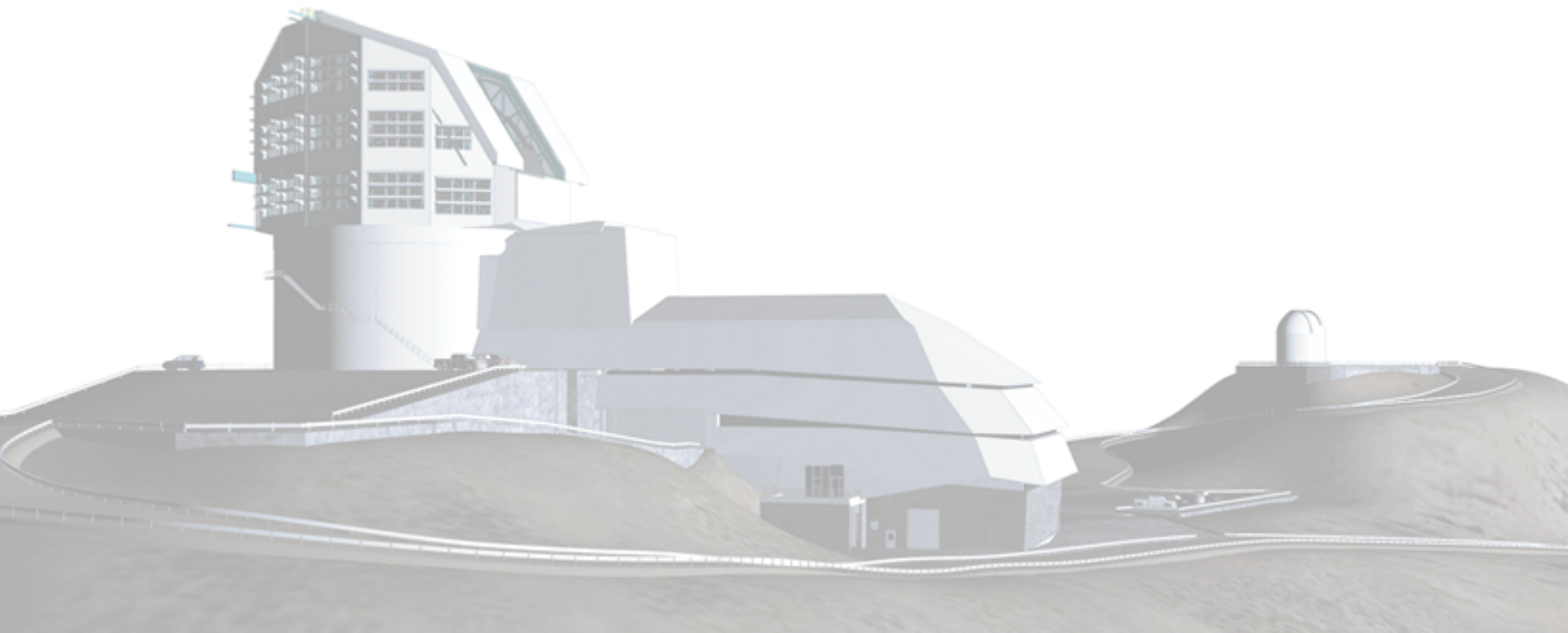


Vera C. Rubin Observatory
Software Test Report

DM-503-EFDa: EFD on Summit for M1/M3 Test Plan and Report

Simon Krughoff

DMTR-291

Latest Revision: 2021-09-27

Abstract

This is the test plan and report for **EFD on Summit for M1/M3** (DM-503-EFDa), an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2021-09-27 . The most recent change to the document repository was on 2021-09-27.

Change Record

Version	Date	Description	Owner name
	2021-03-23	First draft	K. Simon Krughoff
	2021-09-21	Draft for review	K. Simon Krughoff
	2021-09-27	First published version	K. Simon Krughoff
	2021-09-27	Added completed tag for the test plan	K. Simon Krughoff

Document curator: K. Simon Krughoff
Document source location: `https://github.com/lsst-dm/DMTR-291`
Version from source repository: 9341d87

Contents

DM-503-EFDa: EFD on Summit for M1/M3 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 Engineering Facility Database (EFD) as implemented with Kafka, InfluxDB and Chronograf. This plan will describe the prerequisites for beginning a test campaign, step by step instructions for each test can 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.

The highest level description of this test plan is to run the M1/M3 subsystem in an active state for no less than 5 contiguous days. During this time, all telemetry produced by the M1/M3 subsystem will appear in the InfluxDB instance running at the summit with latency less than 4 seconds 99% of the time. The maximum latency shall be less than 20 seconds. All telemetry shall also be available for interrogation by Chronograf on similar time scales. Any gap in telemetry or dropped/missing messages will be considered a deviation. Successful completion of the test campaign will show that:

1. users are able to access data ingested in the InfluxDB at the summit in near real time from the Chronograf interface
2. the latency from message production to ingestion in InfluxDB is less than the nominal limit (4 sec) 99% of the time and never more than 20 seconds during nominal operation
3. the M1/M3 telemetry is successfully being mirrored to another influxDB instance at the data facility with latency less than 24 hours under normal circumstances (e.g. no network

outages). Though the requirement flowed down from project and data management subsystem requirement documents is 24 hours, this exercise will quantify the realized latency. We expect the typical latency to be less than 20 seconds. With the maximum latency being less than 5 minutes.

4. users are able to access and analyze telemetry data from the M1/M3 subsystem from notebooks running in the notebook aspect of the RSP both at the summit and at the data facility

1.2 System Overview

The tests will be carried out from within an instance of the notebook aspect of the RSP running at either the summit or the data facility. 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-P78 Jira Test Plan and related Test Cycles (LVV-C163).

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 3, 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-P78 in Jira is **Completed** .

1.4 References

2 Test Plan Details

2.1 Data Collection

Observing is not required for this test campaign.

2.2 Verification Environment

The environment will be largely within notebooks running a modern stack, though for the first test case and interactive environment is necessary. In that case commands will be executed with a notebook. The results of those commands will be viewed in the summit version of the chronograf interface.

2.3 Entry Criteria

1. Before beginning this test, as 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 typical latency of the system is less than 4 sec for a pre defined set of topics
   (c) The summit data is being replicated to the instance at NCSA
   (d) Chronograf is set up and running at both the summit and NCSA
2. The summit network and Kubernetes cluster are performing nominally
3. The EFD is deployed in the summit Kubernetes cluster
4. The M1M3 sub-component is reliably producing telemetry via Kafka producers with correct versions of the schema
5. The notebook aspect of the RSP is deployed in the summit Kubernetes cluster
6. The summit EFD is reliably replicated to an EFD instance running in a data facility
7. The notebook aspect of the RSP is deployed in the same data facility as that running the replicated EFD
8. The most recent version of the EFD client python modules are installed in the various deployed notebook aspects
9. A requirement test that the system demonstrates reliability of greater than 99.9%

2.4 Related Documentation

Jira Attachments

To LVV-C163 results	LVV-T2111-report.pdf
To LVV-C163 results	Slew_with_M1M3.ipynb
To LVV-C163 results	mt_utils.py
To LVV-C163 results	LVV-T2112.ipynb
To LVV-C163 results	summit_M1M3_forceActuatorData.log
To LVV-C163 results	summit_M1M3_heartbeat.log
To LVV-C163 results	summit_M1M3_forceActuatorData(1).py
To LVV-C163 results	summit_M1M3_heartbeat(1).py
To LVV-C163 results	summit_M1M3_heartbeat(2).py
To LVV-C163 results	LVV-T2117.ipynb
To LVV-C163 results	sept_hbeat.log
To LVV-C163 results	LVV-T2116.ipynb
To LVV-C163 results	LVV-T2115(1).ipynb

All documents provided as attachments in Jira are downloaded to Github and linked here for convenience. However, since they are not properly versioned, they should be considered informal and therefore not be part of the verification baseline.

2.5 PMCS Activity

Primavera milestones related to the test campaign:

- DM-503-EFDa

3 Personnel

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

	T. Plan LVV-P78 owner:	**Simon Krughoff**	
	T. Cycle LVV-C163 owner:	**Simon Krughoff**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T2111	Simon Krughoff	Simon Krughoff	
LVV-T2112	Simon Krughoff	Simon Krughoff	
LVV-T2117	Simon Krughoff	Simon Krughoff	
LVV-T2116	Simon Krughoff	Simon Krughoff	
LVV-T2115	Simon Krughoff	Simon Krughoff	

4 Test Campaign Overview

4.1 Summary

T. Plan LVV-P78:	**DM-503-EFDa: EFD on Summit for M1/M3**			Completed
T. Cycle LVV-C163:	**DM-503-EFDa: EFD on Summit for M1/M3**			Done
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T2111	2	Pass		
LVV-T2112	1	Pass		
LVV-T2117	1	Pass		
LVV-T2116	1	Pass		
LVV-T2115	1	Pass		

Table 3: Test Campaign Summary

4.2 Overall Assessment

Not yet available.

4.3 Recommended Improvements

Not yet available.

5 Detailed Test Results

5.1 Test Cycle LVV-C163

Open test cycle *DM-503-EFDa: EFD on Summit for M1/M3* in Jira.

Test Cycle name: DM-503-EFDa: EFD on Summit for M1/M3
Status: Done

5.1.1 Software Version/Baseline

Not provided.

5.1.2 Configuration

Not provided.

5.1.3 Test Cases in LVV-C163 Test Cycle

5.1.3.1 LVV-T2111 - Access to M1/M3 telemetry data in near real time via the Chronograf interface

Version **2**. Open *LVV-T2111* test case in Jira.

Show that users can get access to visualizations in Chronograf with telemetry arriving in less than 5 seconds from a command run on the DDS network.

Preconditions:
See prerequisites in the Test Plan LVV-P78

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Log in to whatever VPNs are necessary to both see Chronograf at the NCSA test stand (NTS) and the control network necessary for commanding components fo the M1/M3 subsystem

Expected Result

VPN connects are live

Actual Result

In the attached video VPN login begins at 00:00 and finishes at 00:45. The Cisco VPN client is configured to attach to the VPN server at NCSA. After entering my Kerberos password and "push" as the second password, a 2FA challenge arrived on my phone which was answered. Note that the login process took longer than it normally does. This is attributed to load on the computer since it was simultaneously doing the screen recording.

Step 2 Step Execution Status: **Pass**

Description

Log in to chronograf running at the NTS. The endpoint is currently https://lsst-chronograf-nts-efd.ncsa.illinois.edu, though https://sqr-034.lsst.io shall be considered the primary source of truth for service endpoints relating to the EFD

Expected Result

The browser showing the front page of the chronograf interface

Actual Result

At 01:10 in the attached video I navigate to the Chronograf URL, namely https://lsst-chronograf-nts-efd.ncsa.illinois.edu/. Login is via github authentication. In this case, my access token was already cached by my browser. If it had not been, there would been a login cycle through the GitHub authentication flow.

Step 3 Step Execution Status: **Pass**

Description

Open the dashboard labeled "Test Case 1 LVV-P78"

Expected Result

There should be several panes all showing a different trace for the time window of now() - 1hr

---

Actual Result

At 01:25 I have navigated to to the dashboard created for this test case: “Test Case 1 LVV-P78”. The six panels show up. Some have data in them and some of them do not because the time window is the past 15 minutes. Chronograf remembers the time window you last specified, so we did not get the 1hr default. I do not believe this is a deviation since the intent was simply to show the dashboard in the default state which is what was done on my system.

Step 4 Step Execution Status: **Pass**

Description

Set the refresh in chronograph to 5 seconds and adjust the time window to show the past 15 minutes of data. Both of these operations are done using buttons immediately above the graph window on the right side

Expected Result

- The trace is now the last 15 minutes and the view refreshes every 5 seconds
- The query used to produce the trace plots will be captured as an artifact

Actual Result

These settings are applied at 01:25 - 01:50 in the attached video. In consultation with Michael Reuter, we decided to make the time window 5 minutes instead of 15 minutes since the test only lasts about 5 minutes. This allows us to see more detail in the traces as they scroll by. The suggested refresh rate of 5 seconds was preserved.

Step 5 Step Execution Status: **Pass**

Description

Issue a series of commands to the M1M3 subsystem via a notebook.

Expected Result

- Confirmation that the command was acted upon
- This will also include a qualitative description of whether the command produced the expected change in telemetry.
- The notebook executing the commands will be captured as an artifact

Actual Result

At 01:50 control was handed over to Michael to execute the commands in his notebook. Through the duration of the test, several commands are issued. See the attached notebook for the exact commands. The telemetry observed in Chronograf tracked with the expected values. The one oddity is that the "Hardpoint Corrections" command reports "false" when they are turned on. This is a convention used with some CSCs where the value in the command is not important, just that the command was issued.

Note that there are some convenience functions used in the notebook that are defined in the included helper library.

Step 6 Step Execution Status: **Pass**

Description

Verify new values are showing up in the summit instance of InfluxDB by observing that values with newer timestamps are appearing in the Chronograf visualization

Expected Result

Record the session using a screen capture tool

Actual Result

From 01:50 to 07:25 Michael continues to execute the notebook. Throughout, the telemetry tracks with what is being commanded by the notebook. We can see the traces update almost immediately modulo the 5 second refresh rate throughout. There was no noticeable lag at all.

Step 7 Step Execution Status: **Pass**

Description

Produce a report for the test

Expected Result

- A document explaining the procedure including the topics monitored, the command issued and the query used to do the visualization

Actual Result

The document has been attached to the test case. Note that the video was too large to attach to the final report because of GitHub hosting limits. A link to the video is available here.

5.1.3.2 LVV-T2112 - Latency from production to ingestion and telemetry can be analyzed via notebooks: High and Low Cadence

Version **1**. Open *LVV-T2112* test case in Jira.

Measure the latency of production to ingestion in the EFD and show it is less than 4 seconds 99% of the time.
Show that this analysis can be completed via a notebook running in an instance of the notebook aspect of the RSP.

Preconditions:
See prerequisites in the Test Plan LVV-P78

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Log in to whatever VPNs are necessary to access to the summit notebook aspect of the RSP

Expected Result
VPN connection is active

Actual Result
I am logged into the summit VPN via the tunnelblick client using my IPA account credentials.

Step 2 Step Execution Status: **Pass**

Description
Log in to the summit notebook aspect: https://summit-lsp.lsst.codes/nb
Make sure to choose a recent weekly and a large instance

Expected Result

The JupyterLab interface is displayed in the browser

Actual Result

I have logged into the summit instance of the nublado service using the above URL. I am using the following image with a large container: Weekly 2021_37 (SAL Cycle 0021, Build 005)

Step 3 Step Execution Status: **Pass**

Description

Open a notebook:

1. Navigate to the File->New->Notebook
2. When prompted, select the LSST kernel

Expected Result

An empty notebook running in the LSST kernel

Actual Result

I have open in the JupyterLab interface a notebook running the LSST kernel as indicated in the upper right and in the “Idle” state as indicated in the bottom status bar.

Step 4 Step Execution Status: **Pass**

Description

Connect to the summit EFD

Example Code

```
from lsst_efd_client import EfdClient
efd = EfdClient('summit_efd')
```

Expected Result

A notebook with an instance of the ‘EfdClient‘ configured to talk to the summit EFD

Actual Result

It turns out we couldn’t do this study via a direct query of the EFD because of technical reasons I will explain in a

later step, but we do still need to use the EFD to query for periods of time where the MTM1M3 subsystem was down for one reason or another. Thus, it is still important for us to set up a connection to the summit EFD at this point.

I now have, in my notebook, an instance of the EfdClient class connected to the "summit_efd" instance.

Step 5 Step Execution Status: **Pass**

Description

Choose a topic to query and select a 5 day window of data. The topic and window are arbitrary, but it shall be explicit (not relative to now()) so that it can reproduced. The topic shall be one of high enough cadence that there are many measurements for the topic, i.e. not a command or log topic that could potentially be very sparsely populated over the 5 day window. A high cadence, ideally greater than 20 Hz, topic will give the most precise measurement of the distribution of the latency.

Expected Result

- A table-like object in memory containing data from the chosen topic and time window.
- The window and topic are artifacts to be preserved

Actual Result

We chose the 'lsst.sal.MTM1M3.forceActuatorData' topic since it is sampled at 50 Hz natively. It is also a particularly wide topic, so represents one of the topics with the most throughput in the MTM1M3 subsystem. The time window chose is 2021-05-24T21:43:31.853 to 2021-05-29T21:43:31.853 in TAI. This odd chose for time window is explained in the following step.

I now have a numpy array with the various timestamps necessary to compute the latency in memory in my notebook.

Step 6 Step Execution Status: **Pass**

Description

1. The total latency is the time from the message being published, private_sndStamp, and when it is ingested in the influxDB database. Currently the index timestamp is private_sndStamp There will be an additional field added to the measurement to reflect the ingest timestamp. This may be technically difficult, in which case a subset of topics will have an extra column added by hand for the purpose of this analysis. Care shall also be taken to ensure the messages are in the same (TAI) time system. In the past some CSCs have been

reporting TAI and some report UTC. Currently, the difference is 37 seconds.
2. Compute the total latency by taking the difference of the two columns, arr['timestamp'] - arr['private_sndStamp'] (this is where correction for TAI/UTC would be included if necessary). The result shall be in seconds.

Expected Result

An array-like object in memory containing the latency in seconds for every message in the window

Actual Result

It turned out that it was too hard to add an actual field containing the ingestion time of each message. This has to do with the way data is transferred from kafka to influxdb. Instead, we chose an alternative approach where we used the EFD client to poll particular topics looking for the last ingested message. For very high frequency data, this is a proxy for the ingest time for that message. For lower frequency topics, the resolution of measured latency is limited by the topic frequency.

We ran the process querying the EFD for the most recent messages for the duration of the nominal operating window of 5 days and then beyond. The timestamps from the messages are logged into a file on disk along with the machine time immediately before doing the query and immediately after the query. This log file is then imported into the notebook to provide the data used to compute the latencies presented here.

In addition to the log data, it's necessary to account for gaps in the telemetry for two reasons:

1. There were several times faults occurred. We simply mask over those time windows by inspecting the EFD for the appropriate states. Any time the state changes to something other than ENABLED, we mark the beginning of a gap in the telemetry. When the state changes back to ENABLED, we mark the end of that gap.
2. On two occasions services went down without the normal warning in the state. Once it was a hard crash of the MTM1M3 system that prevented changing into a non-ENABLED state before it went off line. The other was when my polling script was killed because of maintenance on nublado at the summit. Both of these gaps in the data were recorded and are applied in the same way the EFD defined gaps are

There is one more filtering step that is necessary. In some instances, the query arrives in influxDB before a message has been completely ingested. This results in the log file containing NaN values for the missing information. I have spot checked and confirmed that the data in influxDB is consistent, but these data are not useful for computing latencies. This is a rare occurrence and does not impact the statistics of the analysis.

We took the beginning of the analysis window to simply be the first message in the log file produced by the polling script.

I now have two arrays in memory. Once containing the time for every message and one containing the latency for every message in seconds.

See the attached script, summit_M1M3_forceActuatorData.py, for the exact script run to record these data.

Step 7 Step Execution Status: **Pass**

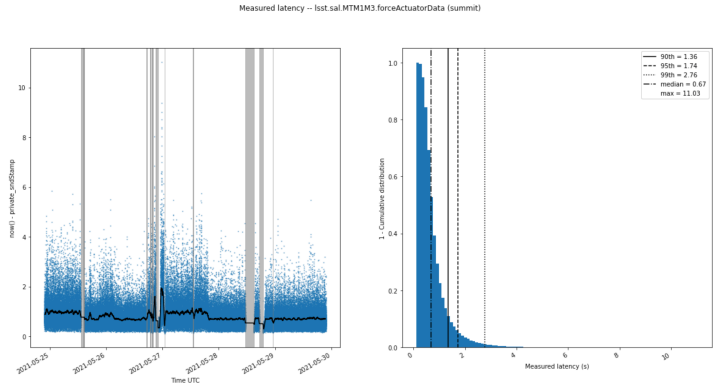
Description

Count the number of entries less than 4 seconds and divide that by the total number of entries. This value shall be greater than or equal to 99%.

Expected Result

- A plot showing a histogram of the latency values indicating the 99% value.
- If the 99% latency is greater than 4 sec, an explanation shall be supplied describing why the latency is higher than expected more often than expected
- The plot shall also indicate the maximum latency observed
- If the maximum latency is above the nominal maximum (20 sec), an explanation shall be provided

Actual Result



The attached plot shows the distribution of latency as a function of time over the 5 days on the left and shows the distribution of latency values on the right. The bars are the gaps. Data from those regions was excluded. The median, 90th percentile, 95th percentile, and 99th percentile are plotted in the right pane as vertical lines. This shows that we meet the less than 4 seconds 99% of the time. 4 seconds represents the 99.8th percentile for this dataset which is greater than the 99% required by this test. Maximum latency is 11.03 seconds.

These values all meet the expected results, so additional explanation is necessary.

Step 8 Step Execution Status: **Pass**

Description

Choose a topic to query and select a 5 day window of data. The topic and window are arbitrary, but it shall be explicit (not relative to now()) so that it can reproduced. The topic shall be low latency in order to to measure the impact of publishing low and high latency topics simultaneously

Expected Result

- A table-like object in memory containing data from the chosen topic and time window.
- The window and topic are artifacts to be preserved

Actual Result

For the low latency topic, we choose the "lsst.sal.MTM1M3.logevent_heartbeat" which is an aliveness topic that publishes a message with minimal content every second. We choose the same time window as for the high frequency topic: 2021-05-24T21:43:31.853 to 2021-05-29T21:43:31.853 in TAI.

I have read the logging file into memory and have a numpy array with the timestamps.

Step 9 Step Execution Status: **Pass**

Description

1. The total latency is the time from the message being published, private_sndStamp, and when it is ingested in the influxDB database. Currently the index timestamp is private_sndStamp There will be an additional field added to the measurement to reflect the ingest timestamp. This may be technically difficult, in which case a subset of topics will have an extra column added by hand for the purpose of this analysis. Care shall also be taken to ensure the messages are in the same (TAI) time system. In the past some CSCs have been reporting TAI and some report UTC. Currently, the difference is 37 seconds.
2. Compute the total latency by taking the difference of the two columns, arr['timestamp'] - arr['private_sndStamp'] (this is where correction for TAI/UTC would be included if necessary). The result shall be in seconds.

Expected Result

An array-like object in memory containing the latency in seconds for every message in the window

Actual Result

We follow the same process as step 6 for filtering and producing the latency values. Note that because the latency of the topic is 1 second and the polling happens every second, the uncertainly in the measured latency can be up to a second.

See the attached script, summit_M1M3_heartbeat.py, for the exact script run to record these data.

Step 10 Step Execution Status: **Pass**

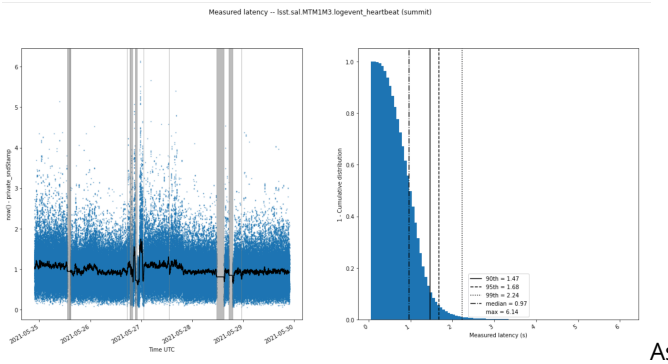
Description

Count the number of entries less than 4 seconds and divide that by the total number of entries. This value shall be greater than or equal to 99%. This is intended to show that even when publishing simultaneously with high and low cadence topics we still meet the latency goal for both topics

Expected Result

- A plot showing a histogram of the latency values indicating the 99% value.
- If the 99% latency is greater than 4 sec, an explanation shall be supplied describing why the latency is higher than expected more often than expected
- The plot shall also indicate the maximum latency observed
- If the maximum latency is above the nominal maximum (20 sec), an explanation shall be provided

Actual Result



As expected, the median of the distribution is around 1 second. This is because of the uncertainty in the measurement of the latency. Taking this into account, the scatter for the low frequency topic is significantly smaller than the high frequency topic. It also easily fits in the 4 second envelope. 4 seconds is the 99.98th percentile for this distribution. The maximum latency over this window is 6.14 seconds.

Step 11 Step Execution Status: **Pass**

Description

Document the procedure including latency distributions, time window, and both high and low cadence topics

Expected Result

- A document describing the process including the topic chosen and the time window.
- The document shall be in the form on a notebook with saved outputs, or an instance of an nbreport.

Actual Result

See the attached notebook. The associated data files are also necessary to run the notebook to completion.

The scripts used to do the logging are also included as attachments to this test script.

5.1.3.3 LVV-T2117 - Latency from production to ingestion and telemetry can be analyzed via notebooks: Low Cadence

Version **1**. Open *LVV-T2117* test case in Jira.

Measure the latency of production to ingestion in the EFD and show it is less than 4 seconds 99% of the time.
Show that this analysis can be completed via a notebook running in an instance of the notebook aspect of the RSP.

Preconditions:
See prerequisites in the Test Plan LVV-P78

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Log in to whatever VPNs are necessary to access to the summit notebook aspect of the RSP

Expected Result

VPN connection is active

Actual Result

I am logged into the summit EFD via the tunnelblick client with my IPA credentials.

Step 2 Step Execution Status: **Pass**

Description

Log in to the summit notebook aspect: https://summit-lsp.lsst.codes/nb
Make sure to choose a recent weekly and a large instance

Expected Result

The JupyterLab interface is displayed in the browser

Actual Result

I am logged into the notebook aspect of the RSP via the URL: https://summit-lsp.lsst.codes/nb. I have chosen a large container running the following image version: Weekly 2021_37 (SAL Cycle 0021, Build 005). I now see the JupyterLab interface in my browser.

Step 3 Step Execution Status: **Pass**

Description

Open a notebook:

1. Navigate to the File->New->Notebook
2. When prompted, select the LSST kernel

Expected Result

An empty notebook running in the LSST kernel

Actual Result

I have an open empty notebook running the LSST kernel as indicated in the upper right of the notebook. It is currently in the Idle state according to the status indicator in the bottom bar of the JupyterLab interface.

Step 4 Step Execution Status: **Pass**

Description

Connect to the summit EFD

Example Code

```
from lsst_efd_client import EfdClient
efd = EfdClient('summit_efd')
```

Expected Result

A notebook with an instance of the 'EfdClient' configured to talk to the summit EFD

Actual Result

As discussed in the next cell, the assumption when this test case was written was that we would query the EFD directly for the latency, but for technical reasons, this was not possible. An in situ measurement of the latency was necessary. In other test cases, the EFD client was used to figure out when the system was running by querying the system state, but in this case we specifically wish to observe the system when the M1M3 sub-system may be in states other than "ENABLED".

For these reasons, we do not actually need a connection to an active EFD to successfully complete the rest of this test case. For completeness. We show that it was possible to create the client, even if it's not strictly needed.

Step 5 Step Execution Status: **Pass**

Description

Choose a topic to query and select a 5 day window of data. The topic and window are somewhat arbitrary, but it shall be explicit (not relative to now()) so that it can reproduced. The topic shall be a low cadence topic with reasonably even sampling. I.e. not a command or event topic that could be very sparse and unevenly sampled. The window should be chosen specifically to be during a period where high cadence topics are not publishing at peak, so as to measure the latency of the low cadence topics on a quiescent network.

Expected Result

- A table-like object in memory containing data from the chosen topic and time window.
- The window and topic are artifacts to be preserved

Actual Result

I chose a window over the last 5 days: 2021-09-15T20:24:09.054 – 2021-09-20T20:24:09.054. The start time was simply the time of the first message I logged, and then the window was set for 5 days following that. I was under the impression that the M1M3 sub-system would be fairly quiet now. It turns out that there was some work that happened during my window. This was verified by slack comments in the #summit-announce channel. There wasn't a specific time listed in the slack message, but inspecting the EFD indicates that they started around 09:00 local and finished around 19:30 local. I therefore put a filter over that time to prevent the statistics from being influenced over a period where the system was in active use and therefor not quiescent. That time window is: 2021-09-16T12:00Z – 2021-09-16T20:45Z.

The topic chosen is the same one used in the high latency tests: "lsst.sal.MTM1M3.logevent_heartbeat". The difference here is that we are trying to see how the low frequency topics behave when the high frequency topics are not being driven at their maximum load.

After loading the log file produced by the latency measurement script described in the next step, the notebook now has a table with timestamps from the local machine time and the timestamps reported in the messages for the measurements in the window.

Step 6 Step Execution Status: **Pass**

Description

1. The total latency is the time from the message being published, private_sndStamp, and when it is ingested in the influxDB database. Currently the index timestamp is private_sndStamp There will be an additional field added to the measurement to reflect the ingest timestamp. This may be technically difficult, in which case a subset of topics will have an extra column added by hand for the purpose of this analysis. Care shall also be taken to ensure the messages are in the same (TAI) time system. In the past some CSCs have been reporting TAI and some report UTC. Currently, the difference is 37 seconds.
2. Compute the total latency by taking the difference of the two columns, arr['timestamp'] - arr['private_sndStamp'] (this is where correction for TAI/UTC would be included if necessary). The result shall be in seconds.

Expected Result

An array-like object in memory containing the latency in seconds for every message in the window

Actual Result

As with the other latency measurements in other tests, for technical reasons, we chose to measure the latency in situ, but asking for the system time and comparing to the time the message reports as being sent: "private_sndStamp". This can introduce some uncertainty in how well we can measure the latency based on the

frequency the topic is publishing and the frequency with which the script is polling. In this case the frequency is around 1/second, so without any systematic latency we can expect a maximum uncertainty in any given measurement of one second.

Taking the difference between the system time immediately after the EFD query returns and the "private_sndStamp" gives us a per message latency in seconds. We will use these values to analyze the performance of the low frequency topics while the high frequency topics are producing at their idle state.

The only filtering at this stage is the aforementioned window where the M1M3 was actively being used.

Step 7 Step Execution Status: **Pass**

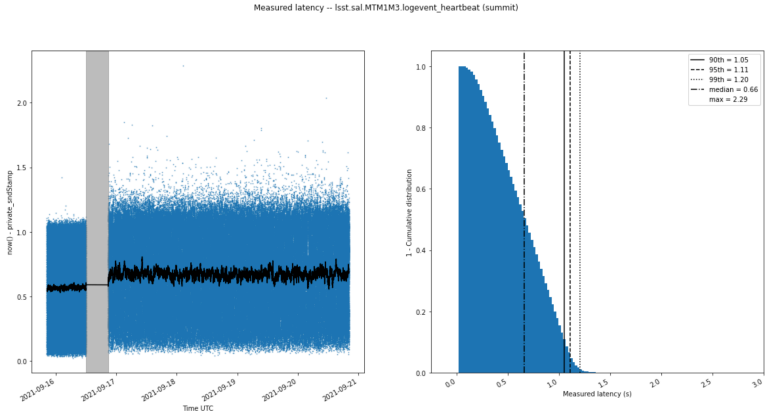
Description

Count the number of entries less than 4 seconds and divide that by the total number of entries. This value shall be greater than or equal to 99%.

Expected Result

- A plot showing a histogram of the latency values indicating the 99% value.
- If the 99% latency is greater than 4 sec, an explanation shall be supplied describing why the latency is higher than expected more often than expected
- The plot shall also indicate the maximum latency observed
- If the maximum latency is above the nominal maximum (20 sec), an explanation shall be provided

Actual Result



As with other plots in this series, the left pane is the latency as a function of time and the right is 1 minus the cumulative distribution. The lines show the median, 90th, 95th and 99th percentiles. As can be seen in the plot, the 99th percentile is 1.24 seconds which

is significantly below the 4 seconds required from this test case. The maximum for this sample is 2.8 seconds. These easily satisfy the pass criteria for this test case.

Step 8 Step Execution Status: **Pass**

Description

Document the procedure including latency distributions, time window, and topics

Expected Result

- A document describing the process including the topic chosen and the time window.
- The document shall be in the form on a notebook with saved outputs, or an instance of an nbreport.

Actual Result

See the attached notebook for implementation of the steps in this test. The script that generated the data file and the data file itself are also attached.

5.1.3.4 LVV-T2116 - Verify telemetry is uninterrupted for 5 days and can be analyzed via a notebook at NCSA

Version **1**. Open *LVV-T2116* test case in Jira.

Test that telemetry is being recorded without missing messages for 5 days. This analysis is to be carried out using NCSA infrastructure.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Log in to whatever VPNs are necessary to access to the NCSA notebook aspect of the RSP

Expected Result

VPN connection is active

Actual Result

Using the Cisco AnyConnect client, I have connected to the NCSA VPN server. I authenticated with my NCSA Kerberos credentials answering the 2FA challenge using the DUO app on my phone.

Step 2 Step Execution Status: **Pass**

Description

Log in to the NCSA notebook aspect: https://lsst-lsp-stable.ncsa.illinois.edu/nb/
Make sure to choose a recent weekly and a large instance

Expected Result

The JupyterLab interface is displayed in the browser

Actual Result

After directing my browser to the appropriate URL, https://lsst-lsp-stable.ncsa.illinois.edu/nb/, and logging in through CILogon using my NCSA Kerberos credentials again and answering the 2FA challenge using the DUO app on my phone, I now see the JupyterLab environment in my browser window.

I chose the most recent weekly: Weekly 2021_37 (sciplat-lab:w_2021_37).

Step 3 Step Execution Status: **Pass**

Description

Open a notebook:

1. Navigate to the File->New->Notebook
2. When prompted, select the LSST kernel

Expected Result

An empty notebook running in the LSST kernel

Actual Result

Instead of going through the File menu, I went the route of using the '+' button above the file browser. The result is the same. I currently have a blank notebook running the LSST kernel as indicated in the upper right. The notebook reports it is in the "Idle" state.

Step 4 Step Execution Status: **Pass**

Description

Connect to the NCSA EFD
Note:The efd identifier is yet to be determined, but shall be ldf_efd or similar.

Example Code

```
from lsst_efd_client import EfdClient
efd = EfdClient('ldf_efd')
```

Expected Result

A notebook with an instance of the 'EfdClient' configured to talk to the NCSA EFD

Actual Result

The actual string that I need to identify the appropriate EFD is "ldf_stable_efd". I was able to find this by using the list_efd_names() method on the EfdClient class.

I now have an efd client configured such that it is connected to the stable version of the LDF EFD.

Step 5 Step Execution Status: **Pass**

Description

Choose a topic to query and select a 5 day window of data. The window is arbitrary, but must be explicit (not relative to now()) so that it can be reproduced. Note that command topics shall be avoided for this purpose since the semantics of private_seqNum are different in the context of commands than for other topic types

Expected Result

A time window and topic name that will be queried for the sequence number

Actual Result

We choose the same topic name used in LVV-T2115. The topic is "lsst.sal.MTM1M3.forceActuatorData". For the time window, we take advantage of the fact that the run was not exactly 5 days. We can show that we have good message intake in a wider window than the strict 5 days by shifting for this test case so the window partially overlaps that of LVV-T2115, but extends beyond the end probed there. the time window starts one day later at 2021-05-26T00:00:00 and continues for 5 days following.

Because of technical difference in how we interact with the database at NCSA vs at the summit, we use a slightly different mechanism to query the database in this test case

---

Step 6 Step Execution Status: **Pass**

Description

Care shall be taken to fix up the id column, private_seqNum. It must be monotonically increasing, but gets reset when the CSC is restarted. When this happens an offset must be applied to the rest of the message ids in order to produce a monotonically increasing sequence

Expected Result

A table-like object in memory with monotonically increasing message numbers

Actual Result

We can tell the message numbers are monotonically increasing because the difference between adjacent ids is never less than one. See cell 5 in the attached notebook.

Step 7 Step Execution Status: **Pass**

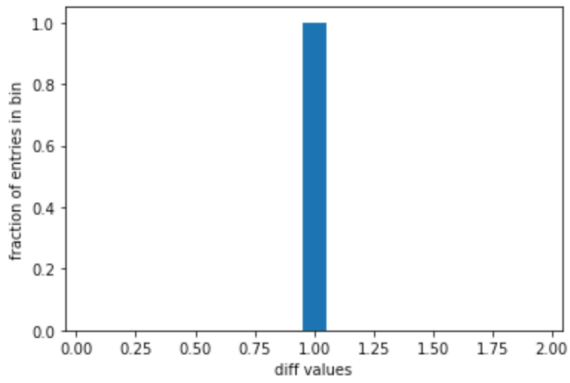
Description

Compute the difference between message ids via diff = arr['private_seqNum'][1:] - arr['private_seqNum'][:-1]

Expected Result

A histogram normalized to the total number of samples from 1 to the maximum number of the array diff.

Actual Result



4The histogram does indeed show that all elements end up in the bin containing the value 1. Even more convincing is that when asking if all the elements in the pairwise difference array are 1, the answer is “yes” (see cell 6).

Step 8 Step Execution Status: **Pass**

Description

Confirm that all values are one. Any larger numbers would indicate a gap in messages

Expected Result

The result is that all differences in the histogram should be 1. Any values larger than 1, which indicates a gap in messages, must be traced to a problem with the CSC and not an issue with the EFD.

Actual Result

There were no gaps in the message IDs so no action was needed.

Step 9 Step Execution Status: **Pass**

Description

Produce a report outlining this test

Expected Result

- A document describing the process including which topics/fields where used and what time window was selected.
- The document shall be in the form of a notebook with saved outputs, or an instance of an nbreport

---

Actual Result
See the attached notebook

5.1.3.5 LVV-T2115 - Verify telemetry is uninterrupted for 5 days and can be analyzed via a notebook at the summit

Version **1**. Open *LVV-T2115* test case in Jira.

Test that telemetry is being recorded without missing messages for 5 days. This analysis is to be carried out using summit infrastructure.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Log in to whatever VPNs are necessary to access to the summit notebook aspect of the RSP

Expected Result
VPN connection is active

Actual Result
I logged in to the summit VPN using the tunnelblick client with my summit IPA credentials. The VPN is connected and active.

Step 2 Step Execution Status: **Pass**

Description
Log in to the summit notebook aspect: https://summit-lsp.lsst.codes/nb
Make sure to choose a recent weekly and a large instance

Expected Result
The JupyterLab interface is displayed in the browser

Actual Result
I have opened the notebook aspect of the RSP by accessing the above URL and using my GitHub credentials. I chose a large instance. The image is "Weekly 2021_37 (SAL Cycle 0021, Build 005)".

Step 3 Step Execution Status: **Pass**

Description
Open a notebook:

1. Navigate to the File->New->Notebook
2. When prompted, select the LSST kernel

Expected Result
An empty notebook running in the LSST kernel

Actual Result
JupyterLab is showing an empty notebook. The string in the upper right says "LSST". The status in the lower left says "Idle".

Step 4 Step Execution Status: **Pass**

Description
Connect to the summit EFD

Example Code
```
from lsst_efd_client import EfdClient
efd = EfdClient('summit_efd')
```

Expected Result
A notebook with an instance of the 'EfdClient' configured to talk to the summit EFD

Actual Result

The example code executed without error and the notebook is once again in the “Idle” phase.

Step 5 Step Execution Status: **Pass**

Description

Choose a topic to query and select a 5 day window of data. The window is arbitrary, but must be explicit (not relative to now()) so that it can be reproduced. Note that command topics shall be avoided for this purpose since the semantics of private_seqNum are different in the context of commands than for other topic types

Expected Result

A time window and topic name that will be queried for the sequence number

Actual Result

We chose a high cadence topic because it should be the most likely to uncover missing messages. The topic is “lsst.sal.MTM1M3.forceActuatorData”. The time window was chosen to overlap the time window used for other test cases in this test cycle for consistency. The chosen time window is 2021-05-25T00:00:00Z + 5 days.

Step 6 Step Execution Status: **Pass**

Description

Care shall be taken to fix up the id column, private_seqNum. It must be monotonically increasing, but gets reset when the CSC is restarted. When this happens an offset must be applied to the rest of the message ids in order to produce a monotonically increasing sequence

Expected Result

A table-like object in memory with monotonically increasing message numbers

Actual Result

After correcting for the fact that the sequence resets to zero when a CSC is restarted, we have a numpy array with monotonically increasing values. See cell 6 for an indication of this. It should not print anything if the sequence is monotonically increasing.

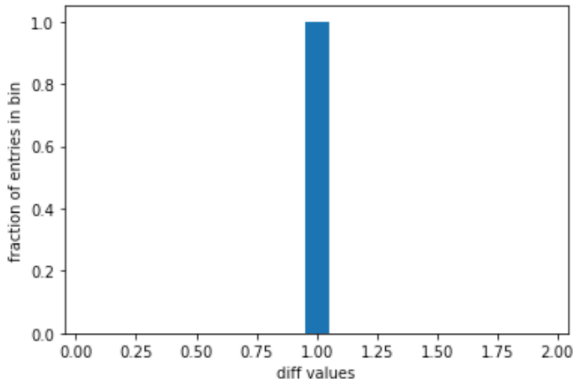
Step 7 Step Execution Status: **Pass**

Description

Compute the difference between message ids via diff = arr[‘private_seqNum’][1:] - arr[‘private_seqNum’][:-1]

Expected Result
A histogram normalized to the total number of samples from 1 to the maximum number of the array diff.

Actual Result
Doing the above calculation results in an array where every entry is one. See cell 10 for the mathematical evidence of this. I.e. subtracting an array of all ones results in an array that sums to zero. We can also show that a histogram of the values has all elements in the bin containing the value 1.



Step 8 Step Execution Status: **Pass**

Description
Confirm that all values are one. Any larger numbers would indicate a gap in messages

Expected Result
The result is that all differences in the histogram should be 1. Any values larger than 1, which indicates a gap in messages, must be traced to a problem with the CSC and not an issue with the EFD.

Actual Result
There were zero instances in the 21373507 entries in our sample that deviated from a value of one, so no further action was required.

Step 9 Step Execution Status: **Pass**

Description
Produce a report outlining this test

Expected Result

- A document describing the process including which topics/fields where used and what time window was selected.
- The document shall be in the form of a notebook with saved outputs, or an instance of an nbreport

Actual Result

See the attached rendered notebook for the report.

A Acronyms used in this document

Acronym	Description
CSC	Commandable SAL Component
DDS	Data Distribution System
DM	Data Management
DMTR	DM Test Report
EFD	Engineering and Facility Database
LDF	LSST Data Facility
LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
LVV	LSST Verification and Validation
M1	primary mirror
M1M3	Primary Mirror Tertiary Mirror
M3	tertiary mirror
MTM1M3	Main Telescope M1M3
NCSA	National Center for Supercomputing Applications
NTS	NCSA Test Stand
PMCS	Project Management Controls System
RSP	Rubin Science Platform
SAL	Service Abstraction Layer
TAI	International Atomic Time
URL	Universal Resource Locator
UTC	Coordinated Universal Time
VE	vendor estimate
VPN	virtual private network

B Traceability

Test Case	VE Key	VE Summary
LVV-T2111		
LVV-T2112		
LVV-T2115		
LVV-T2116		
LVV-T2117		