability of the replication is better than the minimum of 99%
3. archive products are able to be used as a primary source of information for historical examination of EFD topics

## 1.2 System Overview

The tests will be carried out from within an instance of the notebook aspect of the RSP running at the data facility where the EFD replication is currently happening. An appropriate weekly version of the stack will be chosen.

## 1.3 Document Overview

This document was generated from Jira, obtaining the relevant information from the LVV-P90 Jira Test Plan and related Test Cycles ( LVV-R181 ).

Section 1 provides an overview of the test campaign, the system under test (Data Management), the applicable documentation, and explains how this document is organized. Section 2 provides additional information about the test plan, like for example the configuration used for this test or related documentation. Section 3 describes the necessary roles and lists the individuals assigned to them.

Section 4 provides a summary of the test results, including an overview in Table 2, an overall assessment statement and suggestions for possible improvements. Section 5 provides detailed results for each step in each test case.

The current status of test plan LVV-P90 in Jira is **Approved** .

## 1.4 References

[1] **[DMTN-140]**, Comoretto, G., 2021, Documentation Automation for the Verification and Validation of Rubin Observatory Software, URL `https://dmtn-140.lsst.io/`, Vera C. Rubin Observatory Data Management Technical Note DMTN-140

[2] **[DMTN-178]**, Comoretto, G., 2021, Docsteady Usecases for Rubin Observatory Constructions, URL `https://dmtn-178.lsst.io/`, Vera C. Rubin Observatory Data Management Technical Note DMTN-178

[3] **[LSE-160]**, Selvy, B., 2013, Verification and Validation Process, URL https://ls.st/LSE-160, Vera C. Rubin Observatory LSE-160

# 2 Test Plan Details

## 2.1 Data Collection

Observing is not required for this test campaign.

## 2.2 Verification Environment

The environment will be within notebooks running a modern stack.

## 2.3 Entry Criteria

1. Before beginning this test, a set of viability tests shall be performed. These will show:
   (a) The system demonstrates reliability (number of recorded messages/number of produced messages) of greater than 99.9%
   (b) The summit data is being replicated to the instance at USDF
   (c) Chronograf is set up and running at both the summit and USDF
2. The summit network and Kubernetes cluster are performing nominally
3. A number of telemetry topics are reliably producing telemetry at both low frequency (1 Hz) and high frequency (> 10 Hz).
4. The notebook aspect of the RSP is deployed in the summit Kubernetes cluster
5. The summit EFD is reliably replicated to an EFD instance running in a data facility
6. The notebook aspect of the RSP is deployed in the same data facility as that running the replicated EFD
7. The most recent version of the EFD client python modules are installed in the various deployed notebook aspects
8. The replication system is also successfully archiving EFD topics to parquet files on persistent disk at the data facility

## 2.4 Exit Criteria

None

## 2.5 Related Documentation

Docushare collection where additional relevant documentation can be found:

- None

## 2.6 PMCS Activity

Primavera milestones related to the test campaign: None

# 3 Personnel

The personnel involved in the test campaign is shown in the following table.

| | | | |
|---|---|---|---|
| | T. Plan LVV-P90 owner: | **William O'Mullane** | |
| | T. Cycle LVV-R181 owner: | **William O'Mullane** | |
| **Test Cases** | **Assigned to** | **Executed by** | **Additional Test Personnel** |
| LVV-T2338 | William O'Mullane | William O'Mullane | William O Mullane |

# 4 Test Campaign Overview

## 4.1 Summary

| | | | | |
|---|---|---|---|---|
| T. Plan LVV-P90: | **LDM-503-EFDb: Replication of Summit EFD to USDF** | | | Approved |
| T. Cycle LVV-R181: | **LDM-503-EFDb: Replication of Summit EFD to USDF** | | | Done |
| **Test Cases** | **Ver.** | **Status** | **Comment** | **Issues** |
| LVV-T2338 | | | | |
| Execution | LVV-E3446 | Pass | None | |

Table 2: Test Campaign Summary

## 4.2 Overall Assessment

After the meltdown we seem to now have a good replica at USDF.

## 4.3 Recommended Improvements

Non.

# 5 Detailed Test Results

## 5.1 Test Cycle LVV-R181

Open test cycle *LDM-503-EFDb: Replication of Summit EFD to USDF* in Jira.

Test Cycle name: LDM-503-EFDb: Replication of Summit EFD to USDF
Status: Done

Engineering Facility Database of summit values queryable at USDF.

### 5.1.1 Software Version/Baseline

b”

### 5.1.2 Configuration

b”

### 5.1.3 Test Cases in LVV-R181 Test Cycle

#### 5.1.3.1 LVV-T2338 - Replicated telemetry data agrees with telemetry produced at the summit

Version **1.0(d)**. Status **Approved**. Open *LVV-T2338* test case in Jira.

Show that telemetry data can be accessed from the replicated EFD. Further, show that the values in the replicated database agree with the values in the summit EFD over a specified time range and set of topics.

This test case provides partial coverage of the requirement DMS-REQ-0168, Summit Facility Data Communications: ”The DMS shall provide data communications infrastructure to accept science data and associated metadata read-outs, and **the collection of ancillary and engineering data**, for transfer to the base facility.”, as adapted to the current design for EFD

replication (see DMTN-082).

**Preconditions**:
See prerequisites in the Test Plan LVV-P90

Execution status: **Pass**
Final comment:
None

Detailed steps results LVV-R181-LVV-E3446 LVV-E3446-1243141982:
**Note:** Steps ”Not Executed” and with No Result are not shown in this report.

# A Documentation

The verification process is defined in LSE-160. The use of Docsteady to format Jira information in various test and planing documents is described in DMTN-140 and practical commands are given in DMTN-178.

# B Acronyms used in this document

| Acronym | Description |
|---|---|
| DM | Data Management |
| DMS | Data Management Subsystem |
| DMS-REQ | Data Management System Requirements prefix |
| DMTN | DM Technical Note |
| DMTR | DM Test Report |
| EFD | Engineering and Facility Database |
| LDM | LSST Data Management (Document Handle) |
| LSE | LSST Systems Engineering (Document Handle) |
| LSST | Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope) |
| LVV | LSST Verification and Validation |
| PMCS | Project Management Controls System |
| RSP | Rubin Science Platform |
| USDF | United States Data Facility |
| VE | vendor estimate |

# C Traceability

| Test Case | VE Key | VE Summary |
|---|---|---|
| LVV-T2338 | LVV-71 | DMS-REQ-0168-V-01: Summit Facility Data Communications |
| LVV-T2339 | LVV-9979 | DMS-NB-REQ-0023-V-01: Access to All Data Products_1 |

# D LVV-T2338 test Notebook

LVV-T2338

March 4, 2025

## 1 For LDM-503-EFDb

Initialize EFDs Summit and USDF

[26]:
```
from lsst_efd_client import EfdClient, resample

client = EfdClient('summit_efd')
client.output = 'dataframe'
cl=client.influx_client



#usdf_client = EfdClient('summit_efd_copy')
usdf_client = EfdClient('usdf_efd')
usdf_client.output = 'dataframe'
usdf_cl=usdf_client.influx_client
```

### 1.1 Pick five topics

[27]:
```
import random
async def selectTopics(pick):
    topics = await client.get_topics()

    selected_topics = []
    loc = "summit"

    day = '2024-01-01'


    # want to select topics randomly but with messages so randomize all indexes
    randoms = random.sample(range(0,len(topics)),len(topics))
    for r in randoms:
       topic = topics[r]
       if ('Test' in topic):
           next
       result = []
       if len(selected_topics) < 5:
```

```
            query=f'''SELECT * FROM "{topic}" WHERE time >  '{day}T00:00:00.
↪000Z' and time < '{day}T00:00:30.000Z' '''
            result = await cl.query(query)

        if len(result) > 20:
            print (f"{topic} had {len(result)} messages in first 30 min of␣
↪{day}")
            selected_topics.append(topic)
            if len(selected_topics) > 4:
              break
    print(f"Random selction of five '{loc}' topics {selected_topics} with␣
↪messages on {day}")
    return selected_topics
```

### 1.2 Utility function to compare reusults

[33]:
```
def cmp(topic, result, sresult):
    print (f"Compare {topic}")
    problems = 0
    if len(result) != len(sresult):
        print (f"\033[91m  {topic} does not have the same number of results␣
↪{len(result)}  - summit had {len(sresult)}\033[0m")
        problems = problems + 1
    else:
        for k in sresult.keys():
            if result[k].all() != sresult[k].all():
                print (f"\033[91m {topic} {key} does not match \033[0m")
                problems = problems + 1
```

[30]:
```
selected_topics = await selectTopics(5) # do this once at least
```

```
lsst.sal.MTDomeTrajectory.logevent_heartbeat had 30 messages in first 30 min of
2024-01-01
lsst.sal.ATMonochromator.logevent_heartbeat had 29 messages in first 30 min of
2024-01-01
lsst.sal.MTM1M3.logevent_heartbeat had 30 messages in first 30 min of 2024-01-01
lsst.sal.ESS.lightningStrikeStatus had 30 messages in first 30 min of 2024-01-01
lsst.sal.Watcher.logevent_heartbeat had 30 messages in first 30 min of
2024-01-01
Random selction of five 'summit' topics
['lsst.sal.MTDomeTrajectory.logevent_heartbeat',
'lsst.sal.ATMonochromator.logevent_heartbeat',
'lsst.sal.MTM1M3.logevent_heartbeat', 'lsst.sal.ESS.lightningStrikeStatus',
'lsst.sal.Watcher.logevent_heartbeat'] with messages on 2024-01-01
```

### 1.3 Get dataframes and compare summit and USDF

Doing this day by day

[34]:
```
# test gives large window 6 days

# May topics

data = {}
for d in range(1,7):
    if d < 10:
       day = f"2024-01-0{d}"
    else:
       day = f"2024-01-{d}"

    day2 = day
    print(f" Checking {day}")
    for topic in selected_topics:
        query=f'''SELECT * FROM "{topic}" WHERE time >  '{day}T00:00:00.000Z'␣
 ↪and time < '{day2}T23:59:59.000Z' limit 1300000  '''
        # could GROUP BY time(1h) maybe ??
        result = await cl.query(query)
        usdf_result = await usdf_cl.query(query)
        cmp(topic, usdf_result, result)
        data[topic] = usdf_result
```

```
 Checking 2024-01-01
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
 Checking 2024-01-02
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
 Checking 2024-01-03
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
 Checking 2024-01-04
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
```

```
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
 Checking 2024-01-05
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
 Checking 2024-01-06
Compare lsst.sal.MTDomeTrajectory.logevent_heartbeat
Compare lsst.sal.ATMonochromator.logevent_heartbeat
Compare lsst.sal.MTM1M3.logevent_heartbeat
Compare lsst.sal.ESS.lightningStrikeStatus
Compare lsst.sal.Watcher.logevent_heartbeat
```

[32]:
```
## Reliability
```

[35]:
```
for topic in selected_topics:
    seqnum = data[topic]['private_seqNum']
    count = 0
    i = 1 # see if the sequence increases mostly
    prev = seqnum[0]
    while i < len(seqnum):
        if (seqnum[i] < prev):
            #print (f"Reset at {i} seqnum : {prev}, {seqnum[i]}")
            count = count + 1
        i = i + 1
    percent = 100 * ((len(seqnum) - count) / len(seqnum))
    print(f"{topic} private_seqNum increases {percent}% of the sequence")
```

```
/tmp/ipykernel_713/336143790.py:5: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  prev = seqnum[0]
/tmp/ipykernel_713/336143790.py:7: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  if (seqnum[i] < prev):

lsst.sal.MTDomeTrajectory.logevent_heartbeat private_seqNum increases 100.0% of
the sequence
lsst.sal.ATMonochromator.logevent_heartbeat private_seqNum increases 100.0% of
the sequence
lsst.sal.MTM1M3.logevent_heartbeat private_seqNum increases 100.0% of the
sequence
lsst.sal.ESS.lightningStrikeStatus private_seqNum increases 100.0% of the
```

sequence
lsst.sal.Watcher.logevent_heartbeat private_seqNum increases 100.0% of the sequence

# E LVV-T2339 test Notebook

LVV-T2339

March 4, 2025

## 1 For LDM-503-EFDb

Initialize EFD at USDF

[6]:
```
from lsst_efd_client import EfdClient, resample

client = EfdClient('usdf_efd')
client.output = 'dataframe'
cl=usdf_client.influx_client
```

[ ]:
```
## Pick 3  topics
```

[11]:
```
import random
topics = await client.get_topics()

selected_topics = []
results = {}
loc = "usdf"
pick = 3
day = '2023-06-13'
day2 = '2023-06-20'

# want to selct 5 topics randomly but with messages so randomize all indexes
randoms = random.sample(range(0,len(topics)),len(topics))
for r in randoms:
   topic = topics[r]
   result = []
   if len(selected_topics) < pick:
      query=f'''SELECT * FROM "{topic}" WHERE time >  '{day}T00:00:00.000Z' and␣
 ↪time < '{day2}T00:00:00.000Z' '''
      result = await cl.query(query)

   if len(result) > 20:
        print (f"{topic} had {len(result)} messages between {day} and {day2} ")
        selected_topics.append(topic)
        results[topic] = result
        if len(selected_topics) > (pick -1):
            break
```

```
print(f"Random selction of {pick} '{loc}' topics {selected_topics} with␣
  ↪messages between {day} and {day2}")
```

```
lsst.sal.ATPneumatics.logevent_heartbeat had 529266 messages between 2023-06-13
and 2023-06-20
lsst.sal.MTAirCompressor.logevent_summaryState had 22 messages between
2023-06-13 and 2023-06-20
lsst.sal.MTM1M3.logevent_raisingLoweringInfo had 6582 messages between
2023-06-13 and 2023-06-20
Random selction of 3 'usdf' topics ['lsst.sal.ATPneumatics.logevent_heartbeat',
'lsst.sal.MTAirCompressor.logevent_summaryState',
'lsst.sal.MTM1M3.logevent_raisingLoweringInfo'] with messages between 2023-06-13
and 2023-06-20
```

[ ]:
```
## read fields ..
```

[13]:
```
for topic in selected_topics:
   result = results[topic]
   print (f"{topic} has fields:{result.columns}")
```

```
lsst.sal.ATPneumatics.logevent_heartbeat has fields:Index(['heartbeat',
'private_efdStamp', 'private_identity',
       'private_kafkaStamp', 'private_origin', 'private_rcvStamp',
       'private_revCode', 'private_seqNum', 'private_sndStamp'],
      dtype='object')
lsst.sal.MTAirCompressor.logevent_summaryState has
fields:Index(['private_efdStamp', 'private_identity', 'private_kafkaStamp',
       'private_origin', 'private_rcvStamp', 'private_revCode',
       'private_seqNum', 'private_sndStamp', 'salIndex', 'summaryState'],
      dtype='object')
lsst.sal.MTM1M3.logevent_raisingLoweringInfo has
fields:Index(['private_efdStamp', 'private_identity', 'private_kafkaStamp',
       'private_origin', 'private_rcvStamp', 'private_revCode',
       'private_seqNum', 'private_sndStamp', 'waitAirPressure',
       'waitHardpoint0',
       …
       'waitZForceActuator91', 'waitZForceActuator92', 'waitZForceActuator93',
       'waitZForceActuator94', 'waitZForceActuator95', 'waitZForceActuator96',
       'waitZForceActuator97', 'waitZForceActuator98', 'waitZForceActuator99',
       'weightSupportedPercent'],
      dtype='object', length=284)
```

[ ]:
```
## plots ..
```

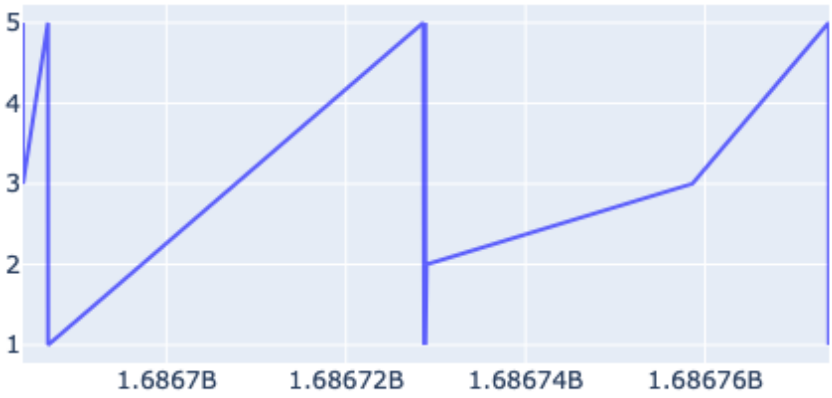
[29]:
```
def plot(topic, results, field):
    df = results[topic]
    bins = (6*24)
    timestep = np.arange(0, bins, 1)
    dates = pd.date_range(start=day, end=day2)
    fig[topic] = go.Figure([go.Scatter(x=df['private_efdStamp'], y=df[field],
                            marker_color='blue',
                            opacity=0.6,
                            name=field)])

    fig[topic].show()
```

[ ]:
```
import plotly.graph_objects as go
import pandas as pd
import numpy as np

plot('lsst.sal.ATPneumatics.logevent_heartbeat', results, 'heartbeat')
```

[31]:
```
plot('lsst.sal.MTAirCompressor.logevent_summaryState', results, 'summaryState')
```




[30]:
```
plot('lsst.sal.MTM1M3.logevent_raisingLoweringInfo', results,␣
↪'weightSupportedPercent')
```