Improving Sequence Traceability During Testing and Review for the Mars Science Laboratory

Jonathan Denison
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
jonathan.denison@jpl.nasa.gov

Mark Maimone
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
mark.w.maimone@jpl.nasa.gov

Abstract—**While operating spacecraft, many teams break operations planning into two processes. One process focuses on planning for the upcoming cycle, while another process governs the development of unique activities that require careful study, testing, and review before they are ready for use in flight. While developing unique activities that require extra testing and review, teams often make use of testbeds on earth to ensure the activity is well planned and that sequences of commands are compatible with hardware and software configurations. Despite careful tracking, it can be easy for mistakes to creep in before sequence delivery and uplink while teams are testing and iterating over difference sequence versions. For example, a sequence that required an update for success in the testbed could have an earlier version inadvertently submitted to the flight sequence database.**

The Engineering Operations team for NASA's Mars Science Laboratory Curiosity Rover had a desire to improve the traceability of these sequences as they are tested, reviewed, and uplinked. This paper details the design, development, and implementation of a new tool that uses the Sequence Checksum to make it easier for operators to prove three key things during the review process. First, that the sequences evaluated in the testbed match what has been delivered to the sequence database for use in flight. Second, that all sequences being delivered as part of an activity were indeed tested. Third, that all sequences necessary for activity completion have been delivered to the sequence database. While this may be trivial to accomplish for simple activities, keeping track of all this information can be difficult for activities that include tens of sequences. This paper will discuss the design and usefulness of the tool, and also the overall activity review and approval process.

TABLE OF CONTENTS

1. INTRODUCTION

In the more than 12 years and 4300 Martian Solar Days (sols) that the Mars Science Laboratory (MSL) project has been using the Curiosity Rover to explore Gale Crater on Mars, a dizzying array of unique Engineering and Science activities have completed successfully. The rover is commanded using what are called Flight Software (FSW) Commands. FSW commands allow the operators to command basic activities, change parameterized values, and adjust autonomous behaviors by changing system modes. Most commands are event-driven, meaning subsequent commands will not start until the preceding command has fully completed. As the one-way light time between Earth and Mars can vary from 3 to 22 minutes, the MSL team operates the spacecraft by sending a set of activities that span an entire sol or several sols at a given time [1]. When several simple FSW commands are combined to elicit a more complex and higher-level activity, this is called a sequence. A sequence is a file generated by operators that can contain hundreds of FSW Commands.

While FSW Commands are the basic unit to measure a specific action by the rover, a sequence is often a basic measure by which a more complex activity is defined. A typical sequence delivered might contain all the commands to retransmit data products that weren't properly received, or all the images to be taken with the Navigation Camera after a drive. Sequences are often designed to be re-usable to save uplink bandwidth. In our two sequence examples, a new retransmit sequence would need to be generated each planning day as its contents must change from plan to plan. The post-drive imaging sequence, however, can be crafted so it can be left onboard the spacecraft and called by a higher-level sequence in such a way as to remove the need for it to be re-written each planning cycle. A typical planning session for 1-3 sols on Mars will take 6-8 hours for the uplink team to complete. On average, a given plan utilizes 98 sequences, 36 of which are unique to that plan. To date, more than 76,000 unique sequences have been executed onboard the flight rover.

To tell sequences apart and make it simpler for the operations team to manage them, each sequence is assigned a name when it is created. Figure 1 displays the components of a sequence identifier (sequence ID). Each sequence ID starts with one of several four-character sequence categories that give an indication of what the sequence is for or what team is responsible for it. That sequence category is followed by a numeric identifier that can range from 00000 to 16383. The sequence category and the numerical identifier together are referred to as a sequence ID. If a sequence needs to be updated for some reason but its purpose is unchanged, the team updates a separate sequence *version* but keeps the sequence ID the same. To make it clear what version of the sequence is being discussed, each sequence ID is accompanied by a flight and ground version: the flight version is a number ranging from 0000 to 4095 and the ground version is a letter from a to z.

979-8-3503-5597-0/25/$31.00 ©2025 IEEE

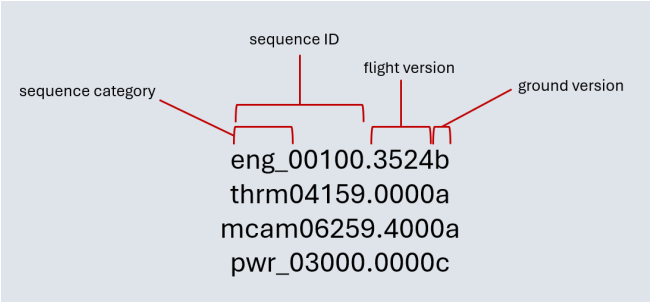


Figure 1. Examples of sequence ID and versions notation

MSL makes use of a sequence database (SeqDB) to provide version control for sequences and allow for seamless contribution of sequences from several teams to one uplink planning session. A sequence will be written locally by a user and will not enter strict configuration control until it is "delivered" to SeqDB. Once a specific version of a sequence is delivered to SeqDB, it cannot have its contents changed and re-delivered unless it is given a new and not previously used sequence version. In instances where you need to change the contents of a sequence and the sequence has never been incorporated into the uplink process, the convention is to increment the ground version and leave the flight version unchanged. In instances where that criterion is not met, such as when an update to a sequence is being made and that sequence has already executed in flight, the convention is to increment the flight version to one that has not been previously used and reset the ground version to 'a'.

A sequence can change significantly before it is ready to be used in flight or finalized enough to be delivered to SeqDB and enter version control. This can happen when several team members pass files back and forth, when an issue is encountered in the testbed and someone finds a mistake or makes improvements, or when a change is needed to account for unique sequencing that is only valid in the testbed. As the activity being planned becomes more complex, this can begin to overwhelm even the most careful team as they collaborate on sequence development and testing. For context, a typical arm activity on the flight rover that included drilling a new target over the course of 1 hour and 45 minutes included 76 unique sequences (some of which are called multiple times). In total, over 7500 FSW commands were dispatched from sequences during the drill attempt on sol 3512.

With all this sequence development and testing complexity, our main problem begins to come into focus. There are several ways that a mistake can be made in the sequencing that are very difficult to catch. How can the team be sure that once we are satisfied with how the activity was run in the testbed that same confidence will translate to the sequences we deliver for flight? The sequence iteration and testing happen before a sequence is ready for delivery, so we cannot rely on the configuration control of SeqDB. Delivering the sequences to SeqDB before testing would not help either as SeqDB would become flooded with poorly written and untested sequences that may still have changes made in the testbed that were not brought back into the already delivered versions. Much of the review process that an activity is subjected to is designed to help catch potential issues, but that can be hard for a human to notice when tens of sequences are changed countless times as the activity is prepared for testing.

This paper introduces a new tool developed for use by the MSL Engineering Operations team that makes it easier to confirm that the sequences intended for use in flight were properly tested and had the desired effect. There is a concept included in FSW already that provides the key link that makes all these sequences much easier to review. FSW makes use of something called a Sequence Checksum that can be used by operators to further discern one sequence's contents from another. Much like a checksum used to verify the validity of file contents, a Sequence Checksum can provide a concise way to verify a given sequence's commands are unchanged between uses. Before detailing the design of the tool and illustrating how it makes use of the Sequence Checksum, we must first take a closer look the development and review process for new sequences, the pre-existing version controls in place for sequences, and how sequences are compiled and sent to the spacecraft.

2. Strategic Activity Background

While the Curiosity rover can be commanded to complete a wide range of science and engineering actions, these activities can be broadly categorized into two types. The first type of activity is known as a tactically planned activity. These activities are typically lower complexity, carry less risk, and the tools available to the operations team can be fully trusted to check and simulate these actions. The operations team can sequence and check these activities for correctness during a typical 8-hour planning shift. The second type of activity is more complex and will take much longer for the operations team to test, plan, and review the design of the activity and the sequences themselves. These are called 'strategic activities' and will be the focus of this paper as it is this complexity that can create sequence traceability issues.

The Different Paths to Approval

The definition of strategic activities used by MSL is applied broadly and can include activities that may not be uplinked to the flight rover. For example, while the sequencing related to conducting a Flight Software update [6] or diagnosing an issue and creating a new drilling method [2] certainly come to mind as a strategically planned flight activity, the term strategic activity also includes deploying an update to a ground software tool, redesigning a ground process, or conducting exploratory testing in a testbed to inform an anomaly investigation. There are various paths that strategic activities can take to approval, but the ones that involve developing sequences for use in flight require what is called an ActID (short for activity identifier) to generate accompanying documentation that is reviewed and approved before use.

MSL makes use of a custom web tool called MSLReports which acts a repository for several planning products and operational analyses. It also has an area for strategic activity tracking. When a new activity documentation template is created on MSLReports, it is assigned a numerical ID. This is where the term ActID originates. The numerical ID is used to differentiate between ActID's. For example, ActID-1031 contains all the relevant documentation for the uplink and installation of a new Flight Software release called R13 which was conducted in 2023 [6]. For each ActID, an Activity Lead is identified who will manage its development, testing, and approval process as well as serve as the main point of contact for others on the team who have questions.

Activity Development, Testing, and Review

Once the need for a strategic activity is identified, the typical Engineering Operations team process for approval includes the eight steps shown in Figure 2, some of which is done in parallel and much of which can require iteration. First, there needs to be a discussion amongst several members of the team to decide on the basic structure of the activity and what type of test venue will be required to prove the activities' effectiveness. MSL has access to several testbeds that are designed to accommodate different types of testing, and choosing a venue which allows the team to test in a realistic manner is an important decision. Second, the design of the activity and the ActID documentation needs to be developed. ActID documentation includes an activity overview, a listing of spacecraft sequences and files, modeling and simulation results, uplink product testing results, and pertinent information for the teams planning and subsequently assessing the activity. Third, the sequences themselves and a procedure for testing them in a testbed need to be written and potentially reviewed. Fourth, the testing is performed. Fifth, the test data is analyzed by the Activity Lead and annotated to help prove the activities' effectiveness. Sixth, all the documentation in the ActID is finalized in preparation for final reviews. Seventh, the finalized ActID and the associated test data is reviewed internally within the engineering team. Finally, any necessary final approvals are obtained which can vary widely depending on the nature of the activity. High complexity activities often require additional approvals from the Project Manager and Mission Manager at reviews that are larger than the internal engineering review.

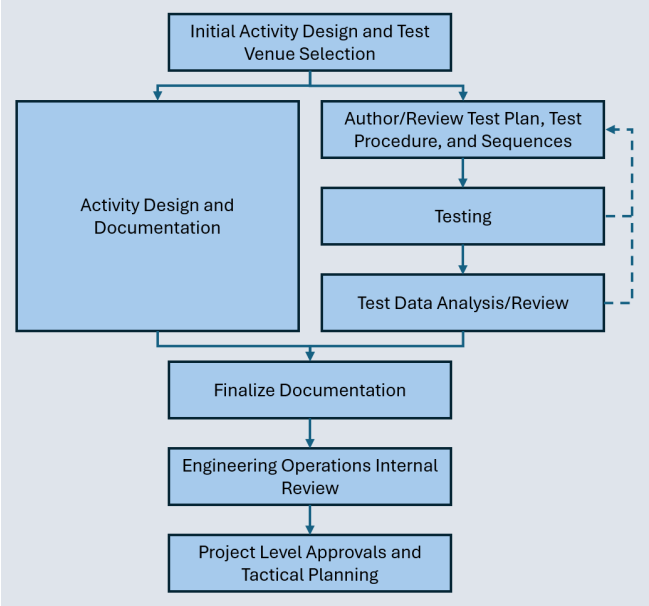


Figure 2. Simplified Diagram for the ActID Development, Review, and Approval Process

As the final approvals are obtained, the Activity Lead's focus begins to shift toward making the activity happen in flight. They will work with several members of the uplink team to find a logical time for the activity to be included in a plan given all the constraints and cross-team impacts that were identified during the ActID review process. As the planning day for the activity draws near, the sequences that will need to be uplinked to the spacecraft are prepared for delivery to a database called SeqDB that acts as the sequence version control system.

3. Sequence Control and Delivery

When the sequences are finalized and the activity development team prepares for the internal review held by the Engineering Operations team, all the sequence files are stored in the sequence database (SeqDB). Operators who add sequences to SeqDB can do so in one of two ways. First, they can "draft" the sequence. This adds the sequence to the database but still allows for changes to be made without modifying its version. Second, the sequence can be "delivered" which starts the configuration control of the sequence and prevents any change to the sequence contents without a change in sequence version.

SeqDB is also where the tactical team will retrieve sequences when the final approvals have been obtained and it is time to execute the activity in flight. To bridge the gap between strategic planning and tactical planning, MSL has a supratactical planning process that balances the needs of all the various teams to come up with a high-level overview of what the spacecraft will be doing in the coming two weeks [3]. Once the activity has been scheduled by the Supratactical Lead and the planning day has arrived, it is the job of the team staffing the planning shift that day to understand the activity is scheduled to be included in that planning cycle, pull in the relevant sequences from SeqDB, and ensure the activity criteria and planning constraints outlined in the ActID are upheld while the plan is being built. In cases where the activity is particularly complex or carries unusual risk, the Activity Lead will attend tactical planning to ensure proper activity inclusion.

Since SeqDB does not allow changes to a delivered sequence, this has always aided in sequence traceability. Once a sequence has been delivered, all the planning team needs to do is include the correct version of the sequence in the plan. Before final approvals are given at the end of the strategic activity review process discussed above, it is required that the Activity Lead have all sequences delivered to SeqDB and documented in the ActID. What is missing from this architecture is a guarantee that what is being delivered as part of the review matches what was tested in the testbed. One might be able to mitigate this by requiring sequences to be delivered to SeqDB prior to uplinking them to the testbed for testing, but there are several things that make that approach impractical. First, requiring this would have a large impact on the rate of iteration and progress in test sessions. Second, it would flood SeqDB with sequences that have not been tested and are certainly not safe for flight. While there are several safeguards in place that prevent the team from delivering an incorrect sequence, having a database full of invalid and untested sequences is not good practice. Finally, there is no easy or practical way to enforce this change given how we've built our test support infrastructure. The sequence traceability tool that is the topic of this paper will address this gap without adding complexity to testbed operations.

4. Sequence Compilation and Uplink

As all the plan's details are finalized and the tactical planning cycle concludes, the team must prepare the various sequences and files for uplink to the rover. Typically, the files are sent via one of the Deep Space Network stations directly to the rover's X-band High Gain Antenna in the Martian morning. To understand the significance of the Sequence Checksum and how it can improve sequence traceability, we must first understand how a sequence comes to be used by the spacecraft.

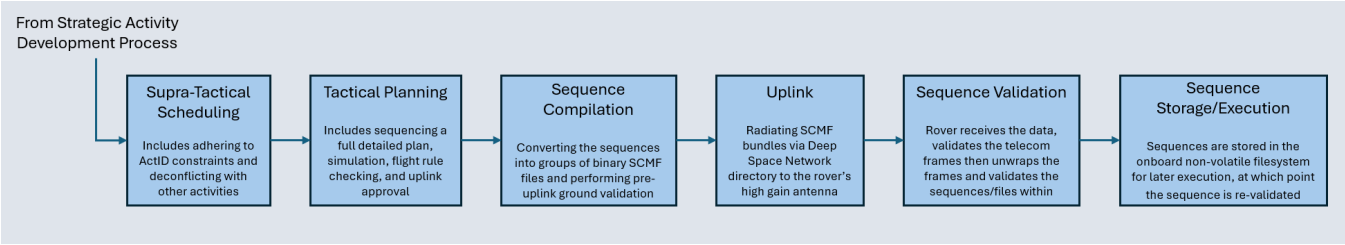


Figure 3. A simplified process diagram of sequence handling after an activity has been approved for use in flight

Sequence Content File and Robot Markup Language File

In its most basic form, sequence contents are stored in plain text in a file format called a Sequence Content File (.seq file extension). The Sequence Content File contains the commands and arguments that makeup a sequence as well as sequence comments (analogous to code comments). Multiple sequence files are aggregated in the sequence editor by the operator into an XML-based file type called the Robot Markup Language (RML) File (.rml file extension). RML files still contain all the commands, arguments, and comments of the Sequence Content Files, but adds metadata that will be used later to properly compile the sequence into its binary format including dictionary version and sequence type flags that are used by Flight Software for validation and storage. When compiling sequences that will go to the spacecraft, the Sequence Integration Engineer is responsible for aggregating sequences from each team into an integrated RML file, which will contain all the sequences that are to be transmitted to the spacecraft for that particular plan.

Sequence Compilation

Command sequences are transmitted and executed in a compiled, binary format. A series of tools developed internally at JPL take an RML file as input and produce a series of binary Spacecraft Message Files (SCMFs) as well as several other intermediate products. The set of SCMFs for all the sequences and files going to the spacecraft for that planning cycle is the key deliverable at the end of each planning session. An SCMF contains the binary sequence data to be transmitted to the spacecraft which includes header content used by Flight Software for validation.

Transmission and FSW Validation and Use

The SCMFs that are ready to be uplinked are packaged into appropriately formatted frames to be sent to the spacecraft [7] and transmitted via one of the Deep Space Network (DSN) stations at a predetermined time when the rover is expected to be waiting for an X-band signal. The inclusion of the SCMFs into these frames adds another layer of link overhead in the form of a frame header. This frame header is used by the rover's Telecommunication Interface Card to determine if the frame is valid, error-free, and intended for the receiving spacecraft before proceeding. Once the Telecommunication Interface Card validates the frame and hands the contents to Flight Software, another validation occurs with the header from the SCMF before the sequence contents are stored for later use in the non-volatile file system. While there are multiple points at which data is validated and checksums are used to verify data integrity, it is at this point that an entirely different kind of checksum is calculated and reported by flight software. This additional checksum, called the Sequence Checksum, is the last puzzle piece that will allow us to implement the sequence traceability tool.

5. THE SEQUENCE CHECKSUM

The Sequence Checksum is an MD5 checksum [9] computed over the binary copy of a sequence (every compiled command and argument, but not the initial header) and is distict from the checksum used by FSW to validate the entire sequence file with the header. MD5 is a 16-byte checksum, but MSL only uses the 4 high-order bytes.

This section describes the original motivation for adding Sequence Checksums to MSL flight software, and some of the design decisions that led to the current implementation.

Genesis

The Sequence Checksum was added to MSL Flight Software (FSW) to address a problem that occurred during Spirit Mars Exploration Rover (MER) flight operations. On sol 1237, MER Rover Planners (RPs) successfully created and uplinked three arm control sequences, but they inadvertently neglected to flag the names of those sequences as having been used in their online database. As a result, on sol 1238 a new set of RPs attempted to uplink arm control sequences that reused the same names as the prior sol's sequences. At the time MER RP sequence delivery tools were configured to minimize uplink volume by rejecting new deliveries if the sequence was already onboard. As a result, on sol 1238 the new sequences were *not* delivered and the prior sol's sequences ran again. Fortunately, rerunning the prior sequences did not result in any physical damage; a minor error in the execution of the 1237 sequences had left the arm in an error state that precluded additional motion, so no actual arm motion occurred on sol 1238.

However, there was an impact on the downlink analysis. To save downlink bandwidth, when executing commands the MER FSW only reports the name of the command and its line number within a sequence. The MER downlink teams had automated the annotation of FSW Event Report text messages (EVRs) by retrieving RML files from the same-sol's delivery areas in the filesystem and merging command details from that original RML file into the list of messages [10]. During the sol 1238 downlink that process led to some very confusing annotations, as the same-named sequences onboard (from sol 1237) had run completely different commands than those stored in the sol 1238 filesystem RMLs. So, the annotations were completely incorrect.

That event led to the idea that FSW should also describe sequences by their *contents*, not only their names, to be sure we understand exactly what is running onboard. So MSL Sequencing FSW was updated to include Sequence Checksums everywhere: in EVR text messages, data products, and channelized telemetry. That makes it possible to automatically validate the sequence contents even from a small amount of downlinked telemetry; there is no longer any

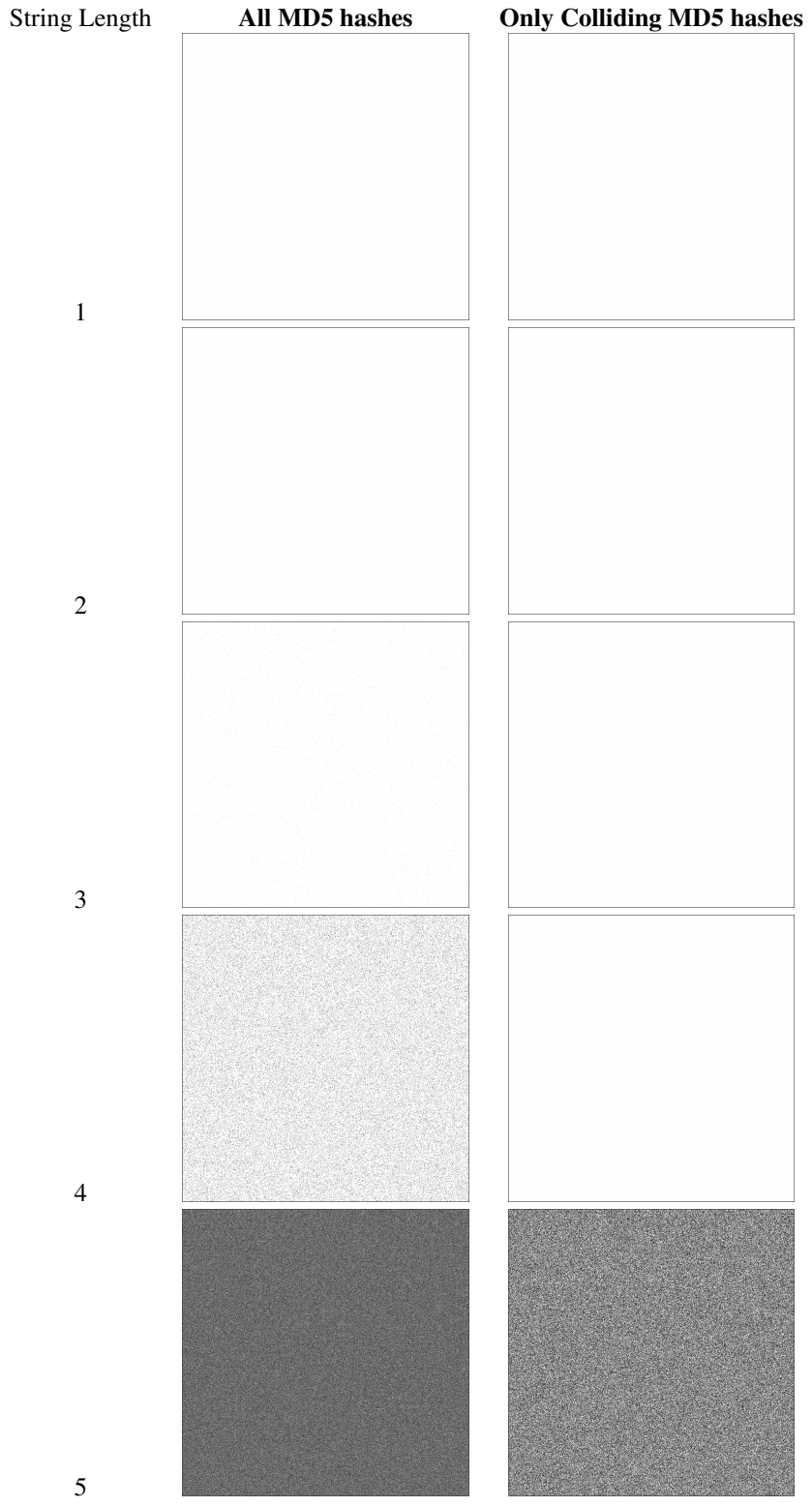


Figure 4. MD5 Randomness images. These visualize the distribution of actual 4-byte MD5 hash prefixes in the space of 2^{32} possible values. Each row R in the figure corresponds to the hashes of all possible strings of length R using uppercase alpha, numeric, and underscore characters (a subset of the characters allowed in command stems and arguments). The left image in each row shows every hash value computed from R-character strings; the right image shows all collisions (hashes which represent more than one string). Images are 2^{10}x2^{10} pixels, and each pixel represents 2^{12} adjacent hash values; the more hash values that were generated in that pixel's range, the darker the pixel. Each image is normalized to render the greatest sum within it using the darkest pixel value. As expected, the distribution of hash values is nicely spread throughout each image, suggesting that in practice the number of Sequence Checksum collisions will be small.

need to guess which version of a sequence was run.

Development

Using Sequence Checksums to document uplinked sequences is far more robust than relying on human-maintained version numbers. The FSW-reported checksum is derived from the commands and arguments within a sequence, not extraneous header information that doesn't change commanded behavior. Only meaningful sequence changes that impact what the spacecraft does will be reflected in the checksum reported by the FSW.

MD5 was chosen over simpler XOR-style checksums due to its preferred hashing characteristics: small changes in an input result in large changes to the resulting hash value. This has made it popular as a method for uniquely "fingerprinting" files by their contents, e.g., in GitHub at the time it was added (October 2008). However, we were unable to accommodate the full 128-bit hash simply into our telemetry, and instead chose to use just the 32-bit prefix of MD5 hashes generated by the algorithm [9]. As a sanity check, we tested the MD5 prefix hashes of short length alphanumeric strings to see how well they were distributed through the space of 2^{32} possible values. We chose to visualize the results in images, several of which are shown in Figure 4. They suggested that even the 32-bit prefix of the MD5 Checksum would be well-distributed with few collisions. And since we never plan to run different versions of the same-named sequence on any given sol, using a combination of (sequence name, Sequence Checksum) to identify sequences can only result in at most a few thousand combinations, virtually ensuring there will be no collisions (different sequences encoding to the same hash value) over the lifetime of the mission.

FSW had previously only reported sequence names and flight version numbers for sequences, but we added the Sequence Checksum information to many parts of the downlink telemetry: text-based Event Report messages (EVRs), new channelized telemetry channels, and data product binary files that describe the onboard sequence contents. This addition to the MSL FSW also persisted when the Mars 2020 rover FSW and ground tools were being built. While initial ground tool use of Sequence Checksums on MSL was limited to downlink annotation tools, SENDIT now makes use of the Sequence Checksum in the uplink planning and strategic review process.

6. THE TOOL: SENDIT

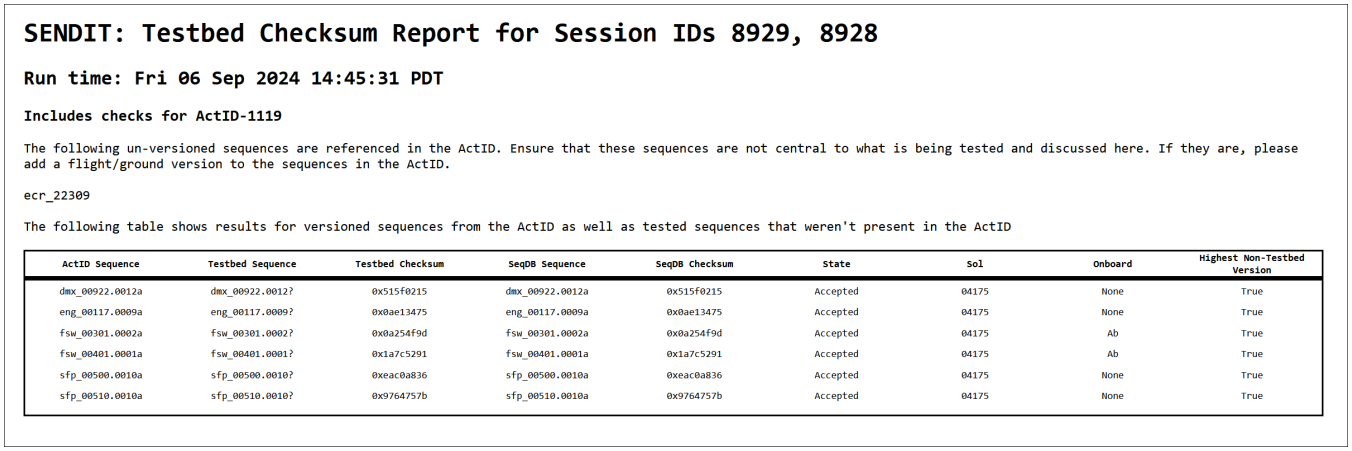
Even though a formalized review process and a sequence database aid in sequence traceability, they are unable to provide much coverage while the sequences are being developed and tested iteratively. When the desire to further increase sequence traceability prior to sequence delivery was first considered, the Sequence Checksum quickly became seen as the missing link that would be most helpful in the tool's development. Sequence Checksums are reported each time a new sequence is received or run in the spacecraft's event telemetry, making it easy to distinguish between similarly named and versioned sequences that have different contents. We permanently record this telemetry not just for the flight vehicle, but also for all the testbeds. Each time a testbed is used by a tester, a unique session ID is created that can be used to query testbed data only from that specific test effort. Each ActID comes clearly marked with any relevant sequences and the session IDs of any testbed sessions used to validate the activity, meaning a tool could be built to quickly analyze all the testbed data and compare it to the contents of sequences delivered to SeqDB.

The tool, named Sequence Equivalence Navigator Designed to Increase Traceability (or SENDIT), now had a basic design. The tool would take as inputs the ActID number the user is interested in as well as any testbed session IDs to be used to analyze testbed data. It would then read the ActID contents to find any sequences that are part of the activity, query testbed data to look for sequences that were run during testing, and query SeqDB to obtain information about sequences found in the ActID and testbed data. Once all that data is analyzed, a table could be produced that would quickly alert the user to potential issues with the sequences or the testing methodology, vastly simplifying the review process and making it easier to focus on relevant material in the strategic activity review.

Tool Output

Figure 5, shows an HTML report example of the tool's output. In addition to this HTML report, a similarly formatted report is printed to the terminal that contains the same information. The HTML format can be useful for uploading the report to the ActIDs as an attachment. Each row in the table represents a sequence. The first column shows the sequence ID and full version (flight version and ground version) that the tool found while parsing the contents of the ActID. The second column shows the sequence ID and version of a matching sequence that was seen executing in testbed telemetry. Note that this column does not include the ground version. When sequence execution is reported in telemetry, sequence ID, checksum, and flight version are noted in event telemetry but not the ground version. The spacecraft has no knowledge of the ground version of a sequence, so no presumption is made that the ground version the operator sent matches what is shown in the ActID. The third column shows the Sequence Checksum taken directly from testbed telemetry. The fourth column notes the sequence ID and version of a sequence found in SeqDB that matches the sequence ID, flight version, and checksum. The fifth column shows the checksum of the sequence it was matched with from SeqDB. Note that this value must be computed by the tool, as Sequence Checksum is not stored in the database since what version of Flight Software it was compiled for could change the checksum. The rest of the columns are pieces of metadata stored in SeqDB that can clarify the state of a given sequence. This includes the state of the sequence in SeqDB (options are drafted, delivered, accepted, descoped, and rejected), the most recent sol to which the sequence was delivered, the onboard state of the sequence (referring to whether it is currently known to be on the flight rover and which computer it is on), and whether it is the highest version in SeqDB. This last piece of metadata can be particularly helpful because the convention is to only increase the flight and ground versions as new versions of a sequence are delivered for use in flight. If a sequence version were not the highest of its particular sequence ID, that could be the result of someone mistakenly using an outdated sequence.

In addition to the main contents of the report, messages routinely appear above the table to alert operators and reviewers to important information that is helpful for interpreting the report. For the ActID shown in Figure 5, only one such message was included. For this report, it cautions that an un-versioned sequence was referenced in the ActID. In many cases, someone may mention a sequence in the activity description for context and not because it is going to be part of the activity. This is only cause for concern if the



SENDIT: Testbed Checksum Report for Session IDs 8929, 8928

Run time: Fri 06 Sep 2024 14:45:31 PDT

Includes checks for ActID-1119

The following un-versioned sequences are referenced in the ActID. Ensure that these sequences are not central to what is being tested and discussed here. If they are, please add a flight/ground version to the sequences in the ActID.

ecr_22309

The following table shows results for versioned sequences from the ActID as well as tested sequences that weren't present in the ActID

ActID Sequence	Testbed Sequence	Testbed Checksum	SeqDB Sequence	SeqDB Checksum	State	Sol	Onboard	Highest Non-Testbed Version
dmx_00922.0012a	dmx_00922.0012?	0x515f0215	dmx_00922.0012a	0x515f0215	Accepted	04175	None	True
eng_00117.0009a	eng_00117.0009?	0x0ae13475	eng_00117.0009a	0x0ae13475	Accepted	04175	None	True
fsw_00301.0002a	fsw_00301.0002?	0x0a254f9d	fsw_00301.0002a	0x0a254f9d	Accepted	04175	Ab	True
fsw_00401.0001a	fsw_00401.0001?	0x1a7c5291	fsw_00401.0001a	0x1a7c5291	Accepted	04175	Ab	True
sfp_00500.0010a	sfp_00500.0010?	0xeac0a836	sfp_00500.0010a	0xeac0a836	Accepted	04175	None	True
sfp_00510.0010a	sfp_00510.0010?	0x9764757b	sfp_00510.0010a	0x9764757b	Accepted	04175	None	True

Figure 5. SENDIT HTML report for ActID-1119 which updated the spacecraft's fault protection communication windows

Activity Lead omitted version information necessary for the ActID. Additional caution messages can be displayed to alert operators to sequences that could not be properly compiled to have their checksum computed, SeqDB queries that were unable to provide the relevant RML file, and instances where there are multiple ground versions that have a matching Sequence Checksum.

For the ActID shown in Figure 5, the objective was to update the timing and duration of the spacecraft's fault protection communication windows [8]. This ActID is fairly straightforward and did not require many sequences or complicated testing. As a result, the report shows that none of the sequences have tripped a warning. Figure 6 shows a more exciting report for ActID-1073. This ActID replaced several onboard reusable drilling sequences that needed to be updated to be compatible with a new FSW version that was installed in April 2023 [6].

A quick review of this report shows that several table cells have been highlighted in orange. This is a visual indicator that there might be an issue with a particular sequence. In this case, the sequence in the leftmost column is highlighted as well as at least one other cell in that row to show the operator why the sequence was flagged. A sequence in this report can be flagged for one of 8 reasons.

1. Sequences that appear in the ActID but are not found in testbed data will be highlighted and "NOT TESTED" will be shown in the "Testbed Sequence" column. This is an issue because the reviewer might otherwise have the impression that all ActID sequences were tested but it could be hard to confirm. Conditional sequencing or other factors could create a scenario where everyone presumed a sequence was tested when it was not.

2. Not all the sequences necessary for testing need to be referenced in the ActID. For example, an engineering sequence that is run daily may have run in the testbed to make testing more realistic but is not part of the activity. If that is the case, any sequence used in testing should already be onboard the spacecraft. So, sequences not in the ActID and not already onboard will be highlighted and the text "NOT IN ACTID/ONBOARD" will appear in the "ActID Sequence" column.

3. For each sequence included in the ActID, SENDIT will look through SeqDB for a Sequence Checksum match to the tested version. If the tool was unable to find a sequence whose checksum matched what was tested, it will be highlighted and "NO CHECKSUM MATCH" will be included in the "SeqDB Sequence" column.

4. If a sequence is not included in the ActID but was invoked during testing, SENDIT will also look to find a Sequence Checksum match. If the tool is unable to find a checksum match in this scenario, "CHECKSUM NOT FOUND" will be included in the "SeqDB Sequence" column.

5. SENDIT assumes that it will only need to check sequences that match a flight version to find a checksum match. For example, if the ActID and testbed telemetry both reference flight version 1 of sequence ID eng_00100, only flight version 1 sequences will be evaluated for a match (eng_00100.0001a, eng_00100.0001b, etc.). In the event the SeqDB query returns no sequences with that flight version, the sequence is highlighted and "VERSION NOT IN SEQDB" is shown in the "SeqDB Sequence" column.

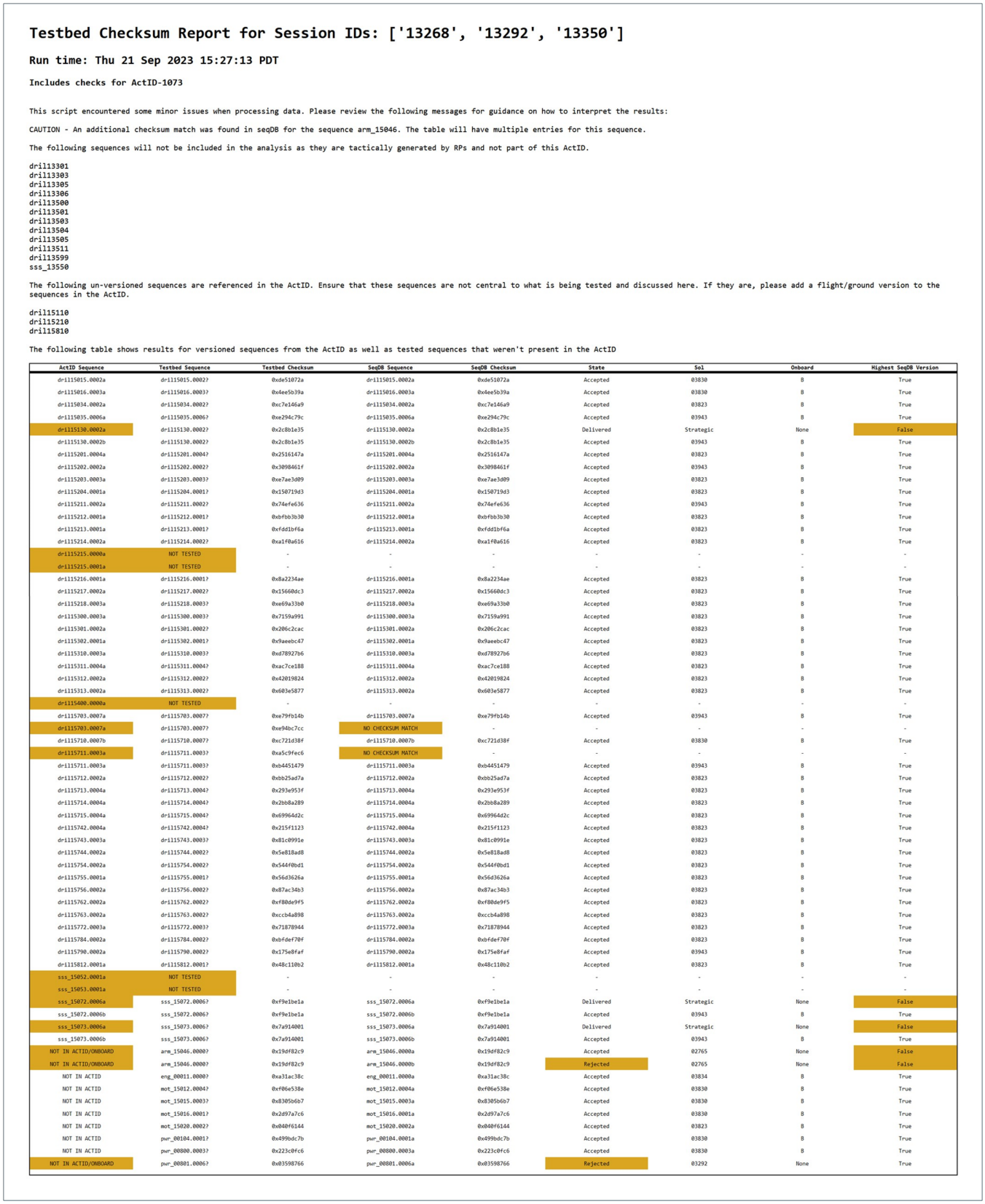
6. If the value of the "Highest Non-Testbed Version" column is False, the sequence is highlighted. A sequence that is not the highest version could indicate someone mistakenly tested or delivered an outdated sequence. For example, eng_00101.0000a may be referenced in documentation, but if eng_00101.0001f exists in SeqDB already it is possible that the wrong sequence version is being referenced. The term non-testbed creates an exception for sequences that are delivered to a special sol in SeqDB called "testbed". This accounts for nuanced internal rules for how SeqDB is typically used and is outside the scope of this discussion.

7. Any sequence that has a SeqDB state of "Rejected" is highlighted. Rejected sequences have been placed into that state by a Sequence Integration Engineer on a tactical shift and indicates that the sequence is not good for uplink and should never be used.

8. Any sequence that has a SeqDB state of "Descoped" is highlighted. Descoped sequences have been placed in that state by a Sequence Integration Engineer on a tactical shift and indicates that the sequence is not good for uplink for the current planning session but could be used in the future. However, the team's sequence versioning conventions make it unusual that it would be used again in the future.

Three Key Insights from the Increased Traceability

The ability to generate this level of detailed reporting in a matter of seconds increases sequence traceability and makes



Testbed Checksum Report for Session IDs: ['13268', '13292', '13350']

Run time: Thu 21 Sep 2023 15:27:13 PDT

Includes checks for ActID-1073

This script encountered some minor issues when processing data. Please review the following messages for guidance on how to interpret the results:

CAUTION - An additional checksum match was found in seqDB for the sequence arm_15046. The table will have multiple entries for this sequence.

The following sequences will not be included in the analysis as they are tactically generated by RPs and not part of this ActID.

drill13301
drill13303
drill13305
drill13306
drill13500
drill13501
drill13503
drill13504
drill13505
drill13511
drill13599
sss_13550

The following un-versioned sequences are referenced in the ActID. Ensure that these sequences are not central to what is being tested and discussed here. If they are, please add a flight/ground version to the sequences in the ActID.

drill5110
drill5210
drill5810

The following table shows results for versioned sequences from the ActID as well as tested sequences that weren't present in the ActID

ActID Sequence	Testbed Sequence	Testbed Checksum	SeqDB Sequence	SeqDB Checksum	State	Sol	Onboard	Highest SeqDB Version
drill5015.0002a	drill5015.0002?	0xde51072a	drill5015.0002a	0xde51072a	Accepted	03830	B	True
drill5016.0003a	drill5016.0003?	0x4ee5b39a	drill5016.0003a	0x4ee5b39a	Accepted	03830	B	True
drill5034.0002a	drill5034.0002?	0xc7e146a9	drill5034.0002a	0xc7e146a9	Accepted	03823	B	True
drill5035.0006a	drill5035.0006?	0xe294c79c	drill5035.0006a	0xe294c79c	Accepted	03943	B	True
drill5130.0002a	drill5130.0002?	0x2c8b1e35	drill5130.0002a	0x2c8b1e35	Delivered	Strategic	None	False
drill5130.0002b	drill5130.0002?	0x2c8b1e35	drill5130.0002b	0x2c8b1e35	Accepted	03943	B	True
drill5201.0004a	drill5201.0004?	0x2516147a	drill5201.0004a	0x2516147a	Accepted	03823	B	True
drill5202.0002a	drill5202.0002?	0x3098461f	drill5202.0002a	0x3098461f	Accepted	03943	B	True
drill5203.0003a	drill5203.0003?	0xe7ae3d09	drill5203.0003a	0xe7ae3d09	Accepted	03823	B	True
drill5204.0001a	drill5204.0001?	0x150719d3	drill5204.0001a	0x150719d3	Accepted	03823	B	True
drill5211.0002a	drill5211.0002?	0x74efe636	drill5211.0002a	0x74efe636	Accepted	03943	B	True
drill5212.0001a	drill5212.0001?	0xbfbb3b30	drill5212.0001a	0xbfbb3b30	Accepted	03823	B	True
drill5213.0001a	drill5213.0001?	0xfdd1bf6a	drill5213.0001a	0xfdd1bf6a	Accepted	03823	B	True
drill5214.0002a	drill5214.0002?	0xa1f0a616	drill5214.0002a	0xa1f0a616	Accepted	03823	B	True
drill5215.0000a	NOT TESTED	-	-	-	-	-	-	-
drill5215.0001a	NOT TESTED	-	-	-	-	-	-	-
drill5216.0001a	drill5216.0001?	0x8a2234ae	drill5216.0001a	0x8a2234ae	Accepted	03823	B	True
drill5217.0002a	drill5217.0002?	0x15660dc3	drill5217.0002a	0x15660dc3	Accepted	03823	B	True
drill5218.0003a	drill5218.0003?	0xe69a33b0	drill5218.0003a	0xe69a33b0	Accepted	03823	B	True
drill5300.0003a	drill5300.0003?	0x7159a991	drill5300.0003a	0x7159a991	Accepted	03823	B	True
drill5301.0002a	drill5301.0002?	0x206c2cac	drill5301.0002a	0x206c2cac	Accepted	03823	B	True
drill5302.0001a	drill5302.0001?	0x9aeebc47	drill5302.0001a	0x9aeebc47	Accepted	03823	B	True
drill5310.0003a	drill5310.0003?	0xd78927b6	drill5310.0003a	0xd78927b6	Accepted	03823	B	True
drill5311.0004a	drill5311.0004?	0xac7ce188	drill5311.0004a	0xac7ce188	Accepted	03823	B	True
drill5312.0002a	drill5312.0002?	0x42019824	drill5312.0002a	0x42019824	Accepted	03823	B	True
drill5313.0002a	drill5313.0002?	0x603e5877	drill5313.0002a	0x603e5877	Accepted	03823	B	True
drill5400.0000a	NOT TESTED	-	-	-	-	-	-	-
drill5703.0007a	drill5703.0007?	0xe79fb14b	drill5703.0007a	0xe79fb14b	Accepted	03943	B	True
drill5703.0007a	drill5703.0007?	0xe94bc7cc	NO CHECKSUM MATCH	-	-	-	-	-
drill5710.0007b	drill5710.0007?	0xc721d38f	drill5710.0007b	0xc721d38f	Accepted	03830	B	True
drill5711.0003a	drill5711.0003?	0xa5c9fec6	NO CHECKSUM MATCH	-	-	-	-	-
drill5711.0003a	drill5711.0003?	0xb4451479	drill5711.0003a	0xb4451479	Accepted	03943	B	True
drill5712.0002a	drill5712.0002?	0xbb25ad7a	drill5712.0002a	0xbb25ad7a	Accepted	03823	B	True
drill5713.0004a	drill5713.0004?	0x293e953f	drill5713.0004a	0x293e953f	Accepted	03823	B	True
drill5714.0004a	drill5714.0004?	0x2bb8a289	drill5714.0004a	0x2bb8a289	Accepted	03823	B	True
drill5715.0004a	drill5715.0004?	0x69964d2c	drill5715.0004a	0x69964d2c	Accepted	03823	B	True
drill5742.0004a	drill5742.0004?	0x215f1123	drill5742.0004a	0x215f1123	Accepted	03823	B	True
drill5743.0003a	drill5743.0003?	0x81c0991e	drill5743.0003a	0x81c0991e	Accepted	03823	B	True
drill5744.0002a	drill5744.0002?	0x5e818ad8	drill5744.0002a	0x5e818ad8	Accepted	03823	B	True
drill5754.0002a	drill5754.0002?	0x544f0bd1	drill5754.0002a	0x544f0bd1	Accepted	03823	B	True
drill5755.0001a	drill5755.0001?	0x56d3626a	drill5755.0001a	0x56d3626a	Accepted	03823	B	True
drill5756.0002a	drill5756.0002?	0x87ac34b3	drill5756.0002a	0x87ac34b3	Accepted	03823	B	True
drill5762.0002a	drill5762.0002?	0xf80de9f5	drill5762.0002a	0xf80de9f5	Accepted	03823	B	True
drill5763.0002a	drill5763.0002?	0xccb4a898	drill5763.0002a	0xccb4a898	Accepted	03823	B	True
drill5772.0003a	drill5772.0003?	0x71878944	drill5772.0003a	0x71878944	Accepted	03823	B	True
drill5784.0002a	drill5784.0002?	0xbfdef70f	drill5784.0002a	0xbfdef70f	Accepted	03823	B	True
drill5790.0002a	drill5790.0002?	0x175e8faf	drill5790.0002a	0x175e8faf	Accepted	03943	B	True
drill5812.0001a	drill5812.0001?	0x48c110b2	drill5812.0001a	0x48c110b2	Accepted	03823	B	True
sss_15052.0001a	NOT TESTED	-	-	-	-	-	-	-
sss_15053.0001a	NOT TESTED	-	-	-	-	-	-	-
sss_15072.0006a	sss_15072.0006?	0xf9e1be1a	sss_15072.0006a	0xf9e1be1a	Delivered	Strategic	None	False
sss_15072.0006b	sss_15072.0006?	0xf9e1be1a	sss_15072.0006b	0xf9e1be1a	Accepted	03943	B	True
sss_15073.0006a	sss_15073.0006?	0x7a914001	sss_15073.0006a	0x7a914001	Delivered	Strategic	None	False
sss_15073.0006b	sss_15073.0006?	0x7a914001	sss_15073.0006b	0x7a914001	Accepted	03943	B	True
NOT IN ACTID/ONBOARD	arm_15046.0000?	0x19df82c9	arm_15046.0000a	0x19df82c9	Accepted	02765	None	False
NOT IN ACTID/ONBOARD	arm_15046.0000?	0x19df82c9	arm_15046.0000b	0x19df82c9	Rejected	02765	None	False
NOT IN ACTID	eng_00011.0000?	0xa31ac38c	eng_00011.0000a	0xa31ac38c	Accepted	03834	B	True
NOT IN ACTID	mot_15012.0004?	0xf06e538e	mot_15012.0004a	0xf06e538e	Accepted	03830	B	True
NOT IN ACTID	mot_15015.0003?	0x8305b6b7	mot_15015.0003a	0x8305b6b7	Accepted	03830	B	True
NOT IN ACTID	mot_15016.0001?	0x2d97a7c6	mot_15016.0001a	0x2d97a7c6	Accepted	03830	B	True
NOT IN ACTID	mot_15020.0002?	0x040f6144	mot_15020.0002a	0x040f6144	Accepted	03823	B	True
NOT IN ACTID	pwr_00104.0001?	0x499bdc7b	pwr_00104.0001a	0x499bdc7b	Accepted	03830	B	True
NOT IN ACTID	pwr_00800.0003?	0x223c0fc6	pwr_00800.0003a	0x223c0fc6	Accepted	03830	B	True
NOT IN ACTID/ONBOARD	pwr_00801.0006?	0x03598766	pwr_00801.0006a	0x03598766	Rejected	03292	None	True

Figure 6. A more complicated SENDIT HTML report for ActID-1073 which updated reusable drill sequences for a new version of Flight Software

it easy to find three key issues. Each of these issues can be connected to one or more of the reasons a sequence would be highlighted discussed in the previous section.

1. The report identifies any sequences whose tested checksums do not match what is delivered for use in flight. This could mean that an incorrect sequence was delivered or that the version that was tested did not properly include sequence changes made toward the end of development.

2. The report identifies sequences that were used in testing but are not part of the ActID. This could indicate that there are sequences not documented as part of the ActID that are required for the activity to complete successfully. While there are tactical tools in place to ensure that every sequence called is bundled, not having it as part of the documentation that is approved would likely prevent the activity from being included in a plan if it is not caught before planning begins. The activity would have to go back through reviews and rescheduled for tactical.

3. The report identifies sequences that are in the ActID but were never invoked in testing. This means there is a gap in testing and the activity has not been fully validated. While this may seem unlikely, the conditional sequencing capability available on MSL (running certain commands only when an "if statement" condition is met for example) means it is easy to miss that a conditional branch was never tested.

These insights have already proved to be very helpful at several points in the activity development and review process. For example, a tester can run it after completing their testing but before preparing documentation for a review to ensure that all the sequences were properly tested and included in the ActID. To highlight another use case, a reviewer can run the report prior to arriving at the internal review to help them formulate questions and get a sense of anything unusual about the sequences that were tested.

Dispositioning Warnings and Common Rationale

Since each of the highlighted sequences represents a potential issue or mistake, it has become routine to produce a write-up to accompany the tool output that dispositions each of the caution messages and highlighted sequences. This can be helpful because there are scenarios that SENDIT would highlight and warn the operator about a given sequence that might be acceptable. Common occurrences that have been seen to date while using the tool include:

- If a mistake is discovered during testing, the report will typically show multiple lines for a single sequence with different checksums. This is due to the operator making sequence content changes and re-running the sequence. In this case, one line will show "NO CHECKSUM MISMATCH" and one will show as a match. The reviewers need to make sure that the updated version of the sequence was the one delivered to the sequence database, which is typically addressed during the test data review portion of the internal review. For an example of this scenario, see the sequence dril_15703 in Figure 6.

- If an Activity Lead references an old sequence version in the ActID documentation, it will be included for analysis since the tool has not been given the ability to account for the context surrounding the sequence ID in the ActID. In most cases, this will result in the older version of the sequence being shown on the report as "NOT TESTED" despite its irrelevance to the ActID. Refer to the sequence sss_15072 for an example of this scenario in Figure 6.

- The MSL project has several different testbeds that each have their advantages and disadvantages when it comes to their effectiveness as a test venue. No testbed is a perfect replica of the flight vehicle on Mars so there are some sequences which must have different contents in the testbed. For example, the sequence pwr_00801 in Figure 6 shows that testing used a rejected version of the sequence that is not currently onboard. This sequence changes the value of power parameters after drilling occurs, but because of the differences of the configuration between the rover on Mars and the testbed on Earth that is not powered with a Radioisotope Thermoelectric Generator, the parameters need to be set to different values than what they are in flight.

- In some cases, it is impractical or dangerous to attempt to force a specific conditional sequencing path to be run in the testbed. In limited cases, the team may decide to allow sequences to go untested provided that they are limited in complexity, able to be inspected, and have previous testbed or flight data that validates the commanding approach. An example of this case can be seen in Figure 6 for the sequence dril15215_.0001a. In this case, the sequence is responsible for decrementing the percussion level used in drilling [2] in the event the speed with which drilling depth is achieved through a sample is above a certain threshold. It would be difficult for operators to repeatably provide a rock sample that allowed this sequence conditional to be met, so it was highlighted by the tool and reviewed by inspection in front of the internal reviewers.

- Sometimes, sequences used specifically for testbed setup are highlighted as "NOT IN ACTID/ONBOARD" to caution operators that the sequence may have been inadvertently left off the ActID. The sequence arm_15046 in Figure 6 is an example of a setup sequence that was run in the testbed but is not required to be used in flight.

- One of the caution messages that can appear at the top mentions when multiple ground version checksum matches are found for a specific sequence. Out of an abundance of caution, the tool will currently show all ground versions with a checksum match, meaning those sequences will have multiple rows in the table. While this doesn't have a practical impact since the Sequence Checksum being unchanged between versions means the commanded behavior is not different, it could alleviate confusion as to which ground version is intended for uplink. See arm_15046 in Figure 6 for an example of this scenario.

With relatively few exceptions, the report output should be clean and show few potential issues. Pairing the SENDIT output with disposition discussions allow operators and reviewers to know where to focus their efforts as they prepare the activity for flight and will ultimately lead to higher-quality discussions during the review and approval process.

Analysis Algorithm and Output Generation

Having presented the finer detail of the tool's output, we now describe how SENDIT was architected and how it completes the required analysis. SENDIT makes extensive use of the existing mission system environment where many other similar tools and scripts run on Red Hat Linux machines. The tool is written in Python3 and makes use of modules from the Python Standard Library to interface with the operating system, parse and store data, handle HTML requests, and invoke other MSL python and shell scripts. The tool also relies on MSL-specific infrastructure to query testbed telemetry, query SeqDB, and access the ActID content.

Invoking SENDIT is done from the command line and requires that the operator provide the testbed machine host name, the testbed database host, the testbed database port, and the test session ID. If the operator wishes to have the tools analyze the testbed and sequence data against an ActID, they must also provide the numeral portion of the activity identifier for the ActID.

Once SENDIT validates input, it starts collecting and organizing data from the various sources. First, it sends an HTML request to the MSLReports server to obtain the text from the specified ActID and parses that content to pull out any text strings that are in the format of a sequence ID. Second, it queries testbed data for specific event records that are generated each time a new command is dispatched for execution by FSW. This event record includes the sequence ID, flight version number, and the Sequence Checksum. These three pieces of information are used to build a list of unique sequences that were run in the testbed session. From here, sequences that are known to be testbed setup sequences or are meant to mimic tactically generated sequences that are not a part of the activity are removed from consideration. By comparing the list of sequences referenced by the ActID and the sequences found in testbed data, the sequences are split into three categories. Sequences that have been tested and appear in the ActID, sequences that were testbed but do not appear in the ActID, and sequences that were not tested but appear in the ActID. Having these split into three categories at this point will make later analysis more straightforward.

With lists of all the sequences we want information about, we can now start querying SeqDB for relevant detail. Queries are run for each unique sequence ID and flight version combination. In addition to a variety of metadata, SeqDB also provides the location on the mission's ground filesystem where an RML file containing that sequence can be found. Since Sequence Checksums cannot be stored statically in SeqDB (it can but doesn't always change depending on the FSW version against which it is compiled), we must compile the sequences as if we will be sending them to a spacecraft or testbed in order to calculate the checksum. All sequence files are aggregated into a temporary directory where they are compiled into SCMFs. We can then take the relevant portion of the binary SCMF file and compute the MD5 checksum and record the 4 most significant bytes as the Sequence Checksum.

With a complete list of potentially relevant checksums from SeqDB, a complete listing of sequences and checksums from the testbed, and context from the ActID, we can now complete the analysis to provide increased sequence traceability. For each sequence that was tested (two of the three categories we made earlier), SENDIT attempts to find a checksum match. Based on what is found and whether the sequence was found in the ActID, the rest of the table is populated with a warning or metadata from matched sequences. If there is a match at this point, an additional SeqDB comparison is made to determine if the sequence is the highest version currently delivered to a non-testbed sol. Once all the tested sequences have been analyzed, sequences from the non-tested category are added to the table.

With the table completed, the appearance of the cells that meet warning criteria are adjusted to highlight any concerning issues. Next, the main report content is printed to the terminal and saved into an HTML file for later reference. Temporary directories containing the sequences and the compiled products are cleared and execution completes.

Additional Example and Report Discussion

Figure 7 shows another example report. In ActID-1108, a sequence used to recover from drill attempts that experience a fault while the bit is still inside the rock was updated to be compatible with a new version of Flight Software [5]. This sequence is typically kept onboard for easy usage, so it was being stored onboard in the rover's sequence library instead of being invoked solely in the planning cycle that it was uplinked.

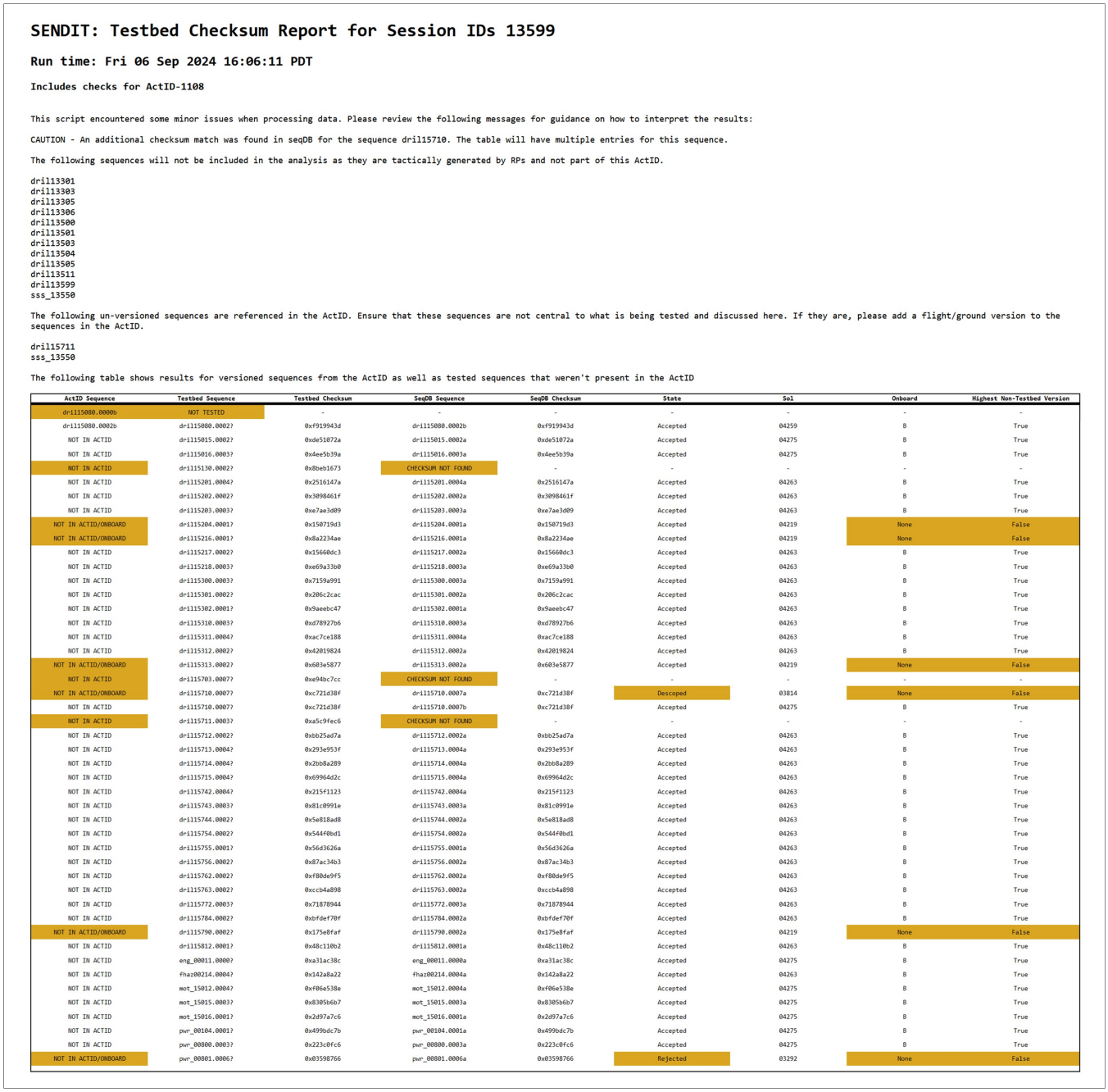
Many of the warnings and dispositions in this example are of a similar nature to those seen in ActID-1073. There are two dispositions of note here that help illustrate the usefulness of this tool. First, dril15130 is highlighted because no matching checksum was found in SeqDB. Since this activity was meant to test bit retraction after an anomalous drill attempt, the team had to figure out how they could reliably introduce a fault and stop the drill sequencing. The method they settled on was to modify an existing drill sequence (dril15130) and place a command that would mimic a fault and preclude future drilling. This would cause the rest of the drill attempt to instantly halt, placing the testbed in a representative anomalous condition where the retraction sequences could be tested. Since this sequence was modified by the testers the day of testing and they did not deliver it to SeqDB, it makes sense this would be the result. Second is a warning that was also present in ActID-1073 but was not discussed yet. The sequences dril15703 and dril15711 both show as having no matching checksum in SeqDB. Both sequences alter parameters related to the Force Torque Sensors present on the robotic arm. While the sensors in use in the flight rover and the testbed are the same, the one installed on testbed happens to have a lower measurement quality. This requires slightly different parameter settings to be used in the testbed. In this report, these minor but expected differences result in an inability to find a checksum match.

This additional example is also helpful to show how useful the tool can be for making a small number of sequence changes to complicated spacecraft actions. This report is lengthy not because the ActID will involve a large number of sequences but because the underlying action of setting up the testbed and drilling in a flight-like way is complicated. Upon closer inspection, one can see that there is only one new sequence that is delivered as part of ActID-1108 (dril15080). Even though many of the sequences on the report have nothing to do with this ActID, understanding that the correct versions of those sequences were run helps the team evaluate the validity of the test.

Alternate Modes

Up to this point, our focus has been on the tool's most typical usage mode. While SENDIT is most useful when pairing the testbed data and SeqDB information with an ActID, the tool can also be run in an alternate mode that does not require an ActID. Previously, we discussed the three key insights provided by the increased sequence traceability of this tool. Two of them dealt with alerting the operator about inconsistencies between testbed data and the ActID. There is one insight that can still be gained even if ActID documentation for the related sequences does not exist. Without a documented ActID, we can still determine whether the sequences that have been tested in a given test session match what is delivered for use in flight or onboard the spacecraft.

Figure 8 shows the report generated in this alternative mode for the same ActID that was analyzed in Figure 7. In this



SENDIT: Testbed Checksum Report for Session IDs 13599

Run time: Fri 06 Sep 2024 16:06:11 PDT

Includes checks for ActID-1108

This script encountered some minor issues when processing data. Please review the following messages for guidance on how to interpret the results:

CAUTION - An additional checksum match was found in seqDB for the sequence dril15710. The table will have multiple entries for this sequence.

The following sequences will not be included in the analysis as they are tactically generated by RPs and not part of this ActID.

dril13301
dril13303
dril13305
dril13306
dril13500
dril13501
dril13503
dril13504
dril13505
dril13511
dril13599
sss_13550

The following un-versioned sequences are referenced in the ActID. Ensure that these sequences are not central to what is being tested and discussed here. If they are, please add a flight/ground version to the sequences in the ActID.

dril15711
sss_13550

The following table shows results for versioned sequences from the ActID as well as tested sequences that weren't present in the ActID

ActID Sequence	Testbed Sequence	Testbed Checksum	SeqDB Sequence	SeqDB Checksum	State	Sol	Onboard	Highest Non-Testbed Version
dril15080.0000b	NOT TESTED	-	-	-	-	-	-	-
dril15080.0002b	dril15080.0002?	0xf919943d	dril15080.0002b	0xf919943d	Accepted	04259	B	True
NOT IN ACTID	dril15015.0002?	0xde51072a	dril15015.0002a	0xde51072a	Accepted	04275	B	True
NOT IN ACTID	dril15016.0003?	0x4ee5b39a	dril15016.0003a	0x4ee5b39a	Accepted	04275	B	True
NOT IN ACTID	dril15130.0002?	0x8beb1673	CHECKSUM NOT FOUND	-	-	-	-	-
NOT IN ACTID	dril15201.0004?	0x2516147a	dril15201.0004a	0x2516147a	Accepted	04263	B	True
NOT IN ACTID	dril15202.0002?	0x3098461f	dril15202.0002a	0x3098461f	Accepted	04263	B	True
NOT IN ACTID	dril15203.0003?	0xe7ae3d09	dril15203.0003a	0xe7ae3d09	Accepted	04263	B	True
NOT IN ACTID/ONBOARD	dril15204.0001?	0x150719d3	dril15204.0001a	0x150719d3	Accepted	04219	None	False
NOT IN ACTID/ONBOARD	dril15216.0001?	0x8a2234ae	dril15216.0001a	0x8a2234ae	Accepted	04219	None	False
NOT IN ACTID	dril15217.0002?	0x15660dc3	dril15217.0002a	0x15660dc3	Accepted	04263	B	True
NOT IN ACTID	dril15218.0003?	0xe69a33b0	dril15218.0003a	0xe69a33b0	Accepted	04263	B	True
NOT IN ACTID	dril15300.0003?	0x7159a991	dril15300.0003a	0x7159a991	Accepted	04263	B	True
NOT IN ACTID	dril15301.0002?	0x206c2cac	dril15301.0002a	0x206c2cac	Accepted	04263	B	True
NOT IN ACTID	dril15302.0001?	0x9aeebc47	dril15302.0001a	0x9aeebc47	Accepted	04263	B	True
NOT IN ACTID	dril15310.0003?	0xd78927b6	dril15310.0003a	0xd78927b6	Accepted	04263	B	True
NOT IN ACTID	dril15311.0004?	0xac7ce188	dril15311.0004a	0xac7ce188	Accepted	04263	B	True
NOT IN ACTID	dril15312.0002?	0x42019824	dril15312.0002a	0x42019824	Accepted	04263	B	True
NOT IN ACTID/ONBOARD	dril15313.0002?	0x603e5877	dril15313.0002a	0x603e5877	Accepted	04219	None	False
NOT IN ACTID	dril15703.0007?	0xe94bc7cc	CHECKSUM NOT FOUND	-	-	-	-	-
NOT IN ACTID/ONBOARD	dril15710.0007?	0xc721d38f	dril15710.0007a	0xc721d38f	Descoped	03814	None	False
NOT IN ACTID	dril15710.0007?	0xc721d38f	dril15710.0007b	0xc721d38f	Accepted	04275	B	True
NOT IN ACTID	dril15711.0003?	0xa5c9fec6	CHECKSUM NOT FOUND	-	-	-	-	-
NOT IN ACTID	dril15712.0002?	0xbb25ad7a	dril15712.0002a	0xbb25ad7a	Accepted	04263	B	True
NOT IN ACTID	dril15713.0004?	0x293e953f	dril15713.0004a	0x293e953f	Accepted	04263	B	True
NOT IN ACTID	dril15714.0004?	0x2bb8a289	dril15714.0004a	0x2bb8a289	Accepted	04263	B	True
NOT IN ACTID	dril15715.0004?	0x69964d2c	dril15715.0004a	0x69964d2c	Accepted	04263	B	True
NOT IN ACTID	dril15742.0004?	0x215f1123	dril15742.0004a	0x215f1123	Accepted	04263	B	True
NOT IN ACTID	dril15743.0003?	0x81c0991e	dril15743.0003a	0x81c0991e	Accepted	04263	B	True
NOT IN ACTID	dril15744.0002?	0x5e818ad8	dril15744.0002a	0x5e818ad8	Accepted	04263	B	True
NOT IN ACTID	dril15754.0002?	0x544f0bd1	dril15754.0002a	0x544f0bd1	Accepted	04263	B	True
NOT IN ACTID	dril15755.0001?	0x56d3626a	dril15755.0001a	0x56d3626a	Accepted	04263	B	True
NOT IN ACTID	dril15756.0002?	0x87ac34b3	dril15756.0002a	0x87ac34b3	Accepted	04263	B	True
NOT IN ACTID	dril15762.0002?	0xf80de9f5	dril15762.0002a	0xf80de9f5	Accepted	04263	B	True
NOT IN ACTID	dril15763.0002?	0xccb4a898	dril15763.0002a	0xccb4a898	Accepted	04263	B	True
NOT IN ACTID	dril15772.0003?	0x71878944	dril15772.0003a	0x71878944	Accepted	04263	B	True
NOT IN ACTID	dril15784.0002?	0xbfdef70f	dril15784.0002a	0xbfdef70f	Accepted	04263	B	True
NOT IN ACTID/ONBOARD	dril15790.0002?	0x175e8faf	dril15790.0002a	0x175e8faf	Accepted	04219	None	False
NOT IN ACTID	dril15812.0001?	0x48c110b2	dril15812.0001a	0x48c110b2	Accepted	04263	B	True
NOT IN ACTID	eng_00011.0000?	0xa31ac38c	eng_00011.0000a	0xa31ac38c	Accepted	04275	B	True
NOT IN ACTID	fhaz00214.0004?	0x142a8a22	fhaz00214.0004a	0x142a8a22	Accepted	04263	B	True
NOT IN ACTID	mot_15012.0004?	0xf06e538e	mot_15012.0004a	0xf06e538e	Accepted	04275	B	True
NOT IN ACTID	mot_15015.0003?	0x8305b6b7	mot_15015.0003a	0x8305b6b7	Accepted	04275	B	True
NOT IN ACTID	mot_15016.0001?	0x2d97a7c6	mot_15016.0001a	0x2d97a7c6	Accepted	04275	B	True
NOT IN ACTID	pwr_00104.0001?	0x499bdc7b	pwr_00104.0001a	0x499bdc7b	Accepted	04275	B	True
NOT IN ACTID	pwr_00800.0003?	0x223c0fc6	pwr_00800.0003a	0x223c0fc6	Accepted	04275	B	True
NOT IN ACTID/ONBOARD	pwr_00801.0006?	0x03598766	pwr_00801.0006a	0x03598766	Rejected	03292	None	False

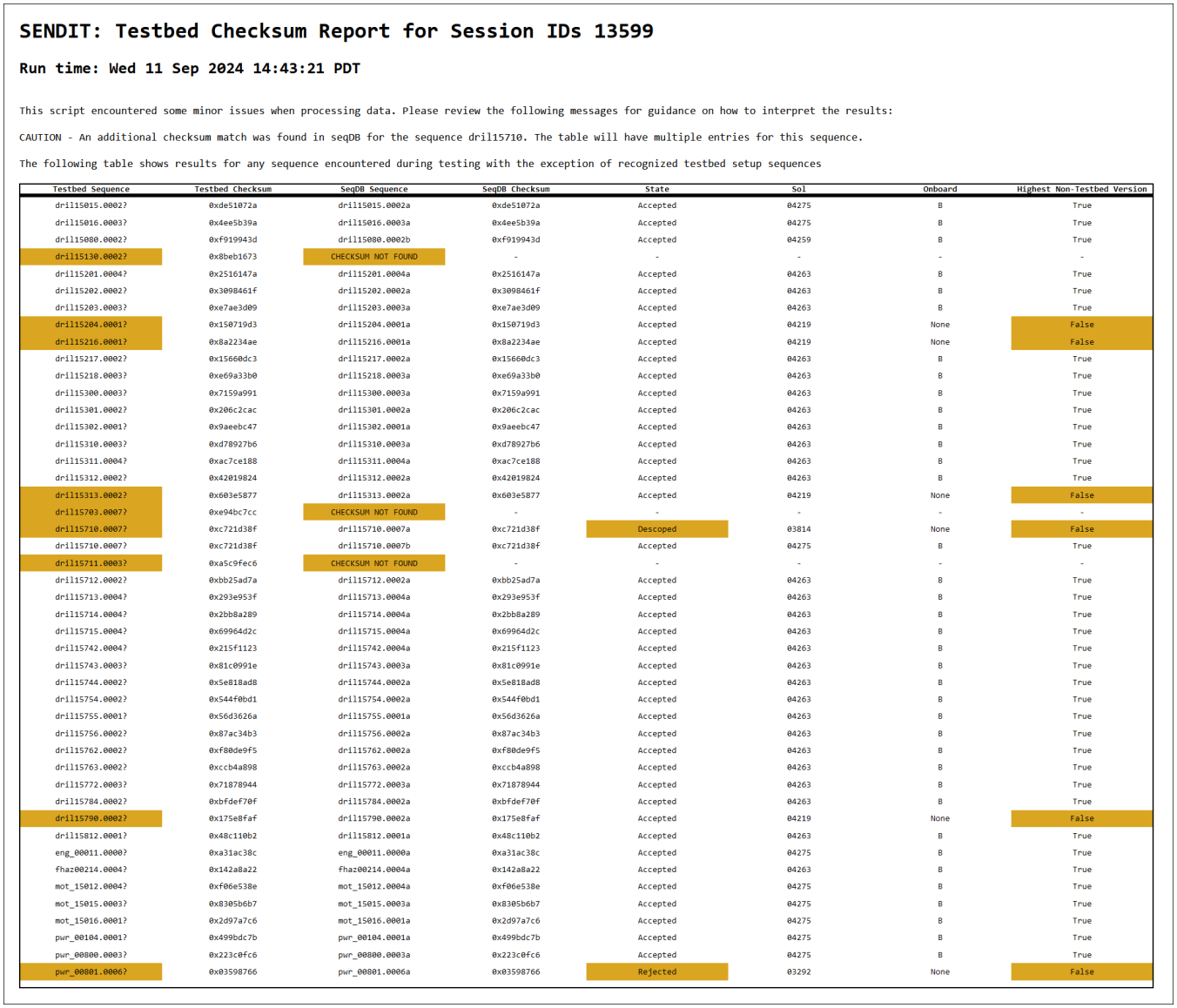
Figure 7. SENDIT HTML report for ActID-1108 which updated off-nominal drill recovery sequences for a new version of Flight Software

mode, SENDIT focuses on all the sequences run in the test sessions and tries to find the corresponding sequences in SeqDB. While information provided in the ActID is not a part of this analysis, many of the potential issues highlighted in Figure 7 can also be seen in Figure 8.

7. Conclusion

When the operations concept and Mission System for Curiosity were designed, careful consideration was given to protect against unwanted sequence changes or inadvertent sequence uplink by implementing sequence controls and checks. While this original design is still effective, extending this sequence control and traceability by allowing a way to check sequences against testbed data as they enter the original controls makes it easier for operators to test and deliver complex activities. While it is impossible to tell whether a given mistake could survive the myriad checks that exist between review and uplink, the tool has allowed the team to find issues much earlier in the development process while significantly reducing the time investment required to check for such mistakes. Perhaps most significantly, an increase in the complexity of an activity no longer means that this process is more difficult or time consuming because of this tool.

While this tool has already proved its usefulness to the MSL team, there are several areas where future work is possible to



SENDIT: Testbed Checksum Report for Session IDs 13599

Run time: Wed 11 Sep 2024 14:43:21 PDT

This script encountered some minor issues when processing data. Please review the following messages for guidance on how to interpret the results:

CAUTION - An additional checksum match was found in seqDB for the sequence dril15710. The table will have multiple entries for this sequence.

The following table shows results for any sequence encountered during testing with the exception of recognized testbed setup sequences

Testbed Sequence	Testbed Checksum	SeqDB Sequence	SeqDB Checksum	State	Sol	Onboard	Highest Non-Testbed Version
dril15015.0002?	0xde51072a	dril15015.0002a	0xde51072a	Accepted	04275	B	True
dril15016.0003?	0x4ee5b39a	dril15016.0003a	0x4ee5b39a	Accepted	04275	B	True
dril15080.0002?	0xf919943d	dril15080.0002b	0xf919943d	Accepted	04259	B	True
dril15130.0002?	0x8beb1673	CHECKSUM NOT FOUND	-	-	-	-	-
dril15201.0004?	0x2516147a	dril15201.0004a	0x2516147a	Accepted	04263	B	True
dril15202.0002?	0x3098461f	dril15202.0002a	0x3098461f	Accepted	04263	B	True
dril15203.0003?	0xe7ae3d09	dril15203.0003a	0xe7ae3d09	Accepted	04263	B	True
dril15204.0001?	0x150719d3	dril15204.0001a	0x150719d3	Accepted	04219	None	False
dril15216.0001?	0x8a2234ae	dril15216.0001a	0x8a2234ae	Accepted	04219	None	False
dril15217.0002?	0x15660dc3	dril15217.0002a	0x15660dc3	Accepted	04263	B	True
dril15218.0003?	0xe69a33b0	dril15218.0003a	0xe69a33b0	Accepted	04263	B	True
dril15300.0003?	0x7159a991	dril15300.0003a	0x7159a991	Accepted	04263	B	True
dril15301.0002?	0x206c2cac	dril15301.0002a	0x206c2cac	Accepted	04263	B	True
dril15302.0001?	0x9aeebc47	dril15302.0001a	0x9aeebc47	Accepted	04263	B	True
dril15310.0003?	0xd78927b6	dril15310.0003a	0xd78927b6	Accepted	04263	B	True
dril15311.0004?	0xac7ce188	dril15311.0004a	0xac7ce188	Accepted	04263	B	True
dril15312.0002?	0x42019824	dril15312.0002a	0x42019824	Accepted	04263	B	True
dril15313.0002?	0x603e5877	dril15313.0002a	0x603e5877	Accepted	04219	None	False
dril15703.0007?	0xe94bc7cc	CHECKSUM NOT FOUND	-	-	-	-	-
dril15710.0007?	0xc721d38f	dril15710.0007a	0xc721d38f	Descoped	03814	None	False
dril15710.0007?	0xc721d38f	dril15710.0007b	0xc721d38f	Accepted	04275	B	True
dril15711.0003?	0xa5c9fec6	CHECKSUM NOT FOUND	-	-	-	-	-
dril15712.0002?	0xbb25ad7a	dril15712.0002a	0xbb25ad7a	Accepted	04263	B	True
dril15713.0004?	0x293e953f	dril15713.0004a	0x293e953f	Accepted	04263	B	True
dril15714.0004?	0x2bb8a289	dril15714.0004a	0x2bb8a289	Accepted	04263	B	True
dril15715.0004?	0x69964d2c	dril15715.0004a	0x69964d2c	Accepted	04263	B	True
dril15742.0004?	0x215f1123	dril15742.0004a	0x215f1123	Accepted	04263	B	True
dril15743.0003?	0x81c0991e	dril15743.0003a	0x81c0991e	Accepted	04263	B	True
dril15744.0002?	0x5e818ad8	dril15744.0002a	0x5e818ad8	Accepted	04263	B	True
dril15754.0002?	0x544f0bd1	dril15754.0002a	0x544f0bd1	Accepted	04263	B	True
dril15755.0001?	0x56d3626a	dril15755.0001a	0x56d3626a	Accepted	04263	B	True
dril15756.0002?	0x87ac34b3	dril15756.0002a	0x87ac34b3	Accepted	04263	B	True
dril15762.0002?	0xf80de9f5	dril15762.0002a	0xf80de9f5	Accepted	04263	B	True
dril15763.0002?	0xccb4a898	dril15763.0002a	0xccb4a898	Accepted	04263	B	True
dril15772.0003?	0x71878944	dril15772.0003a	0x71878944	Accepted	04263	B	True
dril15784.0002?	0xbfdef70f	dril15784.0002a	0xbfdef70f	Accepted	04263	B	True
dril15790.0002?	0x175e8faf	dril15790.0002a	0x175e8faf	Accepted	04219	None	False
dril15812.0001?	0x48c110b2	dril15812.0001a	0x48c110b2	Accepted	04263	B	True
eng_00011.0000?	0xa31ac38c	eng_00011.0000a	0xa31ac38c	Accepted	04275	B	True
fhaz00214.0004?	0x142a8a22	fhaz00214.0004a	0x142a8a22	Accepted	04263	B	True
mot_15012.0004?	0xf06e538e	mot_15012.0004a	0xf06e538e	Accepted	04275	B	True
mot_15015.0003?	0x8305b6b7	mot_15015.0003a	0x8305b6b7	Accepted	04275	B	True
mot_15016.0001?	0x2d97a7c6	mot_15016.0001a	0x2d97a7c6	Accepted	04275	B	True
pwr_00104.0001?	0x499bdc7b	pwr_00104.0001a	0x499bdc7b	Accepted	04275	B	True
pwr_00800.0003?	0x223c0fc6	pwr_00800.0003a	0x223c0fc6	Accepted	04275	B	True
pwr_00801.0006?	0x03598766	pwr_00801.0006a	0x03598766	Rejected	03292	None	False

Figure 8. SENDIT HTML report for ActID-1108 which was generated without providing the ActID as input

further improve and extend the tool's capabilities. Currently, the tool uses simple rules and references configuration files to look for sequences that should be omitted from the analysis and how to deal with multiple checksum matches. The tool takes a conservative approach to avoid removing a relevant sequence and avoid under-reporting checksum matches, but the sophistication of determining what sequences are relevant to the testing or ActID could be increased. This would reduce clutter and false warnings on the report. Another potential area for improvement is the logic used to parse the ActID contents to pull out relevant activity sequences. Relying on a regular expression match to find text strings that likely refer to a sequence frequently results in the tool trying to analyze a sequence that doesn't exist or was just mentioned in the ActID description for context. Improvements could also be made in the tool's knowledge of known sequences or checksums that are meant for the testbeds only and are not safe for flight. While this situation hasn't caused an issue for MSL to date, it would be an easy one to protect against by keeping a list of known sequences that were altered to accommodate testbed hardware. Use of these sequences in testing would also prompt the review team to ask about the relevant parts of the activity and give extra scrutiny to the test data to ensure the differences with the flight vehicle were properly considered.

The final area for potential future work is the adaptation or creation of a more standardized tool that can be used elsewhere at JPL or other institutions that asynchronously operate robots in space and non-space environments. By showing examples from commanding the Curiosity rover, this paper establishes that complete traceability for commanding of any remotely operated robotic system can vastly simplify complex operations and significantly reduce the risk for command error. SENDIT it illustrates one of many ways to accomplish this by using the Sequence Checksum, however, the idea of the Sequence Checksum has only been implemented to date in the rover Flight Software of MSL and Mars 2020. Other missions could decide to address the same problem by implmenting thier own version of the Sequence Checksum and SENDIT, by designing a sequence database to begin configuration control earlier in the development process, or

more heavily invest in simulation to include low-level system behavior and surface interaction in a way that makes it clearer when unintended sequence changes are introducted. No matter what form the solution takes, it should make it easy to verify that when testing results in the desired outcome, all the sequencing that produces that outcome matches what is in the queue to be uplinked to the spacecraft.

ACKNOWLEDGMENTS

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

The authors would like to thank the Mars Science Laboratory for supporting this work.

REFERENCES

[1] A. H. Mishkin, D. Limonadi, S. L. Laubach and D. S. Bass, "Working the Martian night shift - the MER surface operations process," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 46-53, June 2006, doi: 10.1109/MRA.2006.1638015

[2] R. Kinnett, T. Green, D. Klein, M. Richardson Lin, "Remote Diagnosis and Operational Response to an In-Flight Failure of the Drill Feed Mechanism Onboard the Mars Science Laboratory Rover," Proceedings of the 46th Aerospace Mechanisms Symposium, Virtual, May 11-13, 2022

[3] M. Gildner et al., "Commanding Curiosity from the Couch: MSL Remote Operations, Challenges, and Path Ahead," 2021 IEEE Aerospace Conference

[4] M. Muszynski, E. Ferguson and S. Wissler, "The Evolution of Command and Sequencing at JPL: Origins and Flight Software Core Lineage," 2023 IEEE Aerospace Conference

[5] A. Holloway, J. Denison, N. Patel, M. Maimone and A. Rankin, "Six Years and 184 Tickets: The Vast Scope of the Mars Science Laboratory's Ultimate Flight Software Release," 2023 IEEE Aerospace Conference

[6] R. Larsen, et al., "Look Before You Leap: Installing R13 on the Curiosity Mars Rover," 2024 IEEE Aerospace Conference

[7] Mars Science Laboratory Project, "Uplink and Command Functional Design Description Design Document (FDD)", Revision B, JPL Internal, 2009.

[8] K. Rink, R. Boehmer, K. Kaplan, R. Larsen, J. Clark and T. Neilson, "Martian Mayday: The Evolution of Curiosity's Safe Mode Communication Over Ten Years," 2024 IEEE Aerospace Conference

[9] R. Rivest, "RFC 1321 - The MD5 Message-Digest Algorithm" , `http://www.faqs.org/rfcs/rfc1321.html`, April, 1992.

[10] Jeffrey Biesiadecki, Robert Liebersbach, Mark Maimone, "Mars Exploration Rover Mobility and IDD Downlink Analysis Tools," International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS) Proceedings, Los Angeles, CA, 27 February 2008.

BIOGRAPHY

Jonathan Denison *is the creator of SENDIT and has been a member of the MSL Operations team at the Jet Propulsion Laboratory since 2018. He has contributed as a Systems Engineer to the Data Management, Systems, Flight Software, and Ground Software (Teamtools) teams on MSL and as of 2022 is the MSL Engineering Operations Team Chief. Prior to joining JPL in 2016, Jonathan worked on aviation operations software and naval radar systems. Jonathan earned his B.S. in Aeronautical and Astronautical Engineering and M.B.A. from The Ohio State University as well as a M.S. in Computer Science from the University of Southern California.*

Mark Maimone *is a JPL Principal in Autonomous Planetary Rover Navigation, Mars 2020 Robotic Operations Deputy Team Chief, member of the Rover Planner and FSW development teams, and a Robotic Systems Engineer in the Robotic Mobility group at the Jet Propulsion Laboratory. Mark received NASA Exceptional Achievement Medals for designing and implementing GESTALT self-driving Flight Software for MER and MSL; contributed to the Mars 2020 self-driving Flight Software; during MSL operations served as Deputy Lead Rover Planner, Lead Mobility Rover Planner and Flight Software Lead, and contributed to the Sequencing and Parameter flight software modules; and developed downlink automation tools for MER and MSL. Mark is also a member of NASA's Lunar Terrain Vehicle Insight team. Mark holds a Ph.D. in Computer Science from Carnegie Mellon University, and has also developed navigation and image processing capabilities for robots in Chornobyl and the Atacama Desert.*