18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.
Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

SpaceOps-2025, ID # 137

Applying Open-Source Data Science Tools for Accelerated Spacecraft Trending of NASA's James Webb Space Telescope

David Kauffman[a*], Alexander Hunter[b], Amanda Arvai[c]

[a] Senior Software Engineer, *Space Telescope Science Institute, Baltimore, Maryland, United State of America*, *dkauffman@stsci.edu*
[b] Senior Flight Systems Engineer, *Space Telescope Science Institute, Baltimore, Maryland, United State America*, *ahunter@stsci.edu*
[c] Webb Flight Operations Manager, *Space Telescope Science Institute, Baltimore, Maryland, United State of America*, *aarvai@stsci.edu*
* Corresponding Author

Abstract

With a focus on speed, the Flight Operations Team (FOT) for NASA's James Webb Space Telescope (Webb) built upon tools created by the Chandra program to create a novel trending and analysis system for the observatory's forty thousand telemetry points. Quick access to full mission telemetry and integration with Python-based data analysis tools accelerates the pace of garnering insight into the behavior of the observatory. The Lateral Ingest Telemetry Archive (LITA) utilizes a 3-tier architecture similar to traditional web application design. The bottom tier, a data persistence layer, is responsible for archiving and accessing telemetry. The data itself is ingested and archived using HDF5 (Hierarchical Data Format version 5) files with a custom structure that is able to effortlessly handle a large number of data points with fast I/O speed; telemetry data can be fetched at over 6M samples/second. The middle layer is a collection of microservices that include task automation, report generation, a web API (Application Programming Interface), and external data access. Two user-interfaces (U/I), each targeting a different set of users and use cases, serve as the top layer of the LITA system. The first U/I is a Django web application providing rapid access to trending data and plots for simple analysis tasks. The second U/I, based on Jupyter Notebooks [6], provides the LITA core python package (*jeta* [4]) and the full scientific python stack in a browser-based development environment. In this environment, flight operations engineers can make use of the entire Python ecosystem to develop endlessly-flexible notebooks capable of fetching and filtering telemetry, as well as performing complex data analysis and visualization. These tools were quickly adopted by the Webb flight operations team for the full range of analysis tasks, from routine trending to in-depth anomaly investigation. The following paper will delve further into the design and capabilities of the LITA system, along with a description of how it is currently being used and the impact it has on Webb flight operations.

Keywords: Webb, JWST, Trending, Analysis, Data Management

Acronyms/Abbreviations

Hierarchical Data Format version 5 (HDF5), Input/Output (I/O), User Interface (U/I), Application Program Interface (API), Lateral Ingest Telemetry Archive (LITA), Continuous Integration/Continuous Deployment (CI/CD), (JWST), Single Pane of Glass (SPoG), Structured Query Language (SQL), Flight Operations System (FOS), Django Rest Framework (DRF), Free Open-Source Software (FOSS)

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

1. Introduction

This paper intends to describe the design, usage, and scope of the LITA system, an off-line telemetry archive and analysis platform developed for the James Webb Space Telescope (Webb). We start with the motivation and objectives of the system. Next, we detail the system design choices and how those decisions helped meet those aims. We'll continue by reviewing the system performance capabilities, and discuss the tangible impacts LITA has had for Webb flight operations.

1.1 Motivation LITA System Development

The principle goals of the system were to:

1) Support high throughput telemetry ingest and retrieval
2) Provide various means to access telemetry, tailored to individual stakeholder needs
3) Create an extensible platform that can be maintained and adapted for future mission requirements

1.2 Webb Telemetry Description

There are two primary categories of data produced by Webb: science data generated by the instruments during observations and engineering data (i.e. housekeeping telemetry) generated by all spacecraft and payload subsystems which report on their health and status. The LITA system described in this paper is used solely for archiving engineering data.

Telemetry packets are generated onboard the observatory by the spacecraft and payload C&DH per the CCSDS Packet Telemetry specification (CCSDS 102.0-B-5) and then recorded to the Solid State Recorder (SSR), to be available for the next downlink opportunity. The telemetry packet generation is governed by software tables which define the sample rate for each packet type. This allows critical and rapidly-changing telemetry to be sampled frequently in a high-rate telemetry packet, while low-rate telemetry packets are used for less critical and infrequently-changing telemetry. The telemetry sampling period typically ranges from 0.064 s for high-rate telemetry to over 32 s for low-rate telemetry. Additionally, some telemetry packets are generated asynchronously, only as-needed in response to events. The average data rate of the recorded telemetry stream is approximately 100 kb/s.

Data recorded to the SSR is downlinked and the packets are decommutated by the ground system into roughly 36,000 discrete telemetry points, referred to by mnemonics. An additional 4,000 Ground Data Points (GDP) are derived from the telemetered values. Multiple data types are available for use depending on the nature of the telemetry point, including various length signed/unsigned integers, floating point values, and character strings. These 40,000 total telemetry points are archived by the Webb ground system, and also transferred to LITA.

2. History/Chandra

LITA is built off of Chandra's open-source telemetry analysis package *cheta* [12]. *Cheta* was developed by their Flight Director Dr. Tom Aldcroft, one of the core contributors and a founding member of the Astropy Project. Dr. Aldcroft was in the unique position of being both an active working-group member involved in Chandra telemetry analysis as well as a Flight Director, charged with balancing the goals and risks of the observatory. Dr. Aldcroft aptly recognized that telemetry analysis is frequently iterative, interactive, and time-limited. One might not know the right questions at the start, only that something unexpected or undesirable is occurring. Exploring the data may require looking for rare events or correlations that are subtle. As such, Dr. Aldcroft and his team developed the *cheta* telemetry package along with numerous other Python packages comprising the Ska runtime environment. Once in operation, these tools became heavily relied upon by the Chandra Flight and Science Operations Teams.

In addition to quick-loading telemetry (over 20 million data points per second), the Ska toolset includes intuitive, built-in plotting capabilities that allow users to quickly maneuver between years' worth of data to minutes' worth of data. It does this by first loading daily statistical data (min, max, and mean), then increasing to 5-minute statistical data and eventually full-resolution based on the user-defined time range. Even for heavily-sampled data,

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

such as the 4-Hz gyro rates, a user can request a plot of the 25-year mission, identify an unusual spike, zoom in, and get the exact time range all in a matter of seconds.

In adapting Ska to Webb, which has more telemetry points than Chandra, it was identified that it was necessary to modify the approach to telemetry timestamps. Whereas Chandra's telemetry packets occur on a repeatable cadence, with multiple telemetry points sharing a common timestamp, Webb's telemetry contents are not predictable. A modified version of *cheta* evolved into the *jeta* component of LITA.

3. System Design

The design of the LITA system was driven by the objectives as outlined in the introduction. This section reviews the specifics of that design, the trades conducted, and the implementation as it has evolved since the initial release just prior to Webb commissioning. The goal is to demonstrate how the Webb team was able to enhance analysis capabilities by adopting FOSS and an iterative software design model. We will focus the discussion first on the design choices that maximise data throughput into the archive. Next, we'll go on to describe the data access implementation. The system design review will conclude by highlighting the steps taken to achieve high availability by adopting CI/CD and automation. Throughout we will take note that the system used is divided into a 3-tiered design Figure 1 (archive/persistence, service, presentation) to create a logical pipeline from data to user. From those 3-tiers the LITA system is further partitioned into seven major components Table 1 to compartmentalise specific roles and responsibilities. Creating a system in this way not only easies the continued development by isolating changes, but makes it possible to more readily understand individual components.

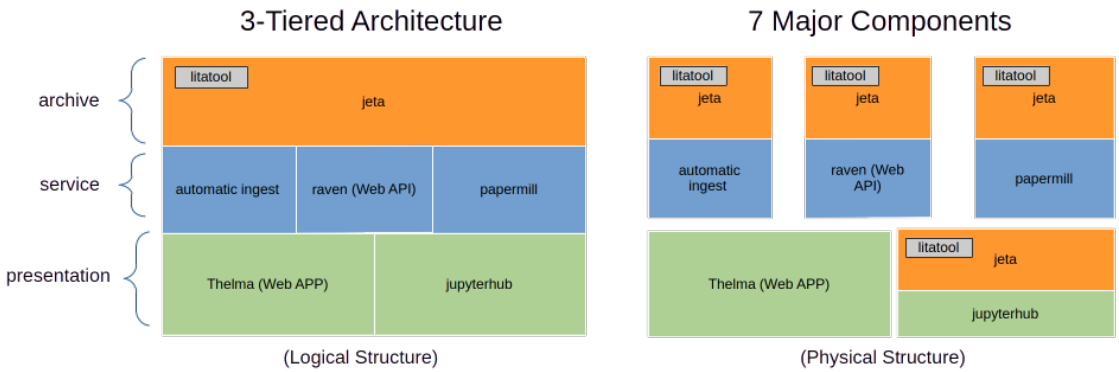

Figure 1: Illustration of logical and physical system design

Table 1: The Seven Major System Components

Major System Components		
name	**Component**	**Description**
jeta	low-level archive interface	This component is the lowest level interface for getting data into (archiving) and out of (fetching) the archive as well as querying archiving state. Core and shared system logic are implemented in this package.
automatic ingest	automated task scheduler	This component provides automated task execution at scheduled intervals. Tasks are scripted as python functions such that any task requiring asynchronous executing is handled by this component.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

Major System Components		
raven	Web API	A RESTful API for fetching telemetry, metadata, and system state via HTTP requests.
thelma	Web UI	A stand-alone web application and interface to the fetch api and system dashboard. Web-based tools are implemented here.
jupyterhub	Web Development Platform	A web-based data science and software development platform
papermill	Automated Notebook Execution	This component provides scheduled execution of specifically Jupyter notebooks for daily/weekly trending reports.
lita-tools	System Management Tool	A command-line tool for performing system operations from adding new users to system recovery.

3.1 Telemetry Archive

3.1.1 Processing and Persistence

As stated, one of the requirements for the system was the capability to support high throughput data. The choice of HDF5 for persistent storage provides significant performance benefits that helped satisfy that requirement. The access and write speeds for HDF5 are orders of magnitude faster than SQL for example, but that speed is traded for the complexity of managing the files and querying the data. Fortunately, the task of managing some of that complexity was already resolved by the source code provided by the team at Chandra. The difference in scale between Chandra and Webb presented different challenges that required novel engineering to resolve. Ingest is where the LITA system begins and is the interface point with the Flight Operations System (FOS), a component of Webb's ground system. The ingest files are delivered from the FOS to a file staging area. An automated task scheduled in celery [2] periodically examines that area, detects new files, and vets files for ingest. The ingest files themselves were designed in partnership with the team from Raytheon, who developed the FOS, and are also HDF5 files. We'll note here that in the Chandra code base, ingest files were in FITS format, and that the ingest routines were re-written entirely to be tailored for this application. On Webb, each ingest file covers approximately 28 mins of telemetry with samples from potentially every mnemonic. One of the early challenges to overcome was how to efficiently process the roughly 10 million samples per file. From the original ConOps our telemetry is required to be written out in ordered sequence to maintain the I/O benefits of HDF5, which itself presents more challenges in the form of constraints. The solution combines the power of numpy, pandas, h5py, and pytables. These are all open-source python libraries. Preprocessing the telemetry prior to inserting or appending the data to the archive was achieved using numpy and pandas vectorization that allows concurrent manipulation of vector arrays. The python library h5py was also instrumental in achieving high ingest throughput by employing the concept of a Virtual Dataset [11]. This allowed the ingest code to logically span multiple ingest files and present them to numpy and pandas as a single array. That made it possible to process multiple ingest files at once. The I/O time for appending one file vs. six files worth of samples is negligible with HDF5. Thus, with the coupling of the python data science stack for data processing and HDF5 for I/O, the ingest routine was able to achieve an average throughput 10x realtime. Of course, ingesting speed is not the only concern. Other challenges in this early part of the system centered around data integrity. As mentioned above, data needs to be introduced or appended to the archive in time-dependent order, but also without unexpected gaps or duplicates. The LITA system handles this with preprocessing routines and enough automated intelligence when examining the staging area to reject ingesting until those constraints are met. Procedure are in place to enforce data validation, at the file and individual mnemonic levels. Employing these techniques gives all stakeholders a measure of confidence that data in the archive is in order and correct.

As was the case with a change from FITS to HDF5 for ingest file format, the archived files format also differs from the Chandra code base. In the Chandra code base, one file per mnemonic was used to house both the observatory times and the engineering values. For Webb the observatory times and the engineering values are split into two HDF5

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

files with parallel arrays. The engineering values are provided in two data types, either text strings or float64 depending on the mnemonic. Timestamps are stored in Julian Date format. The native compression provided by HDF5 keeps files at a reasonable size on disk. Data is stored in consecutive arrays on disk that are pre-sized when adding a mnemonic to the archive. In addition to the two files that track the telemetry points, the system uses the concept of an index file. The index file contains reference points for corresponding time indexes in the times files. This scheme allows the *fetch* algorithm to quickly find the closest index for a subset of data points and extract just that section of data. Combined, these three archive files per mnemonic make up the full resolution telemetry for Webb in LITA. The LITA system derives two more data products from the full resolution data that complete the archived telemetry data. For all engineering telemetry mnemonics, statistics at 5min and daily intervals are calculated using an automated celery task that runs every eight hours. All of the logic for ingesting and archiving rests at the archive layer of the system and is implemented in the *jeta* component. The specific submodules for handling ingest and archive management are *jeta.archive.ingest*, *jeta.ingest.status*, *jeta.archive.operations*.

Table 2: Submodules For Ingesting Telemetry And Managing The Archive

Submodule	Description
jeta.archive.ingest	contains all the routines for preprocessing ingest files and appending data to the archive
jeta.archive.operations	contains the set of routines for configuring and managing the archive
jeta.archive.status	contains all the routines for querying archive state information

3.1.2 Observation Plan Database Interface

Webb science activities operate with an onboard observation plan (OP). The planning and scheduling branch of the Webb Science Operations Center (SOC) develops this observation plan and maintains a record of scheduled and executed visits in a database.

An interface between LITA and this database was developed. This interface allows LITA users to fetch the scheduled and executed observation plan, filtering by date, primary instrument, and/or observation program. The interface runs as a service, receiving structured requests from user code and returning the results of the database query. This interface allows telemetry analysis to be performed in the context of the onboard science execution, without requiring LITA to maintain a copy of this data in the telemetry archive.

3.2 Data Access

In the original ConOps for the LITA system users would access telemetry using a networking protocol such as SSH to remotely access the telemetry server. Once logged in they could activate a Python environment with *jeta* installed to perform their work. Alternatively, users could execute a series of scripts on a laptop or workstation and build the tool environment, then download telemetry locally. It was quickly determined that the amount of data generated for the Webb mission would make it impractical to download. It also creates an issue with having multiple copies of the archive out of sync. The solution was to employ a “cloud-like” approach and avail both telemetry and environment directly in a web browser. This method eliminates the task for users to install, configure, or download anything, and instead can use the tools like an app. The remainder of this section gives some details about how data access percolates up the system layers from the low-level *jeta* package to the browser.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

3.2.1 Low-level Package

As mentioned, the lowest level of data access is the *fetch* interface in the *jeta* package. The fetch API provides a single system interface for data retrieval upon which all subsequence data access is built. The class responsible for doing the heavy lifting is the MSID class that essentially models a mnemonic as a python object. Each service in the LITA system has access to the *fetch* interface, as each service is individually built on top of a *jeta* image

A general example of fetching the full resolution telemetry for the life of the mission of a mnemonic is as follows:

```
mnemonic = fetch.MSID(msid='ZGLD_ACT1')
```

If instead we wanted to just retrieve the 5mins statistics for a closed coverage period (first 30 days of 2025) for that same we could do this:

```
mnemonic = fetch.MSID(
    msid='ZGLD_ACT1',
    start='2025:001'
    stop='2025:030',
    stats='5min'
)
```

3.2.2 WebAPI

The Web API service extends the *jeta* interface by layering an HTTP interface on top and transforming the result into JSON. An indirect query to *jeta* can then be made using an HTTP client such as curl, the python requests package or even a web browser. From there the system can then support any number of downstream services. One such service is Thelma described in the next section. To query the same information as the last example above, but with the Web API instead the following HTTP request could be made to the backend:

```
HTTP Request:

https://lita-
api/api/v1/fetch/stats?msid=ZGLD_ACT1&interval=5min&start=2025:001&stop=2025:030

HTTP Response:

stats: {
   times: [...],
   values: [...],
   mins: [...],
   maxes: [...],
   means: [...],
   midvals: [...],
   interval: "5min",
   tstart: "2025:001:00:00:00",
   tstop: "2025:030:23:59:59"

}
```

3.2.3 Thelma

Thelma is a web application designed to provide a simple interface for accessing telemetry and displaying trends. The app itself is implemented using a python-based web framework, Django [9], to handle the http request/response part of the client-server model. To make the application more dynamic and include some real-time interactions the UI/UX portion of the app is built using JQuery, VueJS, and Bootstrap.

Thelma incorporates some of the ideas from DigitalGlobe [5], specifically the notion of a Single Pane of Glass (SPoG). The idea is to have a single-entry point for accessing the system instead of creating one-off tools. To this end, Thelma includes both a page for viewing automated reports and access to JupyterHub. Future tools and data will also be made available through this interface. This single point of entry unifies the system and simplifies on-boarding for new users.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

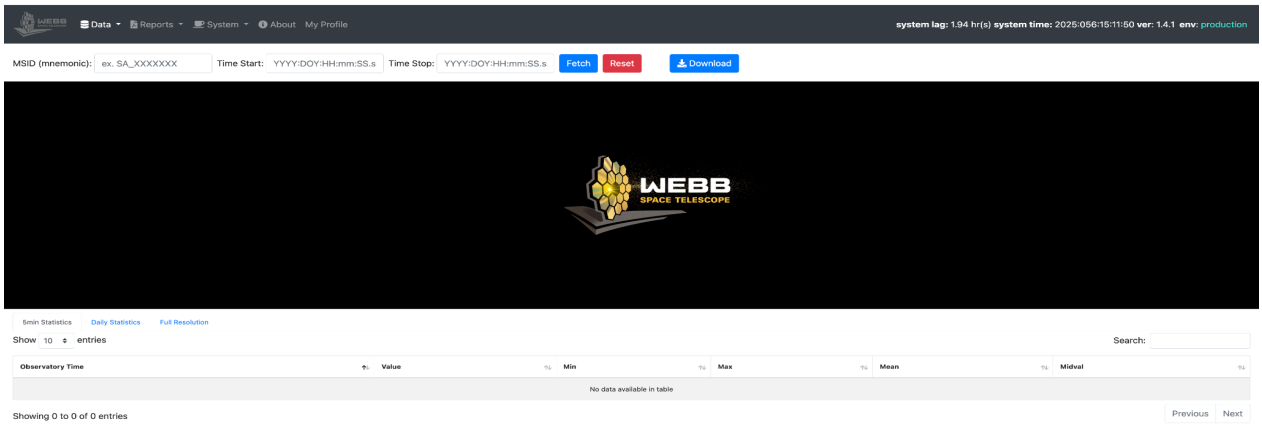

Figure 2: Thelma Landing Page

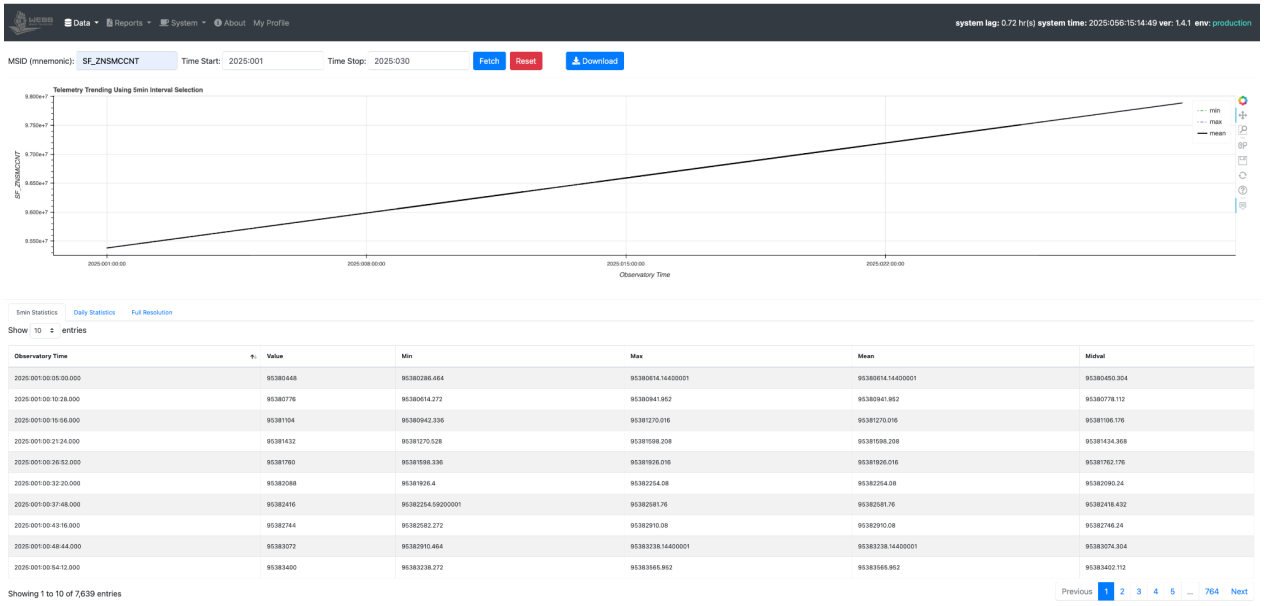

Figure 3: Simple Fetch Example

3.2.2 JupyterHub

Jupyter Notebooks allow users to merge text, images, video, code, data, interactive plots, and mathematical notation into a single portable document format. JupyterLab is a web-based fully integrated development environment for Jupyter Notebooks. The environment includes support for many different language kernels, although as of this writing the LITA implementation only includes the standard python kernel. JupyterHub brings the JupyterLab environment to entire groups of users allowing the team to centralised tools and data access.

In our deployment users are provided a workspace that allows them to organize their work and develop either notebooks or python source to accomplish complex analyses using multiple data sources. In this environment the end users have access to the *jeta.fetch*. Once users have fetched data using the provided API they then can make use of the entire python data science ecosystem for data analysis and visualization tasks. Users have the ability to maintain their notebooks in team-wide version-controlled GIT repositories, which facilitates sharing and collaboration.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

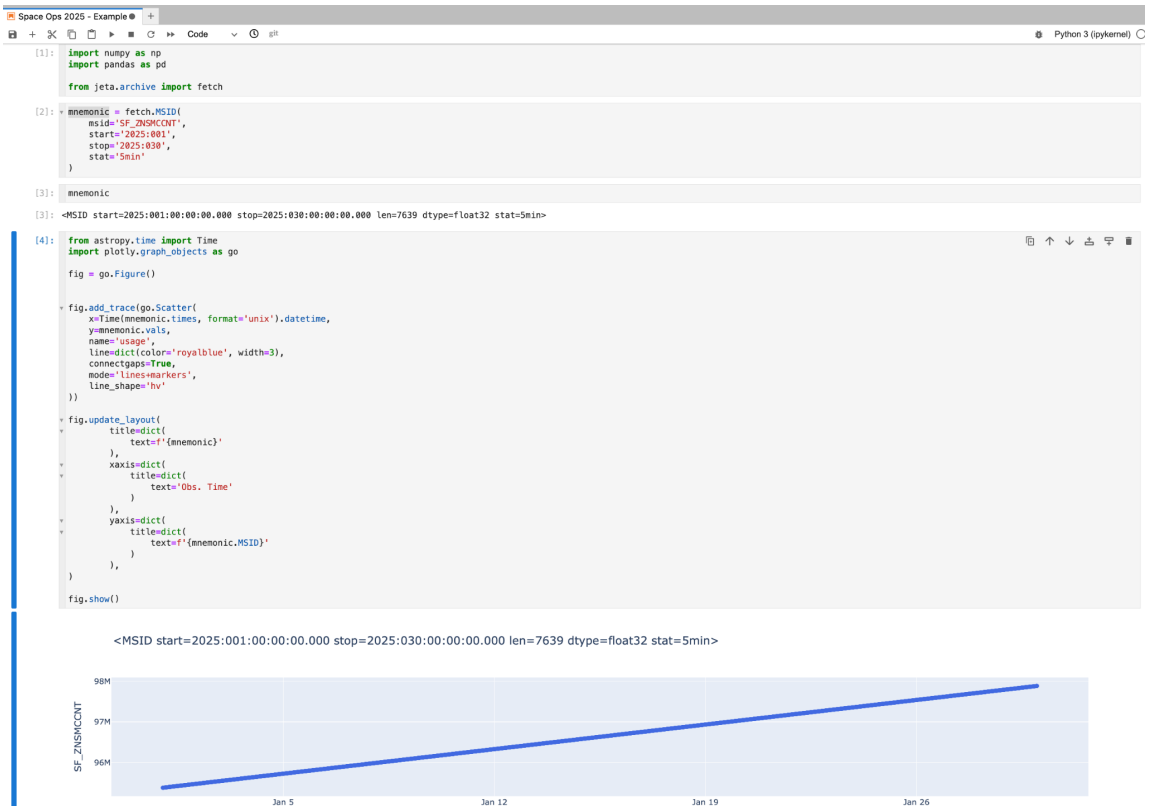

Figure 4: JupyterHub Basic Fetch Usage Example

3.2.3 Automated Subsystem Reports

In order to automate routine tasks, a service was implemented to schedule execution of Jupyter notebooks. This service uses the *papermill* python library to run specific notebooks that have been registered with the service for scheduled daily or weekly execution. A cron job is used to schedule the papermill task. The output of the notebook execution is accessible to users via the Thelma web interface, providing quick access to generated telemetry reports.

3.3. System Maintenance and Reliability

3.3.1 Configuration Management

Configuration management is key to the goal of data availability. The entire system has hundreds of software dependencies with python, javascript, the operating system, and other software packages composing the system. A system for managing external dependencies and the system source was developed in order to keep the LITA stable, reliable, and reproducible. Keeping these packages up-to-date is also a priority, not only to mitigate security vulnerabilities, but to benefit from enhancing the feature sets and performance. The codebase for LITA, meaning each package or service developed in house, is managed in version control using git. Further, for each of the major service components a self-contained image using Docker [7] is built and tagged.

3.3.2 DevOps

Internally the project's configuration management strategy is enforced using GitLab. The service images referenced in the previous section are created as part of a CI/CD pipeline with tagged versions stored in a container

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

registry. These tagged images can be automatically or manually deployed to a host, used locally for development, or used to roll back to a particular software version.

3.2.3 System Notifications

To further the effort to keep the system maintained and reliable, a notification system was integrated to address two categories of events. The first are those events related to the system's ongoing development. The build pipelines responsible for creating, testing, and registering deployable images include messages that are delivered to a slack channel dedicated to LITA DevOps. Examples of those types of events might be failed unit tests, failed builds, successful/failed deployments, and other areas specific to development, with an emphasis on errors and failures. Slack uses a pub/sub paradigm, as such developers, users, and other stakeholders can all be notified about the state of development. Likewise, the other category of events relates the state of the currently deployed version of the system. Events in that category are routed to a system alerts channel. Some examples of alert notifications are: an expected ingest file is missing or the archive is out of date relative to real-time.

4. Performance Results

The following metrics are used to assess the performance of the system.

4.1 Ingest rate

The ingest rate describes how quickly new telemetry data is appended to the archive. The ingest consistently operates at 10x the real time data rate. When there is a nominal downlink of eight hours of telemetry from between contacts, this data is typically ingested and available within one hour after the transfer to the LITA server. This rate allows the ingest process to keep pace with data received from the Webb ground system, and allows it to catch up when a large data downlink is received.

4.2. Fetch rate

The fetch rate describes how quickly data from the archive can be retrieved by a user executing a Jupyter notebook. Table 3 below shows typical fetch rates for both low and high rate telemetry points. The fetch rate varies with the timespan retrieved and the number of samples. For larger fetches, rates of 5-7 million samples/sec are typical.

Table 3: Fetch times for low rate and high rate telemetry

	Low Rate Telemetry (32.768 s period)			**High Rate Telemetry (0.064 s period)**		
Requested Timespan	**Samples (Kilo samples)**	**Fetch Duration (s)**	**Fetch Rate (Mega samples/s)**	**Samples (Kilo samples)**	**Fetch Duration (s)**	**Fetch Rate (Mega samples/s)**
1 Hour	0.1	0.014	0.007	56	0.030	1.857
1 Day	2.6	0.008	0.311	1,350	0.209	6.445
1 Week	18	0.010	1.794	9,450	1.656	5.706
1 Month	79	0.014	5.347	40,500	7.019	5.769
1 Year	962	0.136	7.072	492,741	86.431	5.700
3 Years	2,820	0.415	6.796	1,441,624	245.727	5.866

4.3 Statistics Update and Fetch

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

The statistics calculation function runs three times per day to update the pre-calculated statistics for all numerical telemetry points. Statistics are calculated for 5-minute and 1-day intervals. The statistics update function typically completes in under 20 minutes. This includes fetching recent data for each telemetry point, performing the statistics calculation, and then appending the results to the statistics archive.

The pre-calculated statistics facilitate fast retrieval of long-term data. Table 4 below shows example fetch times for both 5-minute and 1-day statistics. Fetching three years of 5-minute statistics can be completed in 71 ms.

Table 4: Fetch rates for per-calculated telemetry statistics

	5-minute Statistics		**1-day Statistics**	
Requested Timespan	**Number of Intervals**	**Fetch Duration (s)**	**Number of Intervals**	**Fetch Duration (s)**
1 Week	1844	0.064	7	0.006
1 Month	7903	0.065	30	0.006
1 Year	96,147	0.065	365	0.006
3 Years	288,371	0.071	1,095	0.006

5. Use Cases and Mission Impact

LITA improves access to historical mission telemetry. This enables tasks that are not feasible using other telemetry archives, such as fetching life-of-mission data for high sample rate telemetry. Data retrievals that can take hours to complete on traditional systems return in seconds using LITA. This ease of access improves visibility into the observatory state of health and expedites anomaly investigation.

The integrated python environment enables a streamlined user experience. When interfacing with LITA from a Jupyter notebook, data retrieval and data analysis are performed in the same environment. This eliminates steps that are required when using traditional telemetry archives, such as submitting a telemetry request, downloading a telemetry report, and then opening the report with the user's local analysis software. The integrated environment improves productivity when performing routine trending or more tailored analysis/investigation by allowing users to quickly iterate. The environment is accessed via a web browser, eliminating the challenges of local software installation and configuration for individual users. The following cases illustrate how LITA is typically used by the Webb operations team.

5.1 Rapid Telemetry Access

Webb operations engineers encounter situations that require quickly referencing telemetry, such as checking the last occurrence of a particular event or the long-term trend of a telemetry point. The Thelma application addresses these use cases with a simple web interface that plots data for a specified telemetry point. This provides a low-overhead method to quickly view telemetry. After navigating to the web page, the user can generate a plot of a telemetry point over a specified time range. By hovering over a data point, the user can inspect precise timestamps and values. Controls allow panning and zooming the plot to pinpoint areas of interest, and the raw data can be downloaded in comma-separated-value (csv) format.

Over longer time frames, greater than 2 days, Thelma plots the pre-calculated statistics at 5-minute intervals. This improves performance, especially when plotting months/years of data. The 5-minutes statistics include the minimum, mean, and maximum value for each 5-minute period, as shown in Figure 5. If desired, after identifying a

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

specific timeframe of interest, the user can request the data for a short time frame to display a plot with every telemetry sample indicated, as shown in Figure 6.

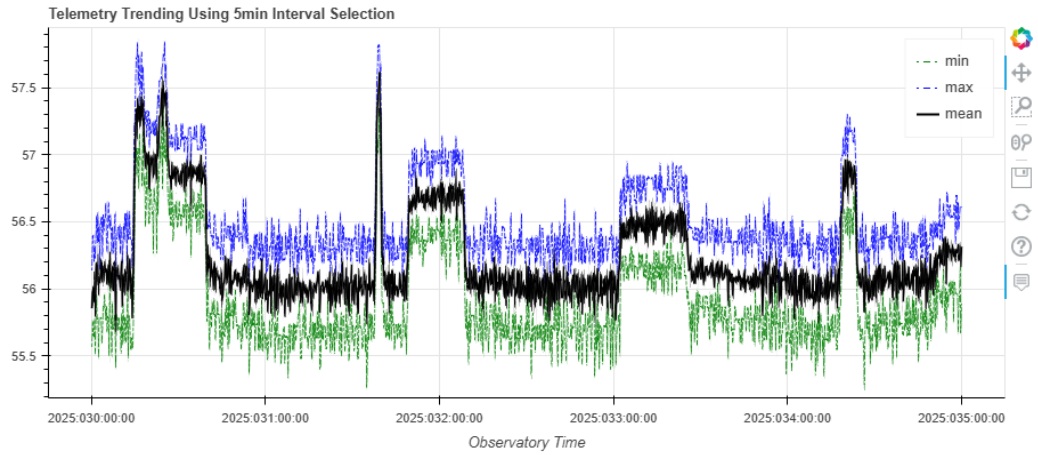

Figure 5: Long-term trend generated with the Thelma application, displaying min/mean/max statistics at 5 minute intervals

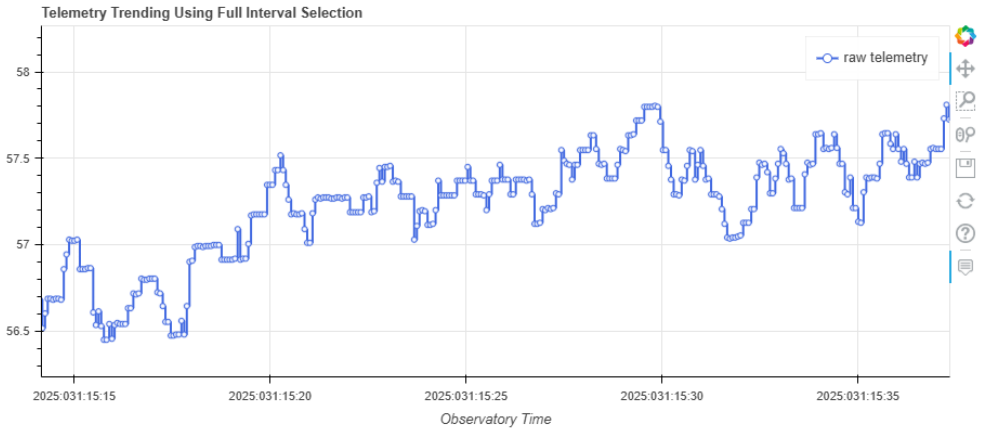

Figure 6: Short-term trend generated with the Thelma application, displaying every telemetry sample

This tool accelerates the pace of initial investigation and analysis. Quickly narrowing in on time frames and telemetry of interest informs subsequent in-depth analysis.

5.2 Routine Trending

A common task for Webb operations engineers is performing routine trending of subsystem telemetry. By reviewing the trends, subsystem engineers can identify anomalous behavior before alarm limits are exceeded.

On the Webb mission operations team, nearly all routine telemetry trending is now performed with LITA. The system allows each subsystem engineer to develop one or more Jupyter notebooks that retrieve telemetry for a particular time frame, generate telemetry plots, and save the output to a PDF. Once this notebook is developed, generating the trending report is as simple as logging into the system and executing the desired notebook to produce a report. This workflow is typical for a weekly or monthly trending report.

To reduce duplication of effort, a standard plotting template has been created. The template allows subsystem engineers to customize a trending report by editing a plain-text configuration file; no coding required. The

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

configuration file, in JSON format, defines which telemetry points to include on which plots, which plots to include together on the same page of the report, and other plot formatting options. Figure 7 is an example of a chart generated using the standard plotting template.

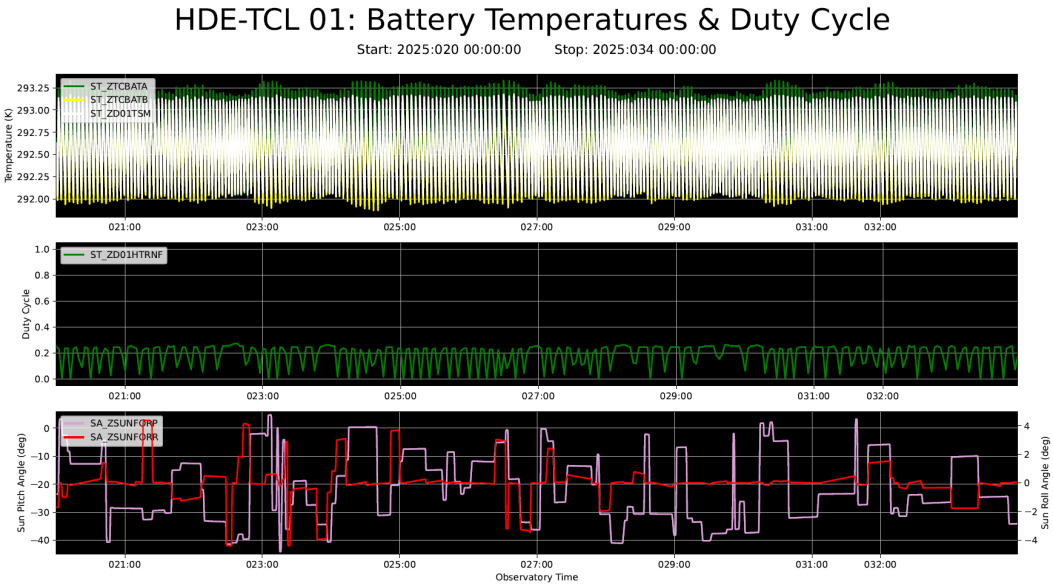

Figure 7: Battery thermal telemetry, plotted with a standard LITA plotting template

LITA supports scheduled execution of daily trending notebooks. Scheduled execution further reduces overhead, allowing engineers to quickly review the relevant status and trending data for a subsystem of interest. Reports are executed twice daily and accessed via a web interface. Any notebook can be scheduled for daily execution, permitting a wide range of data analysis and visualization in these scheduled jobs. This differentiates LITA from other scheduled trending systems available to the operations team, which only provide simple plotting capabilities. There are 23 notebooks scheduled for daily execution, representing 17 observatory subsystems. As discussed in Section 3.2.3, these auto-generated reports are available via the Thelma web interface.

As mentioned above, LITA enables trending of any data that can be calculated or derived in a Jupyter notebook. Figure 8, below, is an example where derived data is included in routine trending reports. The plot displays the duration that each CCSDS File Delivery Protocol (CFDP) downlink transaction is open. These durations are not calculated onboard or telemetered; creating this plot requires fetching multiple telemetry points, calculating the time between specific telemetry events, and binning these durations by ground station that was in use at the time. The resulting trend is generated by the automated trending task and is valuable for quickly assessing the health of the CFDP systems.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

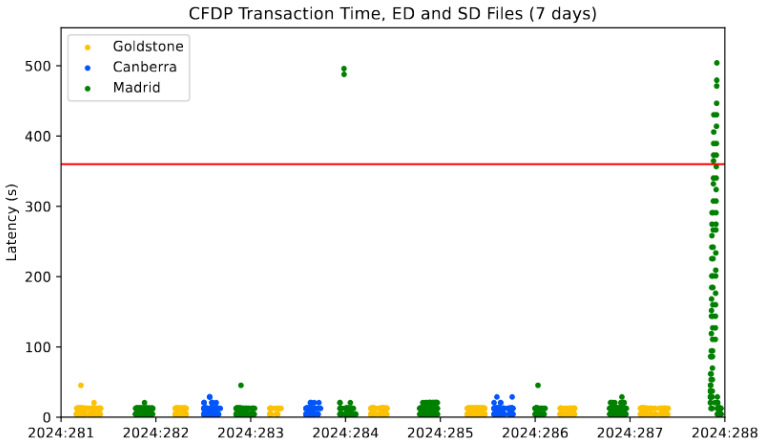

Figure 8: CFDP transaction completion time

5.3 Detailed Analysis

Beyond routine trending, LITA provides a platform for in-depth data manipulation and analysis. Integration of the data retrieval with a general-purpose python environment enables the development and execution of complex analysis tools.

The example in Figure 9 shows how the Webb Attitude Control System (ACS) operations team uses LITA to analyze the estimated Fine Guidance Sensor (FGS) alignment in the spacecraft body frame. The analysis involves fetching multiple observatory telemetry points that indicate when the FGS was active, calculating and averaging the angular offsets during these timeframes, and creating the scatter plot. The points have color assigned to indicate the observatory sun angle at the time of the observation. Now that this notebook has been developed, the analysis can be executed quickly in a single-step process whenever needed. This specific analysis has been instrumental in tracking the alignment of the FGS and determining if corrective action is required.

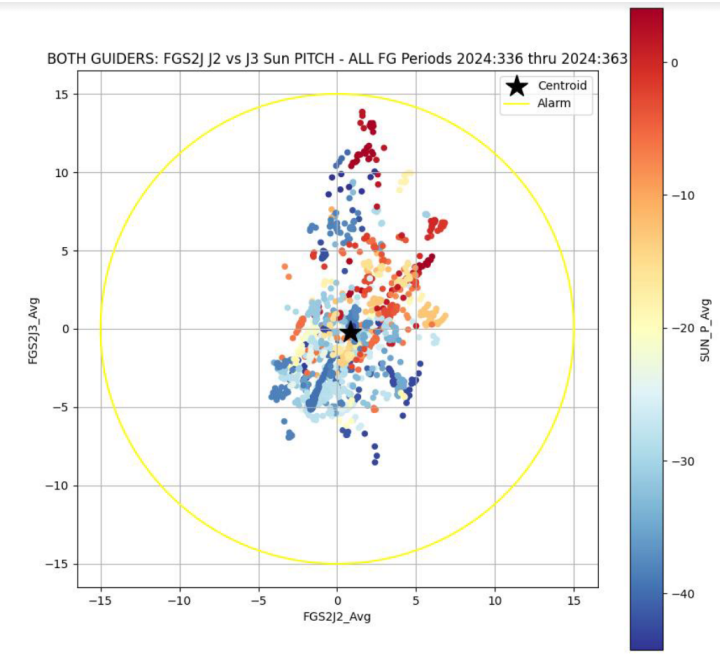

Figure 9: Estimated FGS line of sight correction

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

The integration of multiple data sources into one analysis platform streamlines the development of tools and analysis that may not be feasible otherwise. As discussed above, the LITA system provides an interface for accessing certain data from the SOC observation planning database. In addition, any resources available via the local network or the broader internet can be accessed from a user's notebook. Data from these sources can be brought together with observatory telemetry for analysis tasks.

An example analysis that uses multiple data sources is the SSR data volume prediction tool. This notebook pulls data from multiple sources:

- The ground contact schedule for Webb is accessed via an external resource. This schedule defines the upcoming downlink opportunities for stored science data.
- The scheduled science observation plan, including the expected execution time and data volume, is accessed from the SOC using LITA's interface to the observation plan database.
- The last-known onboard recorded data volume and observation plan progress are fetched from the LITA telemetry archive.

By combining these data sources, a forward-looking onboard data volume prediction can be generated. This plot, shown in Figure 10, has been a valuable tool for maintaining observatory efficiency as it allows the operations team to quickly assess the impact of lost contact time. Various contact scenarios can be tested by modifying contact times and executing the notebook to determine how the onboard data volume will be impacted.

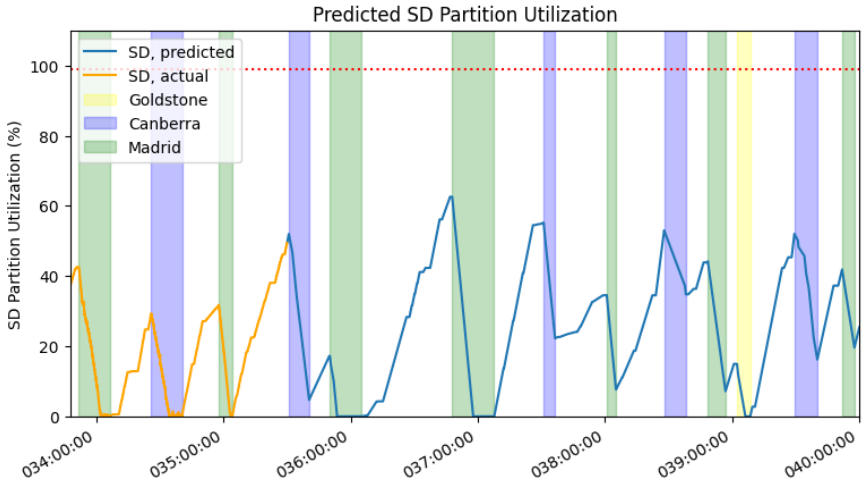

Figure 10: Predicted onboard solid state recorder data volume

5.4 Anomaly Investigation

LITA's analysis environment has been valuable for investigating the root cause of observatory anomalies. One example is the investigation into a change in solar array performance in October 2024.

Routine trending, shown in Figure 11, indicated a possible change in solar array voltage in mid-October. The voltages from the solar array circuits are expected to vary as the observatory slews to various attitudes and the incident sun angle changes. The standard voltage vs. time plot fluctuates, which can obfuscate any permanent changes in behavior.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

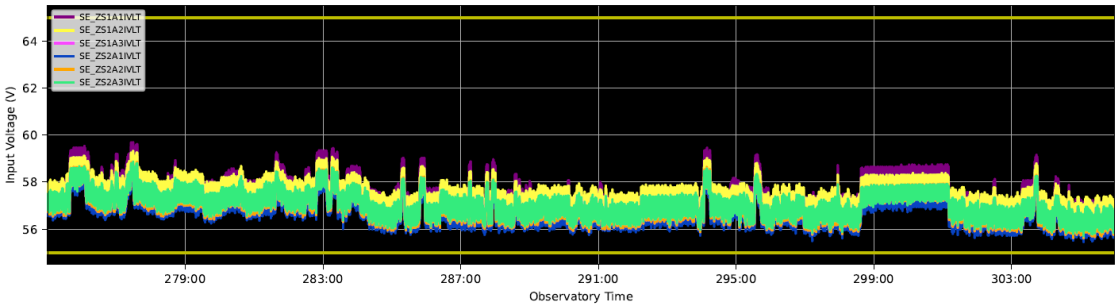

Figure 11: Monthly plot of solar array voltages over time

LITA was used to further analyze the telemetry data and determine if any significant change had occurred. This investigation leveraged built-in LITA functions and features, reducing the development time required.

The primary goal for the analysis was to plot the solar array telemetry (voltage and current) only when the observatory was at a specific attitude. This would indicate if the behavior changed while at a constant sun angle. The initial step in the analysis was to identify the time intervals when a particular attitude constraint was met. Using the built-in *logical_intervals* function (inherited from *cheta*), these time periods are quickly identified:

```
tlm = fetch.MSIDset(['SA_ZSUNFORP', 'SA_ZSUNFORR'], start=t0, stop=t1)

tlm['SA_ZSUNFORP'].vals = abs(tlm['SA_ZSUNFORP'].vals - pitch_target)
tlm['SA_ZSUNFORR'].vals = abs(tlm['SA_ZSUNFORR'].vals - roll_target)

pitch_intervals = tlm['SA_ZSUNFORP'].logical_intervals('<', 1)
roll_intervals = tlm['SA_ZSUNFORR'].logical_intervals('<', 1)
```

Next, the solar array voltage and current telemetry was fetched, and any samples taken outside the defined pitch and roll constraints are discarded. This utilizes another built-in function, *select_intervals*, which was also inherited from *cheta*.

```
tlm = fetch.MSIDset(sa_mnemonics, start=t0, stop=t1, stat=stat)

for m in sa_mnemonics:
        tlm[m].select_intervals(pitch_intervals)
        tlm[m].select_intervals(roll_intervals)
```

The remaining telemetry points were plotted, as shown in Figure 12, with additional low pass filtering applied to reduce noise. A long-term plot of solar array circuit voltage is shown below, indicating that there was a step in October 2024, with all solar array circuits impacted.

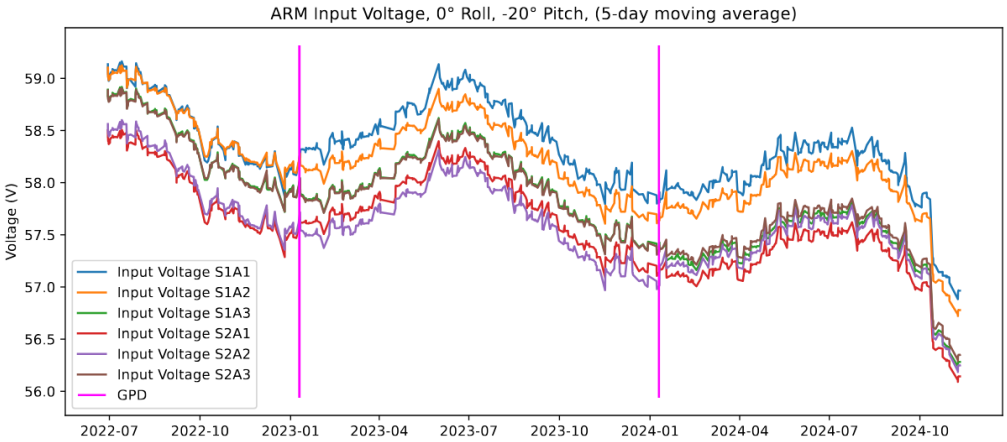

Figure 12: Long-term trend of solar array voltages, filtered by observatory sun angle

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

The analysis was re-executed for just the October 2024 timeframe. Figure 13 shows the behavior change occurred between Oct 11-14 2024. With the behavior and the time frame confirmed, the cause became apparent.

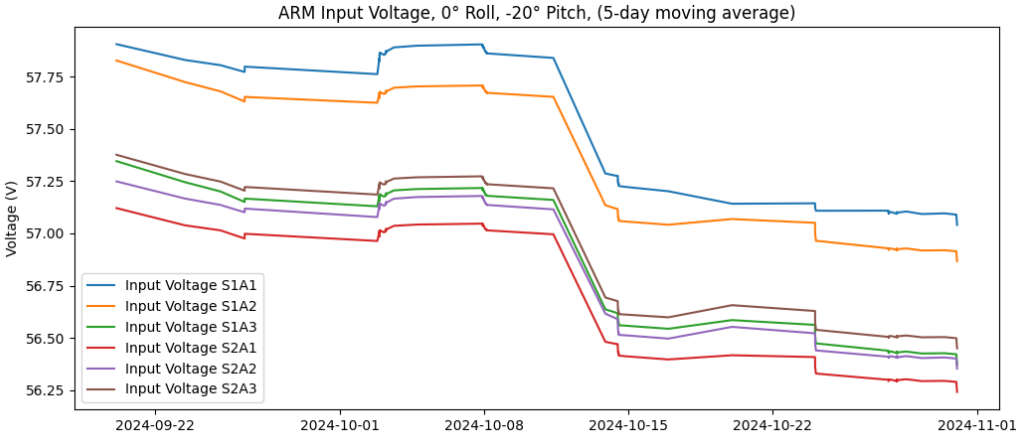

Figure 13: Short-term trend during the anomaly timeframe

A Coronal Mass Ejection (CME) on October 10-11 2024 had led to a Class G4 geomagnetic storm [1]. The Webb operations team had been monitoring for impacts from this storm, and prior to this analysis none had been identified. However, the solar array analysis performed here indicated that the storm had led to a measurable decrease in solar array output, roughly 1% decrease in voltage over baseline degradation.

While it was important to identify and understand this change in behavior, there is no long-term concern. This impact to solar array performance due to solar storms was factored into power system models, and the system has appropriate margins to account for this and future similar events. This analysis performed in LITA has been valuable to feed back into system models to continue refining lifetime predictions.

6. Conclusion

Prior to commissioning the Webb Flight Operations Team identified the potential benefits of leveraging the success of other missions' use of FOSS. Over the course of many months of iteration the LITA system was brought to life in time for Webb commissioning and its capabilities have been successfully applied to both daily operations and extraordinary events. The LITA system has had an immeasurably positive impact in ensuring the health and safety of Webb and improving the productivity of those who support the mission.

Acknowledgements

The authors would like to especially thank NASA Chandra's Dr. Tom Aldcroft for his vision and guidance in adapting *cheta* for Webb's application. His efforts epitomized NASA's goal of cross-mission collaboration and the scientific spirit of open source code.

Additional thanks is extended to the larger team supporting LITA and this paper, including Evan Adams, William Bast, Asa Bromenschenkel, Jean Connelly, Larry Doering, Erin Fontanella, Utah Gannon, Peter Gauthier, Matt Goodrich, David Hunter, Ron Jones, George McCone, Andrew Pedder, Hillary Regan, Dr. Ken Sembach, Matt Sienkiewicz, Rusty Whitman, and Alex Yermolv.

Copyright ©2025 by David Kauffman. Published by the Canadian Space Agency (CSA) on behalf of SpaceOps, with permission and released to the CSA to publish in all forms.

References

[1] NOAA, G4 (Severe) Storm Watch for 10-11 October, 11 October 2024, https://www.swpc.noaa.gov/news/g4-severe-storm-watch-10-11-october (accessed 27.02.25)
[2] Celery Organization, Celery Is …, https://docs.celeryq.dev/en/stable/getting-started/introduction.html (accessed 27.02.25).
[3] Encode OSS, Django REST framework, https://www.django-rest-framework.org/ (accessed 27.02.25)
[4] Space Telescope Science Institute, JWST Engineering Telemetry Archive (JETA), 6 July 2022, https://github.com/spacetelescope/jeta, (accessed 27.02.25)
[5] DigitalGlobe, A. Marquis Gacy, Dustin Powers, 28 May - 1 June 2018 https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2328 (accessed 27.02.25)
[6] Jupyter Team, What is a Notebook?, https://docs.jupyter.org/en/latest/#what-is-a-notebook, (accessed 27.02.25)
[7] Docker Inc., Docker Docs, 2013-2025, https://docs.docker.com (accessed 27.02.25)
[8] Space Telescope Science Institute, MAST: Barbara A. Mikulski Archive for Space Telescopes, https://mast.stsci.edu/portal/Mashup/Clients/Mast/Portal.html (accessed 27.02.25)
[9] Django Software Foundation, Why Django? https://www.djangoproject.com/start/overview/, (accessed 27.02.25)
[10] The HDF5 Group, HDF5® High-performance data management and storage suite, 2006 https://www.hdfgroup.org/solutions/hdf5/ (accessed 27.02.25)
[11] The HDF5 Group, Introduction to the Virtual Dataset - VDS, https://support.hdfgroup.org/documentation/hdf5-docs/advanced_topics/intro_VDS.html (accessed 27.02.25)
[12] Smithsonian Astrophysical Observatory, Cheta Telemetry Archive, 9 February 2025, https://github.com/sot/cheta (accessed 27.02.25)