A Plan for Further Promulgation and Development of the Heliophysics Application Programmer's Interface (HAPI)

Jon Vandegriff (JHU Applied Physics Lab), Robert Weigel (George Mason University), Jeremy Faden (Cottage Systems)

2021-05-12

The Heliophysics Application Programming Interface (HAPI) is a data serving standard that has the ability to enhance the interoperability of low-level infrastructure within the Heliophysics Data Environment. The standard has been evolving slowly over the past few years, and adoption has been also slowly increasing. More widespread adoption of HAPI is expected to help advance Heliophysics science by simplifying and making generic the processes needed to analyze diverse and distributed time series data. The next phase of promulgation for HAPI will require a shift from the development of the specification to targeted support for implementers. Both server and client implementers will benefit from guidance and some specific tools to help make existing data available via HAPI and infusing HAPI capability into existing scientific workflows.

The following sections outline six activities that are important for establishing HAPI as helpful infrastructure for Heliophysics science analysis.

1. Facilitate HAPI adoption by upcoming missions by providing targeted instrument team support

2. Create additional HAPI server reference implementations

3. Create a meta-server to facilitate an integrated HAPI ecosystem

4. Create higher-level services that build on the uniformity afforded by HAPI to provide infrastructure for Machine Learning

5. Develop examples for serving images

6. Continued work on HAPI specification evolution and support for core software packages

1. Facilitate HAPI adoption by upcoming missions by providing targeted instrument team support

Instrument teams could benefit from using HAPI internally in their analysis tool suites. Even if teams find the new standard useful on their own, they will still likely need support integrating new software into their existing tools. Having more teams adopt HAPI strengthens it as a standard. More importantly, providing targeted guidance in software tool design and development will give instrument teams a more straightforward path to using HAPI internally. Wider adoption of HAPI would offer these benefits to mission-focused teams and the broader Heliophysics community:

a. Teams would obtain free help with analysis tool design and development. They would still own the process and tools to be used during their primary mission phase, but the extra initial overhead of using a new standard can be overcome with targeted support to the software developers on the team. This would also require the lead scientists on the team to be convinced.

b. Many teams already use SPEDAS (written in IDL) or PySPEDAS (Python), and these tools would naturally become more useful if more and more teams use them. Both tools have some level of HAPI

integration already and this is a natural area to focus on for HAPI promulgation. The SPEDAS team is already in a pattern of targeted support to teams, and a HAPI focus could be added to this pattern.

c. Using a HAPI-based approach allows the team's data analysis system to seamlessly transition from "data-at-local-team-site" during the main mission to "data-at-the-permanent-archive" after the mission ends. If a team wholeheartedly adopts a HAPI-based approach for their main analysis toolset, this seamless transition to using the archive could have a tremendous effect. Many teams suffer a slow degradation of their tools once a mission ends. By using the same tools before and after archiving keeps the archive's central role intact, and also lets teams focus on custom tools that will keep working long-term.

d. There are longer-term benefits to having instrument teams use HAPI. There is the potential for the creation of ISTP compliant data based on the output of a HAPI server. Right now, it is usually the ISTP compliant data that already exists and is sitting behind the HAPI server, but as HAPI becomes a more primary way of serving data, and it would be possible to create a tool that uses a HAPI stream to generate ISTP-compliant CDF files. (These CDF files then could also be served again via HAPI.) This assumes that a team is using HAPI for the very early stages of data analysis, even during instrument construction and calibration, and before any ISTP metadata exists. Automating the creation of ISTP compliant metadata would be a significant time-saver for instrument teams and archive staff. A similar argument could be made for using HAPI to generate SPASE metadata. Again, HAPI by itself does not contain enough metadata to fill a SPASE record, but it would be possible to create one centralized tool that uses augmented HAPI output to generate SPASE records. Besides saving time, the automating of this process offers a stronger guarantee that the SPASE information exactly corresponds to the contents of the data files and the HAPI metadata.

New promulgation efforts should focus immediately on items a through c, and take a longer-term view on the larger issues involved in d.

A dedicated person assisting existing teams would also be useful in helping existing HAPI server owners upgrade to the latest version. This is especially important in the next year or so as changes to the specification are more likely. Institutions or groups that have created HAPI servers and will need some support for maintenance and migration to newer versions include: CCMC at Goddard, LASP, the PDS / PPI Node, ESAC in Madrid, and AMDA in Paris.

Promulgation involves also connecting to other communities - ground-based systems like SuperMAG (underway right now), INTERMAGNET (also partly done through a wrapped server), radar sites AMISR (they have expressed some interest), and SuperDARN (no contact yet).

2. Create additional HAPI server reference implementations

Having a ready-to-use HAPI server in common programming languages lowers the barrier to adoption for more data centers. Currently, the most complete and generic implementation of a HAPI server uses node.js. Some data centers would likely prefer to use other languages, such as Python or Java. Many existing data reader libraries are being migrated to Python, so having a Python HAPI server would support those efforts. Several data centers are using Java as the core of their data delivery services. The existing node.js server offers a useful framework in terms of separating the HAPI front end from back-end data retrieval modules, so this same approach should guide the development of HAPI servers in other languages.

Open-source implementations of Java and Python servers should be made available, in library form, so that multiple data providers could use these libraries to implement their own servers.

3. Create a meta-server to facilitate an integrated HAPI ecosystem

The HAPI community group is more frequently being asked questions such as “What HAPI servers are out there?” and “What data is available via HAPI?” These kinds of larger-scale questions can be addressed with a HAPI ecosystem. An overarching framework or meta-server (server-of-servers) would let people know about all existing HAPI servers and datasets. A central searching mechanism could be developed to find which datasets are available. It would be rather simple to create a single unified HAPI server that queried all known HAPI servers for results. This is not necessarily the best approach, but having an orchestration layer that organizes all known HAPI servers would allow for interesting options like this.

Promulgation of HAPI should also involve enhancing transparency about server reliability. A centralized resource for monitoring HAPI servers could clarify which servers are currently operational. Publishing and tracking reliability information will help server owners assess any reliability issues, and servers with problems can be identified and given assistance. Also, by showing outage graphs over time, providers face pressure to keep their servers running, and users can view a server’s history to gain assurance of availability.

One other possible activity in this category would be to develop a common way for HAPI servers to expose any additional metadata that is available for many datasets. Some datasets have SPASE records, and some datasets come from CDF files with ISTP metadata, and a generic way to view this metadata from multiple servers could be potentially useful.

4. Create higher-level services that build on the uniformity afforded by HAPI to provide infrastructure for Machine Learning

The uniformity with which data can be accessed via HAPI allows for generic data processing modules to be created that are independent of the storage formats of the data. Client codes for HAPI in Python, IDL, and Matlab can already access any HAPI-compliant repository. Data processing steps that begin with data from a HAPI server would then be useful across many different datasets. One area where this would be particularly useful is in the area of Machine Learning (ML).

It is widely accepted that a large fraction of any ML task is preparing the data for ingest into algorithms that are notoriously picky about numeric uniformity. The early phases of the data cleaning stage are generic and include spike removal, averaging, recasting data to a new time cadence, removing records with partial data, etc. Codes for doing all these steps exist, but are often tied to the incoming format. Providing a set of these codes that work on HAPI data would provide a baseline of generic capabilities for making ML-ready data.

Having cleaned data available again via HAPI would allow further development in the algorithm domain to create libraries of basic ML techniques that could be applied to any cleaned HAPI dataset.

5. Develop examples for serving images

The HAPI specification describes the serving of data of arbitrary dimensionality. Two-dimensional images or even spectral or hyper-spectral images could be served using HAPI. There are existing facilities in Heliophysics for serving image data, most of these focused on the Solar Data Analysis Center (SDAC) and the Virtual Solar Observatory. It may be beneficial for HAPI to also serve image data. The important aspects in making this happen would be to ensure that relevant image metadata is also conveyed, and that client codes would know how to interpret the HAPI data stream in order to reconstruct the right image-centric data structures in client code. HAPI could become useful as a standard for time series of images as well as tabular data.

6. Continued work on HAPI specification evolution and support for core software packages

One useful gauge for the relevance of a standard is the “date of last activity.” Any project not updated or maintained recently is likely not to be trusted. Specification, software, and web pages do not maintain themselves and always deteriorate with age. Similar to the way the SPASE community has maintained a low level of activity to both promote and sustain the SPASE metadata model; the HAPI community will need to sustain and refresh core areas going forward.

The HAPI specification evolves slowly, but a low level of support is needed to understand and address new issues as they arise. New instruments and measurements may bring fresh questions about how to represent new data in HAPI. Also, as HAPI begins to be implemented in more Heliophysics software frameworks, there may be issues that need changing or addressing in the specification. Especially for the next few years, HAPI needs active curation.

Possible future updates to the specification include:

a. Allow for 2-D variation in bins - some multidimensional data has bin values that vary across more than one dimension. HAPI currently supports multi-dimensional bins, but not ones that vary across more than one dimension.

b. Ways to link together related datasets - some datasets are closely related (different calibration levels, selected parameters from multiple constituent data sets, etc), and the current HAPI server catalog has no way of indicating groupings of or links between datasets.

c. Because the time cadence is of primary importance to HAPI, having a way to associate datasets that differ only by time resolution would allow for clients to be able to choose the right time resolution based on the request duration.

d. Dealing with complex numbers - this numeric type currently has no standard representation in HAPI

e. Create science-based interfaces for specific science data types that could convey rich science semantics to plot apps and data fusion apps. This would involve adding another entire semantic layer on top of HAPI.

f. A way to regularize the ways people may want to extend HAPI servers in terms of allowing constraints or filters on the data requests. Currently, HAPI does not allow any constraints other than time. Data providers may want to add other constraints, and it would be good if there was a recommended way to do this so that the HAPI does not splinter into multiple standards based on different methods of managing constraints.

The web page for describing HAPI to scientists and developers needs to be maintained. The current site is plain and needs to be improved to more clearly and rapidly communicate to scientists and developers how to take advantage of HAPI datasets. The creation of many examples and how-to guides for accessing data would be useful. Further examples on how to set up a HAPI server would also be useful. Once set up, these web pages and examples need to be maintained since the computing landscape and underpinnings are always evolving.

7. Conclusion

There is continued interest in expanding the Heliophysics data environment, and HAPI is expected to contribute to this by making data access more uniform. The above recommendations list the areas needed to both promote the HAPI specification and also widen and deepen the software capabilities that allow HAPI to serve as a pathway towards increased data interoperability and therefore enhanced Heliophysics science return.