AN APPROACH TOWARDS SUPPORTING SEAMLESS SEARCH ACROSS PDS3 AND PDS4 METADATA. K. Grimes[1], A. Waldron[1], R. Verma[1], C. DeCesare[1], P. Ramirez[1], J. Padams[1], S. Hardman[1], M. Cayanan[1], [1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. (Kevin.M.Grimes@jpl.nasa.gov)

Introduction: The Planetary Data Systems (PDS) Cartography and Imaging Sciences Node's (IMG) archives retain hundreds of terabytes of planetary imagery, complete with metadata formatted according to the PDS3 data standard. As a consequence of the homogeneity of the archive, the tooling used by IMG to store, process, and search this data has become increasingly dependent on the simple key-value structure defined by PDS3.

With missions such as *InSight* and *Mars 2020* preparing to deliver data adhering to the new, XML-based PDS4 standard, IMG's archive is expected to become much more diverse; in anticipation of this change, IMG has been able to extend the Image Atlas–a web-based tool that enables user search by PDS label values–to support searches across both PDS3- and PDS4-formatted metadata, while minimizing risk of human error by automating the majority of the processes.

Cross-Standard Search: Existing Image Atlas functionality allows users to search by hundreds of different PDS3 keywords across over a dozen planetary missions. Label metadata is stored and organized in an Apache Solr index in such a way that search is quick, accurate, and simple.

To enable this search, the current PDS3 data ingestion model requires several hours of pre-processing to take place, including 1 to 5 hours of manual effort per release, depending on the complexity of the process. Because of the human component, the processes are prone to error: a simple typo can impede the data ingestion workflow and consequently delay the release of the data.

The advances made in Image Atlas functionality presented here are twofold: first, the data ingestion model massages the label metadata in such a way that cross-standard searches may be done; second, the manual component of the data ingestion is eliminated using modern DevOps methodologies and technologies, including Docker, Ansible , and XL Release.

Updated Data Ingestion Model: Apache Solr is the search platform that powers the Image Atlas. It creates an index from a data source and a schema, which may then be searched.

Pre-PDS4, IMG maintained a suite of mission-specific metadata extraction scripts. These scripts would loop through PDS3-compliant image labels, extracting information as it found it. This metadata would then be stored in a MySQL database, where it would wait to be indexed by Apache Solr. Once indexed, the databases would not be touched again. While functional, these scripts were prone to break; additionally, every time a new mission delivered data to IMG, a new set of ingestion scripts would need to be developed. Finally, IMG stored the label metadata in a total of three locations: in the Solr index, in the MySQL database, and in the labels themselves.

Metadata extraction: Rather than continue to develop ad hoc scripts for metadata extraction, the team the has chosen to leverage PDS Engineering Node (ENG) tools going forward; namely, Harvest-Search and Search Service. Where before new scripts would need to be written to parse out label information, the Harvest-Search tool works nearly out-of-the-box on any data set that conforms to the PDS4 XML schema.

In order for the Harvest-Search tool to extract all metadata properly from the PDS4 labels, its configuration file must have within it a list of PDS4 XPaths and "slot names", which are easy-to-understand common names that describe the data being extracted. Another IMG effort, the Label Mapping Tool (LMT), has the ability to store and provide mappings between slot names and XPaths; therefore, the team is able to generate Harvest-Search configuration files programmatically.

Once Harvest-Search is run, it ingests the PDS4 label metadata into an intermediary Solr index. At this stage, the PDS4 data is fully searchable; however, to enable searching across both PDS3 *and* PDS4 data, additional steps were required.

Programmatic schema updates: In addition to the PDS4 index, two more indexes are created: one for PDS3 data, and one "parent" index that links the other two together. The PDS3 index is similar to the PDS4 one: it stores metadata extracted from labels (albeit via ad hoc scripts). The parent index, however, does *not* store any unique data; instead, it is nearly wholly comprised of Solr `copyFields`.

These special fields exist to have other fields' data dynamically copied into it, effectively allowing multiple fields to be searched via a single field. An example is included in Figure 1.

As with the Harvest-Search configuration file, the schemas defining the PDS3 and parent indexes are updated programmatically using the LMT and the Solr schema API. One schema updates are done, the data is indexed and stored in the respective indexes, and is ready to be searched by the end user.

Increased Process Automation: The majority of the steps required of the PDS3 data processing procedure must be completed manually and in a predefined order. Additionally, the steps can be quite complicated, requiring in some cases the operations lead to manually correct errors in the MySQL database. With the added complexity of PDS4, and the availability of a plethora of tooling, IMG engaged in an effort to upgrade its procedures to be as automated and risk-averse as possible.

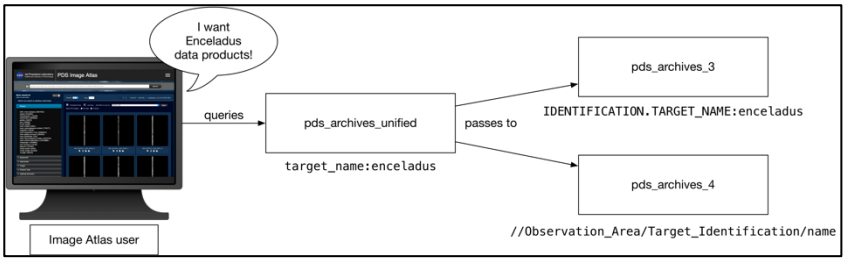

Figure 1: The common name `target_name` points to its PDS3 keyword and PDS4 XPath equivalents.

Isolated runs with Docker: When data releases are done, they typically use a handful of shared resources, such as databases, Python installations, and the like. While there is no fundamental issue with this approach, using unique copies of tools on a per-release basis has several advantages: errors are isolated from previous runs, runs are easy to reproduce, and deployment is environment-agnostic.

Because of these advantages, IMG decided to run nearly every action within its own Docker container. Should any given action fail, the release is immediately halted and any artifacts that were being used by it are dumped to a separate area on disk for debugging. Developers may then spin up copies of the action that failed on their local machines using the same initial conditions, and once they are confident the issue is resolved, resume the operations pipeline.

Configuration management with Ansible: Ansible is an open-source tool developed and maintained by Redhat that makes configuration management easy. Users can define a sequence of tasks to be executed on any number of machines simultaneously.

Currently, IMG uses Ansible to copy build resources onto the build server from a centralized location, and to trigger the actual build. The majority of the effort done is building Docker images and running initialization scripts on them; however, Ansible is capable of much more complex tasks.

Hands-off releases with XL Release: Many aspects of the data processing procedure do not require much human interaction. As such, it is a great candidate for process automation tools such as XL Release.

XL Release is an enterprise solution that automates the tedious and mundane portions of release procedures. Users define templates that outline the various actions that need to be taken, such as: running a shell script, invoking an Ansible play, creating a Github pull request, sending an e-mail, and kicking off a Jenkins job. All of these tasks can be invoked manually; however, they typically require several steps and users may accidentally provide invalid information. Additionally, there may be several hours between steps, and so the pipeline would be blocked until the next time a human has the time to kick off the next step.

XL Release allows users to schedule releases to go off at a given date and time, and will run as many of the above tasks as it can without needing input from a member of the operations team. It also keeps the team updated with e-mail notifications, so they can be quick to debug any issues that may arise with the release.

Conclusions and Further Work: By leveraging PDS Engineering Node tooling, open-source products, and some enterprise solutions, IMG has been able to modernize the data processing component of its release pipeline.

Some of the many areas the IMG team is investigating are as follows:

- Using the Search Service as the Image Atlas's primary search platform,
- Performing metadata extraction via Harvest-Search in AWS [19] with PDS Engineering Node's upcoming cloud-based extractor AMPe,
- Remove software build processes from release procedure into standalone, regular Jenkins job,
- Build and deploy software in IMG container cloud

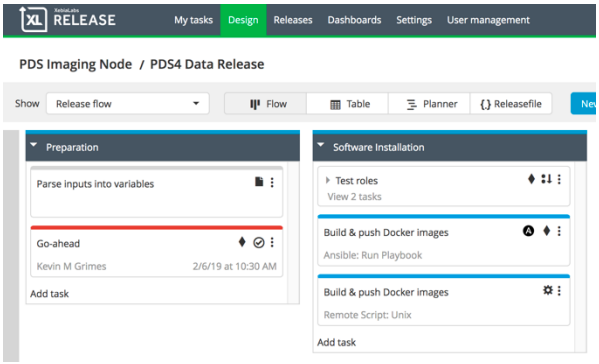

Figure 2: Partial screenshot of PDS4 data ingestion template in XL Release.