18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 427

The JWST Observation Plan Executive

Dean Zak[a]*, Kelsie Crawford[b]

[a] *Space Telescope Science Institute, Baltimore, Maryland*, *zak@stsci.edu*
[b] *Space Telescope Science Institute, Baltimore, Maryland*, *kcrawford@stsci.edu*
* Corresponding Author

Abstract

The James Webb Space Telescope (JWST) Observation Plan Executive (OPE) is a multi-threaded onboard application that schedules and executes the science activities in the mission observation plan. Residing in the JWST Integrated Science Instrument Module (ISIM), the OPE processes observation plan visits uplinked from the JWST flight operations ground system. Each visit contains a sequence of serial and/or parallel high-level user-specified activities. Each activity calls an onboard script, which implements a high-level activity via a sequence of command executions and telemetry verifications, for execution in an event-driven manner. Based on the visit execution status, the OPE can alter the processing of a sequence of activities, and can therefore save time by skipping problem activities affected by guide star acquisition failures, target acquisition failures, lack of space on the science data recorder, hardware problems, visit input errors, or other errors. The decision-making capability of the OPE allows science operations to continue for healthy nominal activities and maximize the science productivity of the mission. The OPE is controlled by ground system scripts, which can start and stop the running of the observation plan, and add or remove visits from the plan. The OPE has been tested with simulators and flight hardware in ambient and cryogenic vacuum environments. This paper presents the design of the OPE, explores its features, and summarizes its history and development process.
Keywords: JWST, James Webb Space Telescope, OPE, Observation Plan Executive

Acronyms/Abbreviations

Activity Description (AD), autonomous momentum unload (AMU), C-Extended Command Interface Language (CECIL), Deep Space Network (DSN), Dictionary Interface (DI), end-of-plan (EOP), Flight Operations Team (FOT), flight software (FSW), Hubble Space Telescope (HST), integrated science instrument module (ISIM), integration and testing (I&T), ISIM Command and Data Handling (ICDH), James Webb Space Telescope (JWST), momentum unload (MU), observation plan (OP), Observation Plan Executive (OPE), Operations Script Subsystem (OSS), Proposal Planning Subsystem (PPS), real-time command procedure (RTCP), science instrument (SI), Science and Operations Center (S&OC), solid state recorder (SSR), Space Telescope Science Institute (STScI)

1. Introduction

The James Webb Space Telescope (JWST) is a 6.5-meter infrared telescope in orbit about Sun-Earth Lagrange point 2. Launched on December 25, 2021, JWST is expanding our knowledge of the universe with its ability to capture images and spectra of very distant astronomical objects and allows us to look further back in time toward the universe's creation. The Space Telescope Science Institute (STScI) operates the James Webb Space Telescope mission, building upon their experience operating the Hubble Space Telescope (HST), to undertake a broad range of investigations for the astronomical community.

Before the launch of the James Webb Space Telescope (JWST), an event-driven model of operations had yet to be successfully employed in a space-based astronomical observatory. Event-driven operations is the paradigm that each planned command should be issued as soon as possible after the previous command without intervention from the ground system. The need for increased science efficiency with JWST lead to the development of the Observation Plan Executive (OPE) to implement an event-driven operations model.

Since JWST is out of contact with ground control a majority of the time (roughly two-thirds of a day), the OPE must be able to the carry out the JWST science mission observations autonomously. The autonomous operations performed by the OPE include executing science activities; monitoring spacecraft momentum, disk usage, and health and safety of the science instruments; and responding to anomalous events while maintaining the health and safety of the observatory.

2. Background Information

The OPE is a subset of a set of onboard scripts that are collectively known as the Operations Script Subsystem (OSS) that live in the ISIM Command and Data Handling (ICDH) RAM filestore. The OSS scripts contain three different subsets of scripts that collectively allow JWST to operate autonomously and acquire science and engineering data: the Observation Plan Executive, the Activity Description (AD) scripts, and the Dictionary Interface (DI) scripts. While the discussion herein is pertaining to the OPE, it is still important to mention the role that the AD and DI scripts play.

The AD scripts are responsible for executing the science instrument and spacecraft operations [1]. Science instrument (SI) operations are performed by the AD scripts communicating with the SI flight software (FSW) to configure the SIs and begin science data collection. Spacecraft operations are performed by the AD scripts communicating with the spacecraft FSW for vehicle attitude changes and telescope mirror adjustments. The DI scripts are essentially wrapper scripts that translate the human readable command, telemetry, and table mnemonics used by the onboard scripts into the correct application IDs and function codes used by flight software via onboard data dictionaries [1]. Figure 1 illustrates the onboard software architecture.

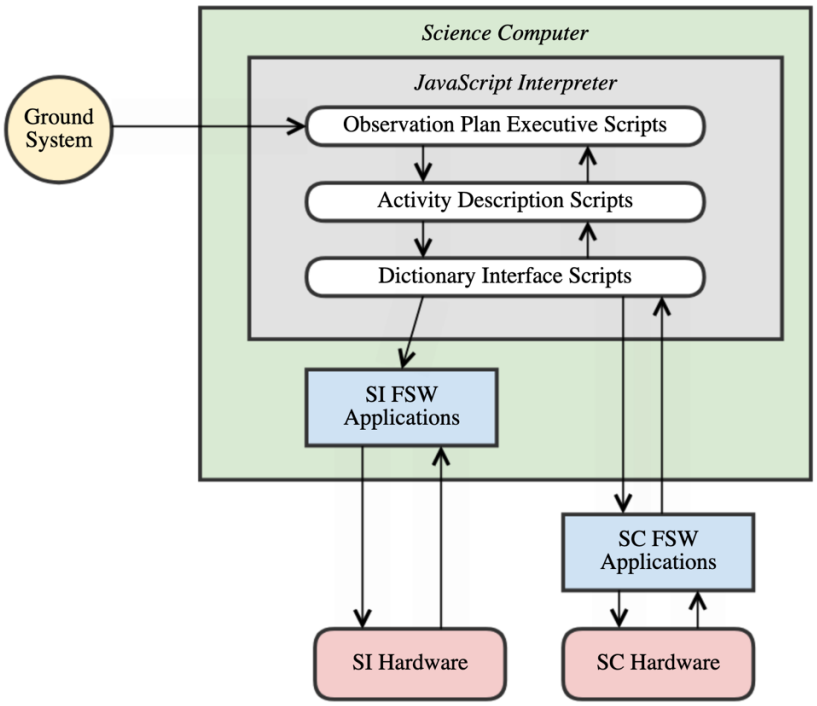

Figure 1. JWST onboard communications paths for event-driven operations.

A commercial-off-the-shelf JavaScript engine was selected as the script engine for the OSS scripts [2]. The script engine is embedded within a dedicated payload flight software application, called the script processor. The script processor can start and stop the script engine and activate and stop a script from ground commands. The script engine must be active in order to start the OPE. Stopping the script engine, whether by ground command or from the OPE reacting to an anomaly, immediately aborts any active threads of execution. The script processor provides the only means for the onboard scripts to communicate with one another and with the JWST FSW [3]. Up to ten execution threads are supported by the script processor to allow for parallel operations and real-time tasks.

To facilitate multiple threads of execution, the OSS scripts utilize shared parameters and semaphores. Shared parameters provide communication between different script threads as any of the ten threads can access them. The OPE creates and initializes the shared parameters to predefined default values each time the OPE is started. Shared parameters persist as long as the OPE is alive. Some shared parameters can be accessed by multiple threads of execution at the same time. In order to protect these shared parameters and prevent any race conditions that may otherwise arise from being accessed by multiple threads at once, OSS uses semaphores.

3. Design

3.1. Event Driven Operations

Instead of following the traditional absolute time commanding model of operations where onboard activities are split into individual time-stamped commands that include extra time padding, JWST employs an event-driven commanding model managed by the OPE. The OPE determines when to send the next command based on the command execution status of the previous command. Time is saved by not having to wait until the ground-calculated execution time, which increases execution efficiency [2]. In an absolute time commanding model, a resource being unavailable or a prerequisite event failing would result in attempting to execute the rest of the commands that would not produce useful science data. The event-driven command architecture allows for the OPE to skip any activities necessary and continue processing the OP without any downtime, thus increasing science efficiency [4]. Examples of when the OPE will skip activities include, but are not limited to the following: guide star acquisition failures, target acquisition failures, lack of space on the science data recorder, hardware problems, visit input errors, and an SI being unavailable for commanding.

Figure 2 provides a simplified view of how absolute time commanding (top panel) versus event-driven operations (bottom panel) work in the event of guide star acquisition failures. Hubble Space Telescope (HST) would remain idle until the spacecraft time matches the time associated with the start time of the next observation. However, the overlapping time windows used in JWST allow the next observation (or "visit") to start immediately after the previous failure, preventing idle time [5].

Figure 2. Absolute Time Commanding (top panel) versus Event-Driven Operations (bottom panel)

3.2. Autonomous Operations

JWST would not be able to follow an event-driven operations model unless the OPE was also designed to operate autonomously.

3.2.1. Observation Plan Package

Approved proposals from science community are assigned for scheduling on long-range and short-range plans. The Proposal Planning Subsystem (PPS) generates these plans by applying a number of constraints to ultimately determine the time windows when the science program observations can be scheduled. PPS lays out all of the observations into schedulable units called "visits", where each visit represents a new pointing and guide star selections for that set of science observations.

The onboard scripts conduct observatory operations as specified by an Observation Plan (OP) package that is generated by the ground system and then uploaded to an onboard RAM filestore in the Integrated Science Instrument Module (ISIM) payload computer. An OP package consists of a segment file, visit files, and optional auxiliary files that accompany a specific visit. Both the segment files and visit files are human readable, plain text files, whereas the auxiliary files can be either compressed (to be uncompressed onboard) or plain text. A segment file contains a list of visits with their associated start time windows and visit cutoff time, along with an end-of-plan (EOP) slew statement, as shown in the provided example (Figure 3).

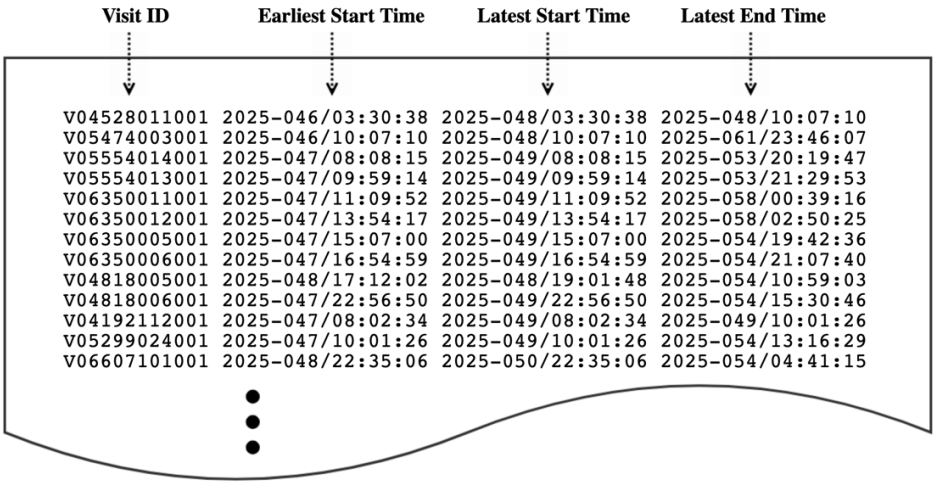

Figure 3. Segment file example

Visit files have a hierarchical structure with a serial list of one or more "activities" and their parameters that comprise a "sequence", up to three sequences within a "group" that can be executed in parallel, and one or more groups that are executed serially [6] (see Figure 4 for an example). Before the first group statement, the visit file lists any requested dithers, the predicted momentum state of the spacecraft at the edges of the visit start time window, a list of any auxiliary files (if any), and a "visit" statement. The visit statement includes the visit ID, earliest and latest start time of the visit, the visit cutoff time, a call to the visit constraint script, and the science instruments required for the visit. The group statement will have a call to the group constraint script and the SIs required for the group unless the group is a slew to a guide star. The sequence statement will have a call to the sequence constraint script and the SIs required for the sequence unless the sequence is a slew or small angle maneuver. The sequence statement will also include a special parallel visit ID if it is a parallel sequence.

Typical visits contain onboard script calls to a slew activity, a guide star acquisition activity, an optional target acquisition activity, and several science observation activities. Each activity in the visit file consists of a call to an AD script followed by a list of pass-in parameters and values.

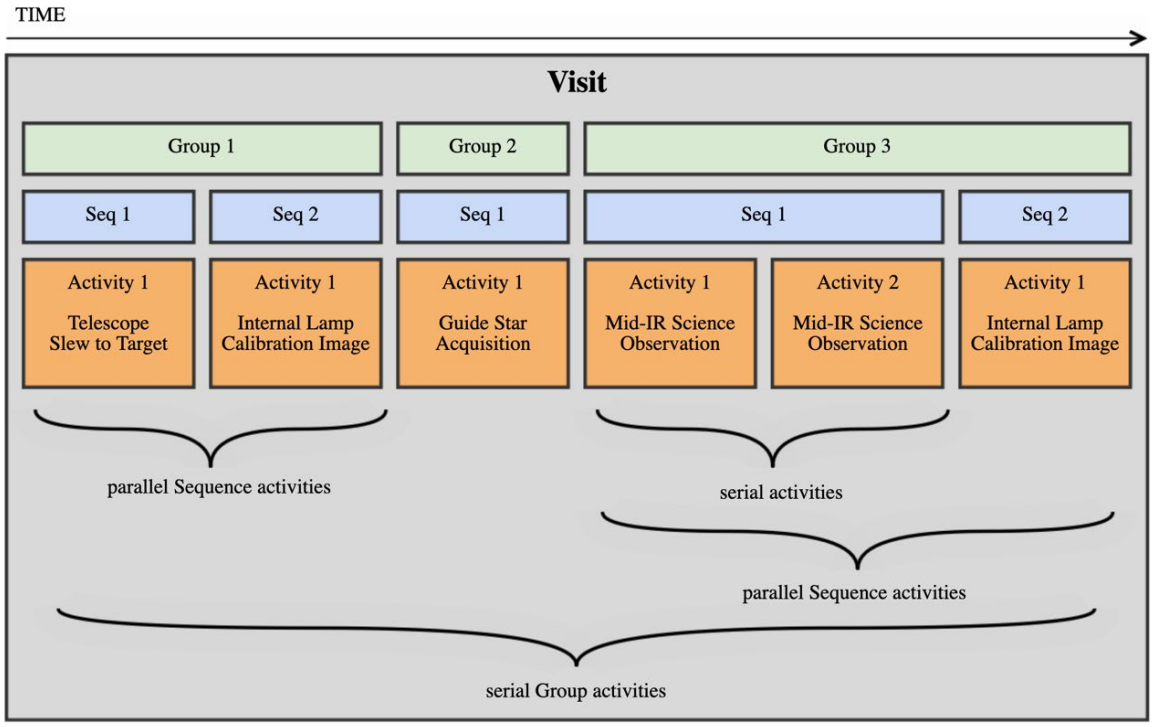

Figure 4. Example structure of a visit file

An Observation Plan segment file is uplinked to the observatory about once per week to add about ten days' worth of visits to the onboard OP, ensuring seamless operations. When a target of opportunity or discretionary time request is received by the S&OC [1], and also in cases where a time gap has opened up on the onboard plan, the currently executing OP will be "intercepted" by deleting the visits in the plan after a specified visit and then uploading a new OP package. The onboard plan is also modified after an anomaly that stopped onboard processing is resolved; however, instead of only deleting visits after a certain point in the onboard plan, the entire onboard plan is cleared before a new plan is uploaded.

When the OPE receives a request to add visits to the OP (see Section 3.3.1 for details about ground requests to the OPE), the visit files listed in the segment file are appended to the existing onboard OP, and the end-of-plan (EOP) slew statement replaces the EOP slew already stored onboard after all of the visits have been successfully added to the plan. The EOP slew is stored separately within the OPE from the OP and is only executed to maintain a legal spacecraft attitude when processing the OP stops.

3.2.2. Observation Plan Processing

When the OP is running, the OPE processes the first visit listed in the onboard OP. When the visit is finished, whether from successful execution or from an error occurring causing the visit to be skipped, the OPE removes the visit off the OP list, deletes the visit file and any associated auxiliary files, and proceeds to the next visit in the list. The OPE will continue to process the onboard plan until the end of the plan is reached. The OPE executes each visit in the list by examining the visit's earliest start time and latest start time, which comprise the start time window. The OPE will begin execution of the visit if the current time is within this start time window. If the current time is before this start time window, the OPE will wait until the time window opens to begin processing the visit; if the current time is after this start window, the OPE will skip the visit.

In addition, each visit must complete by its visit end time; otherwise, the OPE will halt processing of the visit and begin visit cleanup. Being that the amount of time needed for visit cleanup differs for each SI, the OPE calculates how early to cut off execution before the latest end time for each visit based on which SIs were used in the visit. The OPE visit cleanup tasks comprising this exit overhead are described in Section 3.2.5 below.

Once the OPE decides to skip a visit, the OPE determines how much time it will be waiting to begin the next visit, if any; performs the skipped visit's slew (or the EOP slew if the wait duration is long) to maintain a legal spacecraft attitude if needed; then will delete the visit file and remove the visit from the OP. Figure 5 and Figure 6 show two scenarios of when a slew may be executed while the OPE is waiting to begin the next visit.

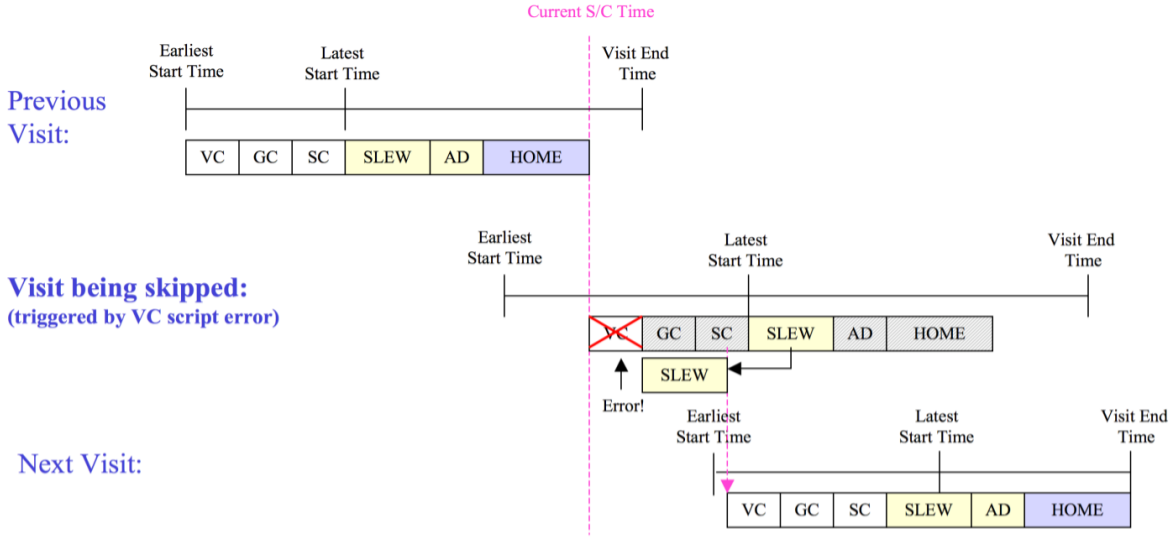

Figure 5. Slew from skipped visit executed in the time gap. VC, GC, and SC show execution of the visit, group, and sequence constraint scripts, respectively. HOME shows execution of the standard end-of-visit commanding.

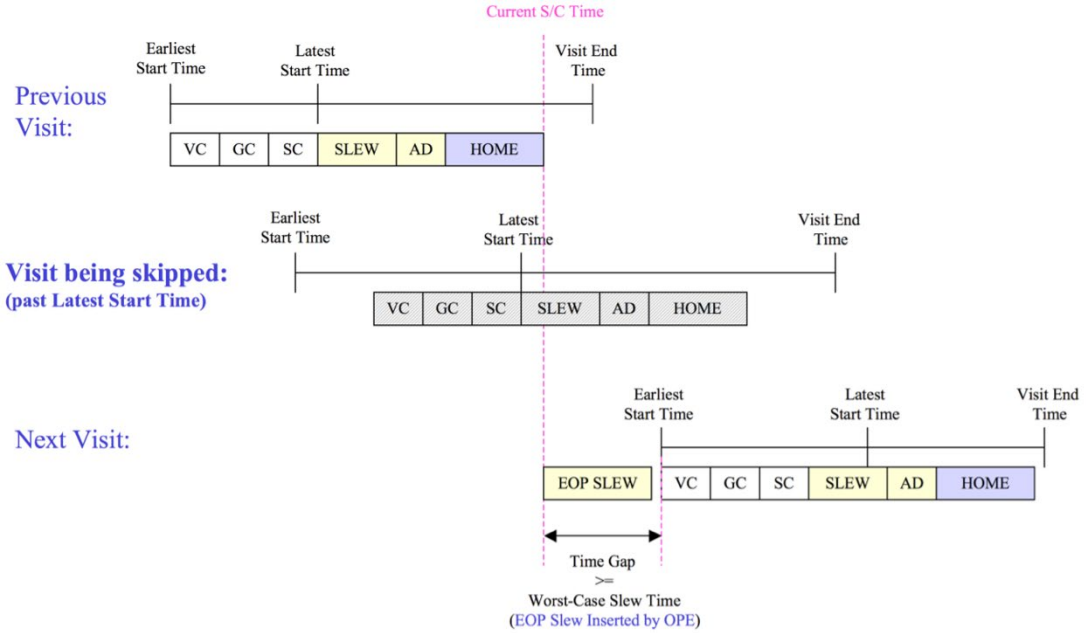

Figure 6. End-of-Plan slew executed in the time gap due to a skipped visit. VC, GC, and SC show execution of the visit, group, and sequence constraint scripts, respectively. HOME shows execution of the standard end-of-visit commanding.

When the end of the onboard plan is reached, the end-of-plan slew is performed to maintain a legal spacecraft attitude, plan processing is stopped, and the OPE remains idling until further instruction from the ground.

3.2.3. Visit Processing

Before the OPE begins executing a visit, the OPE compares the time window provided in the visit statement to the time window in the onboard plan that was provided by the segment file. If the time windows do not match, the OPE

skips the visit. Next, the OPE checks the operational constraints of the visit, group, and sequence statements (explained in detail in Section 3.4.2) in the visit file.

Once all operational constraints have been passed, the OPE executes a visit's activity statement by activating the specified AD script, passing along the parameters defined in the visit file. The AD scripts generate the appropriate series of commands and telemetry requests to execute the activity statement based on the parameters that the OPE passes through [2]. The AD scripts can also open and access any auxiliary files associated with a visit, which are files containing large data sets needed to execute the specified visit [1].

While a visit is running, execution statuses of the commands and telemetry requests are passed from the AD scripts back to the OPE, allowing the OPE to make any modifications to plan processing if necessary. The execution of each command is based upon the execution status of the previous command; as such, a command will not execute unless the previous command completes successfully [2]. This is the heart of event-driven operations. In practice, this means that whenever the AD scripts encounter an error, the AD scripts return the failure status back to the OPE in addition to generating an event message recording the problem. If the activity statement completes successfully, the OPE moves on to process the next activity statement within that sequence.

Throughout the processing of a visit, the OPE constructs time-stamped event message telemetry packets to record the visit file processing status and results. Time-stamped event messages are also constructed by the AD scripts and the science computer FSW.

3.2.4. Multi-Threaded Application

The OPE was designed to use three script execution threads: the OPE request thread, the OPE plan thread, and the OPE visit execution thread.

The OPE plan thread is the main execution thread for the OPE. The plan thread is created each time the OPE is started and remains active until the OPE is stopped. The plan thread is responsible for periodically checking for ground requests, overseeing the execution of the onboard plan, and monitoring spacecraft momentum [1]. The ground request to start the OPE invokes the main OPE script in the OPE plan thread. Once the main OPE script is running, the plan thread is able to process any other ground requests. When a ground request to manage the onboard plan is received, the OPE plan thread will spawn the OPE request thread. The request thread is only active long enough to store the ground request in a shared parameter for the OPE to access. The types of ground requests are discussed in Section 3.4. Other than a request to start running the onboard plan, the OPE plan thread is able to service all other ground requests itself once the OPE request shared parameter is updated. When the plan thread receives a request to start the OP, the OPE begins reading the first visit file in the OP list, verifies all constraint scripts (see Section 3.4.2 for details about the constraint scripts), and then launches the OPE visit execution thread if the visit is within its start time window.

The OPE plan thread checks the estimated spacecraft momentum that would accumulate in the next visit by verifying that the estimated accumulation would not cause a momentum violation [1]. If the OPE predicts that a momentum violation would occur, the plan thread launches a new thread that will perform an autonomous momentum unload (AMU), which are explained in Section 4.5.2.1. The plan thread will periodically check on the status of the momentum unload, and once the unload is complete, will move on to processing the visit.

The visit execution thread manages all the statements within a visit file. The thread remains active until a visit has finished or an anomaly is encountered. The OPE parses the statements within a visit file in the visit execution thread, and will launch the appropriate constraint and AD scripts in their own threads of execution. The visit execution thread is able to launch and monitor the status of up to five AD threads in parallel. If there is a science instrument failure during a visit, the AD thread communicates the failure back to the OPE visit execution thread, and then the visit execution thread will flag the SI so that no further activities with the SI will execute until the ground is able to resolve the issue.

3.2.5. Self-Cleaning

The OPE plan thread also periodically checks the status of a visit and performs a series a of cleanup tasks depending on how the visit ended. While each science instrument has its own set of end-of-visit activities to cleanup before the next visit, the OPE also has its own set of tasks that occur at the end of every visit, shown in Figure 7 and described below.

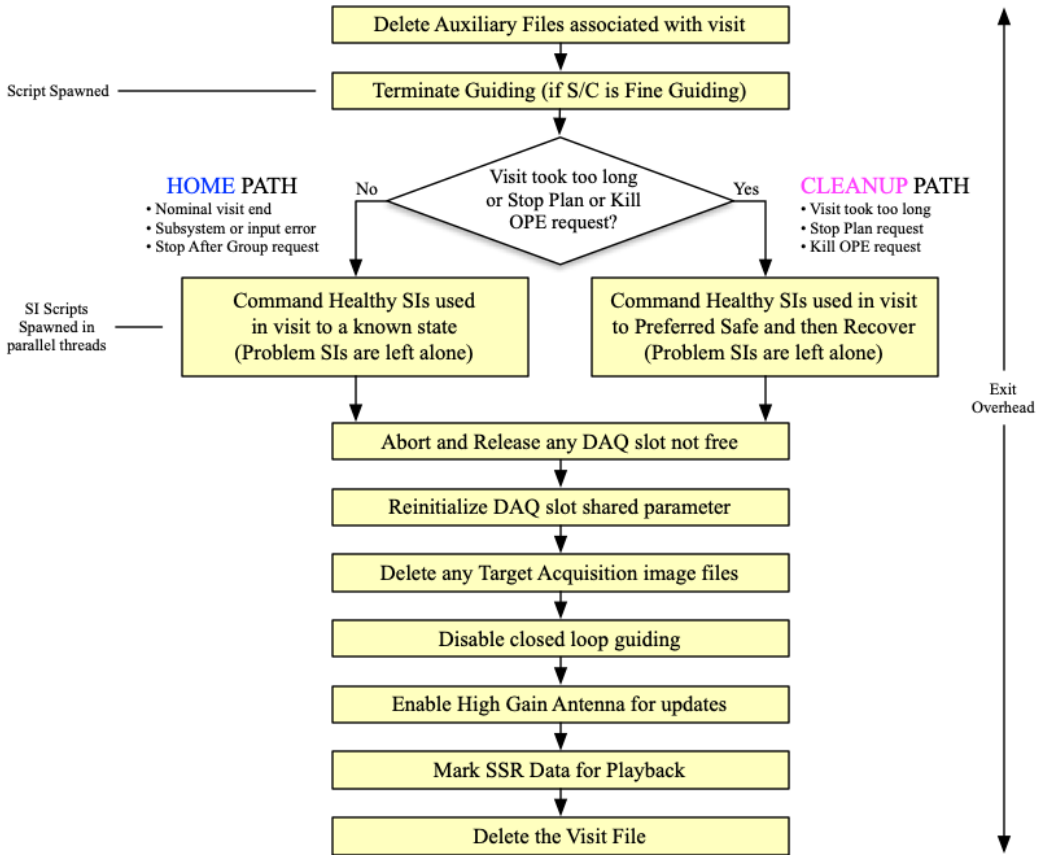

Figure 7. OPE visit end activities

While the AD scripts nominally delete any auxiliary files associated with the visit, it is possible for the auxiliary files to remain if a sequence had been skipped or ended early due to an error condition being handled. As such, the OPE begins its end-of-visit tasks by deleting any remaining auxiliary files associated with the visit.

Next, if the spacecraft was guiding during the visit, an FGS guider termination script is executed in its own AD thread. After guiding has been fully terminated is when the SI end-of-visit or SI preferred safe and recover scripts—depending on how the visit ended—will launch in parallel AD threads.

For nominal visit execution completion, completion with a subsystem or an input error, or completion due to fulfillment of an OPE stop after request (described in Section 3.3.1), an SI end-of-visit script is executed for each available SI that was used in the visit. Each of these scripts commands its SI into a known state in preparation for the next visit in the plan.

For visits needing an interruption in processing due to a visit exceeding its planned execution window, the OPE received a request to stop the OP, or upon receipt of a kill OPE request, an SI preferred safe and recover script is executed for each available SI that was used in the visit. Each of these scripts commands its SI into a preferred safe state, and then recovers the SI so that it is ready for the next visit in the plan.

After the SI end-of-visit scripts complete, the OPE aborts and releases any data acquisition (DAQ) telemetry slot that has remained allocated, and ensures that the DAQ telemetry slot shared parameter has its initialization value. The OPE then removes any leftover temporary target acquisition image files from the filestores in case the AD target acquisition scripts had not deleted them. Next, the OPE calls spacecraft scripts to disable closed-loop guiding, re-enable the High Gain Antenna for updates, and mark SSR data for playback. Finally, the visit file is deleted.

3.3. Ground Interface

As mentioned earlier, the OPE was designed to allow and process requests from the ground system. Ground requests are sent by a Real-Time Command Procedure (RTCP)—written in C-Extended Command Interface Language (CECIL)— from the Flight Operations Team, and each RTCP launches an onboard script to complete the request. Ground requests to the OPE can be broken into two categories: OPE requests and shared parameter requests.

3.3.1. OPE Requests

There are eleven different OPE requests, explained below.

Before the OPE can be initialized, the ground must first create the onboard shared parameters needed for coordination between ground scripts and onboard scripts, then initialize OSS, which will set the names of the dictionary files needed by the OPE to send commands to and receive telemetry data from the ISIM flight software. Once OSS is initialized, the ground system can request to activate the OPE, whereby the OPE plan thread is created, the OPE runs through its initialization tasks, and then begins waiting for further instruction from the ground.

As mentioned above in Section 3.2.4. Multi-Threaded Application, all other OPE requests from the ground are received by the OPE plan thread and then serviced by the OPE request thread. The other OPE requests are as follows:

- The add plan request adds the specified OP segment to the end of the onboard plan.
- The start plan request begins onboard plan processing.
- The show plan request displays the contents of the onboard plan via event messages.
- The stop after visit request gracefully stops OP processing after the specified visit.
- The stop after group request gracefully stops OP processing between two groups within a visit.
- The cancel stop plan request cancels a pending stop after request.
- The delete visits after request deletes all visits from the OP after the specified visit.
- The clear plan request removes all visits from the OP and deletes all of the visit files.
- The stop plan request terminates any currently running AD scripts, terminates the visit, and stops OP processing.
- The kill OPE request stops OP processing and kills the OPE. All currently running AD scripts will be terminated and the OPE will perform its end-of-visit tasks following the cleanup path to safe and recover the SIs prior to the OPE being killed. The OPE also deletes all of the shared parameters from existence. It should be noted that the kill OPE request is not used operationally, and instead, a script processor request to kill the script engine will be sent that will immediately terminate all threads with no cleanup.

3.3.2. Shared Parameter Requests

In addition to managing the onboard plan, the ground system may also run shared parameter requests to override or update the value of a shared parameter initialized by the OPE and used by the science instruments. Each shared parameter request is triggered by an RTCP from the ground system, and invokes an OPE onboard script that executes in its own thread. Since the scripts to update shared parameters execute very quickly, a short wait is included in each script to ensure the script execution is seen in telemetry. The ground is also able to request the OPE display any or all of the shared parameter values in event messages.

3.4. Constraints

3.4.1. Hardware Constraints

The ISIM Script Processor working buffer (also referred to as dynamic memory) is 3 MB. The working buffer holds the interpreted onboard scripts for any currently executing thread of execution, and also holds any dynamically allocated data structures. As such, the OPE, as well as the other OSS scripts, were designed to be as memory efficient as possible while operating. The OSS RAM filestore can hold a maximum of 1500 files; has a maximum capacity of 9 MB; and contains all of the OSS scripts, various SI configuration files, and all visit files. Therefore, the OSS team must be conscious to limit the number of script files needed for operations.

3.4.2. Operational Constraints

The visit, group, and sequence statements in a visit file can each specify a constraint script to execute upon the OPE's processing of that statement. All three constraint scripts verify that the ISIM, the spacecraft, and all required science instruments are available before moving on to the next statement in the visit file. Both the visit constraint script and

the group constraint script will skip the visit if any of the required subsystems are unavailable. The sequence constraint script will simply not process any of the activities in that sequence. The sequence constraint script also launches another script that verifies if there is enough available space on the SSR, via telemetry, for the sequence to execute. If the telemetry is below the parallel threshold, a flag will be set to skip parallel observations until enough space of the SSR is made available. If the prime threshold is crossed, the Observation Plan is stopped and the ground will need to restart it after enough memory on the SSR is available.

4. Features

4.1. Error Handling

One of the most important features of the OPE is its ability to respond to errors based on the severity of the error. The AD scripts provide a script completion status to the OPE. that the OPE evaluates to determine how the OPE should continue. The severity of the AD completion status is evaluated by the OPE to determine whether the OPE's response should be one of the following: "proceed", "skip the sequence", "skip the group", "skip the visit", "stop the plan", or "throw an exception".

A "proceed" response indicates the normal expected script execution (no errors). A "skip the sequence" response will cause the OPE to halt the currently executing sequence. The remaining AD scripts in the halted sequence do not get executed, but any other currently active sequences continue to execute. A "skip the group" response will halt all sequences of activities within the group. Then, the OPE moves on to the next group. As of the writing of this paper, the skip the group response is not used. A "skip the visit" response will halt execution of all remaining activities in the visit file. Then, the OPE moves on to the next visit. It should be noted that when the script completion status results in skipping a sequence, group, or visit, any currently executing AD scripts executing parallel sequences are allowed to complete. A "stop the plan" response will cause the observation plan to stop running, leaving the remaining visits in the plan. Depending on the cause of the plan stoppage, the OPE will either wait for any currently executing AD scripts of parallel sequences to complete or will terminate any currently executing AD scripts immediately. Next, the OPE will perform the end-of-visit tasks, following the safe and recover "cleanup" path of Figure 7, and command the spacecraft to slew to a legal attitude. Ground operations is then able to decide whether to restart the plan or take other actions.

In the event of a completion status indicating a fatal error, the OPE will throw an exception, which triggers a telemetry point monitor in the ISIM fault management to deactivate the script engine and transition the SIs to a preferred safe state. The deactivation of the script engine results in the forced termination of all OSS script threads (OPE threads and AD threads).

4.2. Operational Availability Status of Subsystem

During visit execution, if an AD script completion status reports a SI error code, the OPE may handle the SI error by marking the SI unavailable. When an SI is marked as unavailable, the OPE will not command that SI, therefore skipping over visits and sequences using this SI. The OPE keeps track of a subsystem's availability status in a special shared parameter that is echoed in an ISIM telemetry point for each subsystem. The status of the subsystem conveyed in this special shared parameter does not convey the health and safety of said subsystem, and is only an internal parameter for the OPE to know if it can command the subsystem. As with all shared parameters, the subsystem status shared parameter can be modified by a ground request to the OPE. However, the ground should only change the status of a subsystem from unavailable to available after the error that made the OPE declare the subsystem unavailable has been investigated and mitigated.

4.3. Reporting and Tracking Plan and Visit Processing Status

During visit execution, there are several event messages that are sent from the OPE to the ground and are captured in the ground system's real-time event log. Event messages are sent out by the OPE marking the start and end of a visit, as well as the start and end of each group, sequence, and activity statement within the visit. These event messages bracket each constraint and AD script launched by the OPE, and allow ground operators and analysts to trace the currently executing activity back to the visit file.

Embedded within the activity event messages is a "request ID", the name of the script, and, for activity end messages, the completion status code being passed back to the OPE. The request ID is formed by a concatenation of the visit ID,

parallel ID, group ID, sequence ID, and activity ID. All of the information contained in the activity event messages can be used to correlate an activity in the real-time event log to the visit file activity statement.

In addition to event messages sent by the OPE, the ICDH Script Processor monitors and downlinks telemetry tracking the request ID, the executing script name, and the script thread status (active or inactive) for each script processor thread. This telemetry is available in the ground system's telemetry viewer.

4.4. Real-Time Visit Processing

During Deep Space Network (DSN) contacts with the ground, ground operators often need to send commands to JWST to perform maintenance activities. Instead of interrupting the execution of the onboard plan, a "real-time visit" can be scheduled by PPS whose visit time window matches the time window of the DSN contact. When a real-time visit begins, a special real-time visit AD script is called and waits for the ground to "take control" during a handoff window by executing an OSS RTCP script that tells the OPE that the ground is now in control. If control by the ground is not taken before the handoff window ends, OSS will skip the rest of this visit. If control by the ground is taken, the OPE will wait until the ground releases control via another OSS RTCP script before completing the visit. While the ground is in control, the ground can issue commands without interference from the OSS scripts. Should the command window duration expire before the ground releases control, the OPE will stop the plan since it may not be okay to proceed with the plan. If the ground chooses to skip the commanding window, the ground can set control to OSS during the handoff window. If all is well and the real-time visit is done, the OPE proceeds to process the next visit in the plan seamlessly. Examples of real-time visits include station keeping (i.e., orbital maintenance), software updates, and off-nominal operations (i.e., anomaly recovery).

4.5. Processing Spacecraft Activities

4.5.1. Solid State Recorder Free-Space Monitoring

During every DSN contact, barring any DSN issues, the ground system will downlink as much data as possible from the SSR science partition to free up space for more data to be acquired. If the ground system were to miss too many DSN contacts with the observatory, it would be possible to run out of free space on the SSR science partition. As such, one important feature of the OPE is to verify that the SSR science partition has enough free space available to store the data expected to be produced during a visit.

In addition to the sequence level constraint script launching a spacecraft script to ensure there is enough room on the SSR for the sequence to execute, the OPE also monitors the amount of SSR free space both while running the onboard plan and while idling. While the OP is running, the visit execution thread is responsible for polling the SSR science partition free space periodically. If the free space drops below a "disable parallels" threshold, a shared parameter is set to allow only prime sequences to run, but any currently executing parallel activities are allowed to complete. If the free space later rises above a "re-enable parallels" threshold, the shared parameter is reset to allow future parallel sequences to run. If the free space drops below a "disable primes" threshold, the OP is stopped and another shared parameter is set to prevent the OP from restarting until sufficient SSR space becomes available. If the free space then rises above a "re-enable primes" threshold, the OPE issues an event message indicating that sufficient space is now available to start the plan; however, the ground system is responsible for sending the start plan request to the OPE.

While the OPE is idling (i.e., a startup, during any OP stoppage, or the OP ran out of visits), the OPE plan thread polls the SSR science partition free space and will reset the appropriate shared parameters if the free space increases above or decreases below appropriate thresholds. The OPE will also produce an event message any time the SSR free space crosses any of the thresholds mentioned.

4.5.2. Momentum Management

During a science observation, JWST needs to maintain its pointing vector without allowing orbital momentum and solar pressure to affect the spacecraft's attitude. A set of six reaction wheels absorb the excess momentum during an observation. When the wheels reach their maximum momentum levels, their momentum must be unloaded. Unloading spacecraft momentum from the reaction wheels involves firing thrusters and expending limited fuel resources. Visits are planned in a manner such that slews and offsets will balance out the momentum build up by spinning the reaction wheels up and down as equally as possible. However, significant deviations from the modeled momentum profile can occur due to the event-driven nature of observation plan execution.

There are four distinct categories of momentum unload operations for the JWST observatory: planned, autonomous, contingency, and real-time. Planned and autonomous unloads are both managed by the OPE. Contingency unloads are executed by the spacecraft FSW when a fault limit is violated and result in the cessation of science operations. Real-time unloads may be commanded manually by flight operations staff on the ground. There are no scenarios in which spacecraft fault management is prevented from performing a contingency momentum unload necessary from a health and safety perspective.

Planned momentum unloads (MUs) are special visits scheduled in the OP. The OPE performs planned momentum unloads to either zero-out the magnitude of the momentum vector or bias the momentum vector to set up for one or more long observations that would otherwise lead to a limit violation. A planned momentum unload is also typically performed in preparation for a station keeping visit, used to maintain or refine the observatory's halo orbit about Sun-Earth Lagrange point 2.

4.5.2.1. Autonomous Momentum Unloads

By routinely monitoring the magnitude of the momentum vector reported by the spacecraft and evaluating the impact of each visit in the onboard plan, the OPE can determine when an unscheduled autonomous momentum unload (AMU) must be executed. AMUs help keep the observatory from entering into a safe mode that would halt science operations until flight operations personnel could evaluate and respond to the anomaly, preventing a significant impact to observing efficiency. The OPE will only launch an AMU if more than a pre-defined amount of time has elapsed since the last AMU was executed as a safeguard to prevent fuel waste. The OPE handles four distinct AMU scenarios, described below.

The first AMU scenario occurs while the observation plan is stopped and the OPE is idle. In this state, the OPE checks the momentum vector magnitude in spacecraft telemetry at a rate of ~1Hz and compares the retrieved value against a pre-defined "instantaneous" limit. If this limit is exceeded, an AMU activity is launched by the OPE.

In the other three AMU scenarios, the OPE is executing visits within an observation plan.

In the second AMU scenario, there is no time gap between a visit that has just ended and the next scheduled visit on the OP. The earliest start time of the next visit has already elapsed but the current time is still within the valid execution time of that visit. The OPE checks the current momentum magnitude against a pre-defined "instantaneous" limit, and then makes a second check to determine if the current momentum magnitude plus the momentum build-up expected during the next visit would violate a pre-defined "predictive" limit. If either of the two checks fails, the OPE launches an AMU activity before proceeding with the next scheduled visit, depicted in Figure 8. After the AMU completes, the OPE verifies the scheduled visit is still eligible to be launched based on its latest start time. If its latest start time remains in the future, the visit is launched. If the latest start time is now in the past, the OPE moves on to a subsequent visit on the observation plan.

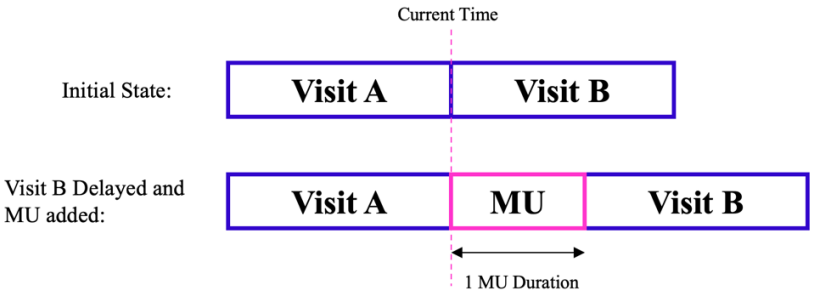

Figure 8. No time gap between visits

The third AMU scenario can occur if there is a time gap between a visit that has just ended and the next scheduled visit on the OP, but the gap is less than the pre-defined duration of an MU. Similar to the second scenario, checks are made against the instantaneous and predictive momentum limits. The main difference is that the computation of predictive momentum build-up adds a worst-case momentum accumulation that could occur during the time gap. If either limit is exceeded, then the OPE launches the AMU activity, delaying the start of the next visit (see Figure 9).

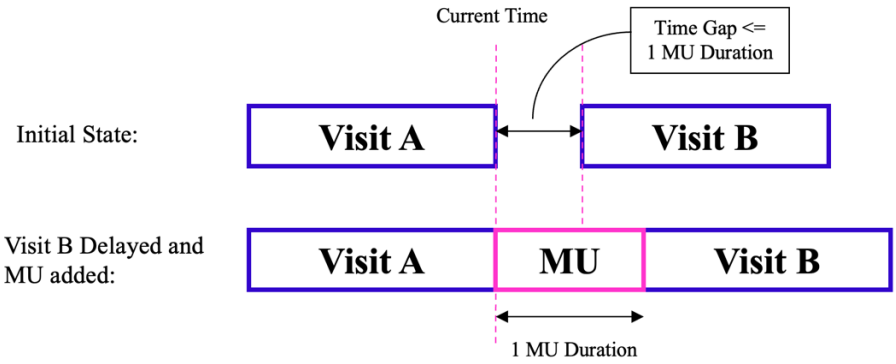

Figure 9. Time gap ≤ 1 MU duration

In the final AMU scenario, the is a time gap between a visit that has just ended and the next scheduled visit on the OP is greater than the expected duration of an AMU. While the OPE waits for the earliest start time of the next visit to arrive, it polls the current momentum magnitude. If the "instantaneous" limit is exceeded and the remaining gap time exceeds the expected duration of an AMU, an AMU activity is launched and completes prior to the earliest start time of the next visit (see Figure 10). If the limit is exceeded and the gap time is now less than the expected AMU duration, the third scenario described above now applies.

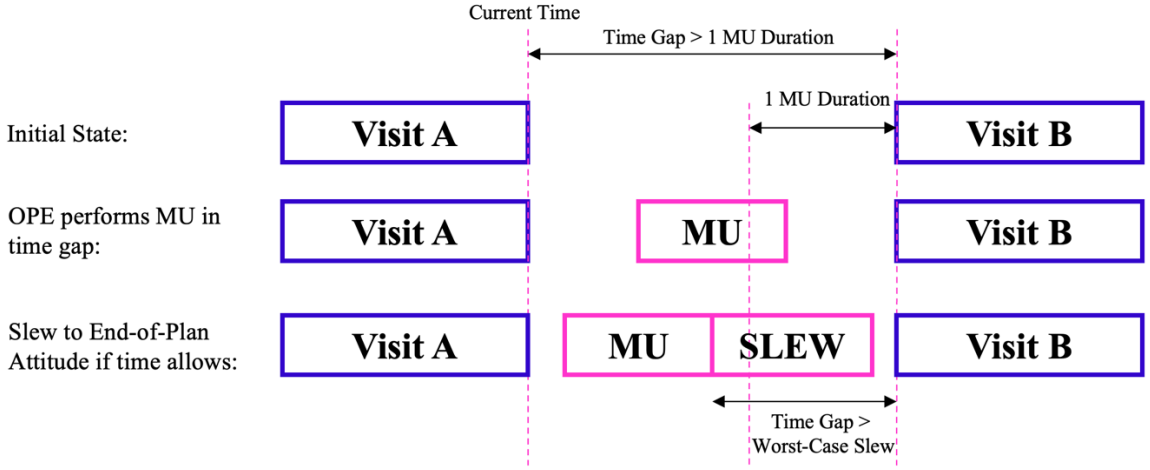

Figure 10. Time gap > 1 MU duration

Since launch, there has been a single AMU executed by the OPE. It occurred during normal operations in June 2023 during a 5 hour 35 minute time gap between visits. The instantaneous limit was not tripped by the OPE momentum management until it switched from scenario 4 to scenario 3 as described above.

5. Testing and Flight History

OPE onboard script development began in 2006 along with the SI onboard script implementation by the OSS Team. All OSS scripts went through a vigorous verification and certification process pre-launch. In normal operations, nominal updates to OSS scripts are grouped into a build, and each build is certified for flight operations.

5.1. Testing and Certification

There are two levels of certification of the onboard scripts within the purview of the OSS development and certification teams: Level 1 testing and Level 2 testing. Level 1 testing verifies that all legal pass-in parameter inputs to the AD script entry points are valid, and ensures that all paths of code are tested, including commanding pertaining to fault management. Level 1 testing is performed on a JavaScript simulator that receives commands and responds back with telemetry set to the correct values; as such, Level 1 testing does not rely on high-fidelity simulators of other flight software systems.

Level 2 testing, on the other hand, occurs in a flight-like environment with interfacing to other flight software system simulators. This level of testing focuses on success-oriented regression testing, targeted testing (for both success and failure cases) pertaining to the code updates associated with the build under test, and day-in-the-life testing whereby a typical Observation Plan is generated by the front-end system and executed. All paths of code are unable to be tested at Level 2 due to insufficient resources necessary to efficiently support thousands of failure scenarios.

Level 2 testing of OPE script updates first consists of executing the OPE request RTCPs to certify the ground to OPE interface. Second, OPs are executed to demonstrate proper onboard plan execution and the many OPE features that can only be tested while the onboard plan is running. Upon successful execution of all OPs, test artifacts are presented to internal and external stakeholders for approval. After approval from all stakeholders, the OPE script updates are certified for execution on flight hardware.

5.2. Pre-flight Mission Tests

Since event-driven operations had never been successfully implemented into a space-based observatory prior to JWST, it was vital for the OPE to finish development early enough that it could be used in numerous mission-level tests to thoroughly prove the design and concept.

During all four cryo-vacuum test campaigns from 2013–2017, the OPE supported several integration and testing (I&T) tests of the science instruments as well as OSS day-in-the-life tests. The OPE also went through numerous tests between 2018 and launch including ground segment tests, mission operations observatory tests, SI rehearsals, and normal operations tests. In all of these test campaigns, the OPE performed very well, providing verification of the OPE features and requirements, and requiring only minor updates.

5.3. In-Flight History

JWST went through a commissioning period that lasted from launch through early July 2022. Normal operations of JWST began shortly after in July 2022 and is still ongoing. From July 1, 2022 through Jan 31, 2025, the OPE has executed 10,220 visits.

Over the course of normal flight operations, the Flight Operations Team and the OSS team found a handful of ways to further mitigate risk of potential anomaly scenarios. These OPE updates were implemented during regular development and release build cycles.

6. Conclusions

The OPE was designed to operate fully autonomously while still providing the ground system the ability to control the observatory when necessary, serving as the interface between the ground system and the science instruments. The OPE implements the event-driven operations methodology by actively monitoring the status of the observatory and ongoing observations, and alters the observation plan to adapt to any events that may occur. The demonstrable success of the OPE makes it clear that future space-based astronomical observatories would benefit greatly from implementing an event-driven operations architecture following the paradigm of the OPE.

Acknowledgements

We would like to acknowledge the help of a few people, without whom this work would not have been possible. Alan Welty for his expertise in OPE operations and his ongoing efforts in mentoring Kelsie Crawford about the system. Kyle Elliot for his OSS expertise and illuminating discussions on this paper's early drafts. Jeff Stys for his expertise and his continual support of Kelsie Crawford. Rachel Vander Vliet for editing assistance.

References

[1] V. Balzano, D. Zak, and W. Whitman, "The care and feeding of the JWST on-board event-driven system", in *Observatory Operations: Strategies, Processes, and Systems III*, 2010, vol. 7737, Art. no. 77370I.
[2] V. Balzano and J. Isaacs, "Event-driven James Webb Space Telescope Operations", SpaceOps Conference Proceedings, AIAA-2006-5747, 2006.
[3] S. Barrow and V. Balzano, "Commanding the James Webb Space Telescope using event-driven operations software," 2012 IEEE Aerospace Conference, Big Sky, MT, USA, 2012, pp. 1-7.
[4] V. Balzano, D. Zak, "Event-Driven James Webb Space Telescope Operations using on-board JavaScripts", Proc. SPIE 6274, Advanced Software and Control for Astronomy, (2006).

[5] S. Zonak, Event-Driven Operations, OSS and JWST, STScI Newsletter, 2020, Volume 37, Issue 02, https://www.stsci.edu/contents/newsletters/2020-volume-37-issue-02/event-driven-operations-oss-and-jwst, (accessed 2025-02-21)
[6] K. Elliott, "Commanding the James Webb Space Telescope: OSS and Event-Driven Operations," in IEEE Women in Engineering Magazine, vol. 17, no. 1, pp. 22-27, June 2023.