A Standard for Accessing Heliophysics Time Series Data

Robert Weigel[1,2], Jon Vandegriff[3], Jeremy Faden[4,5], Todd King[6], D Aaron Roberts[7], Bernard Harris[7], Robert Candey[7], Nand Lal[7], Scott Boardsen[7], Chris Lindholm[8,9], Doug Lindholm[8,9], Thomas Baltzer[8,9], Larry Brown, Eric Grimes[6], Baptiste Cecconi[10], Vincent Génot[11], Benjamin Renard[12], Arnaud Masson[13], Beatriz Martinez[14]

[1]George Mason University
[2]Space Weather Lab
[3]Johns Hopkins Applied Physics Lab
[4]Cottage Systems
[5]University of Iowa
[6]University of California Los Angeles,
[7]NASA Goddard Space Flight Center
[8]University of Colorado
[9]Laboratory for Atmospheric and Space Physics
[10]LESIA, Observatoire de Paris–PSL, CNRS, Meudon, France
[11]Institut de Recherche en Astrophysique et Planétologie, CNRS, Univ. Toulouse, CNES, Toulouse, France
[12]AKKA Technologies
[13]Tpz UK for ESA, ESAC
[14]RHEA Group for ESA, ESAC

Key Points:

- There are many non-standard approaches to serving Heliophysics time series data.
- HAPI provides a standard specification that will simplify data access.
- Many software tools have been developed for servers and clients.

Corresponding author: R.S. Weigel, `rweigel@gmu.edu`

Abstract

Heliophysics data analysis often involves combining diverse science measurements, many of them captured as time series. Although there are now only a few commonly used data file formats, the diversity in mechanisms for automated access to and aggregation of such data holdings can make analysis that requires inter-comparison of data from multiple data providers difficult. The Heliophysics Application Programmer's Interface (HAPI) is a recently developed standard for accessing distributed time-series data to increase interoperability. The HAPI specification is based on the common elements of existing data services and it standardizes the two main parts of a data service: the request interface and the response data structures. The interface is based on the REpresentational State Transfer (REST) or RESTful architecture style, and the HAPI specification defines five required REST endpoints. Data are returned via a streaming format that hides file boundaries; the metadata is detailed enough for the content to be scientifically useful, e.g., plotted with appropriate axes layout, units, and labels. Multiple mature HAPI-related open-source projects offer server-side implementation tools and client-side libraries for reading HAPI data in multiple languages (IDL, Java, MATLAB, and Python). Multiple data providers in the US and Europe have added HAPI access alongside their existing interfaces, and based on this experience, data can be served via HAPI with little or no information loss compared to similar existing web interfaces. Finally, HAPI has been adopted as a recommended COSPAR standard for time series data delivery.

Plain Language Summary

The Heliophysics Application Programmer's Interface (HAPI) specification allows data providers to use a standard set of conventions for returning data in response to a URL-based request. This specification was developed because many data providers provide such data with incompatible conventions so that users wanting automated access to and aggregation of data must develop custom code for each data provider's interface. The benefit to the user is that clients for many common languages have been developed to read HAPI data; if a data provider serves data using the HAPI specification, the user does not need to write additional code to access the data. The starting point for using and learning more about HAPI is http://hapi-server.org/.

1 Introduction

Heliophysics (HP) researchers study the nature and dynamical interactions of the Sun, the heliosphere, and the plasma environments of the planets based on data from a fleet of spacecraft and ground-based platforms termed the "Heliophysics System Observatory" (HSO) (*Heliophysics Data Environment*, 2021). As we seek a more detailed understanding of the global and local dynamics of this system, we increasingly need easy access to data from many of the observatories. The best route to integrating these data is the adoption of standards for data formats, metadata terminology and syntax, and access APIs (Application Programming Interface). HP has nearly universally adopted the data file formats of CDF (*NASA Common Data Format*, 2021), FITS (Ponz et al., 1994), HDF (Folk et al., 1999), and netCDF (Rew & Davis, 1990); and SPASE (King et al. (2010); Roberts et al. (2018)) for metadata. As summarized in the following section, there are many different APIs for obtaining time series-like data stored in these data files. (In this work, we use time series-like to mean data structures that for each time value have an associated scalar, vector, tensor, spectrogram, or higher-dimensional construct.)

The primary design objective for HAPI is to provide the simplest API that allows access to time series data in a streaming form and allows a user not to need knowledge of file system boundaries, directory layouts, and file formats. With this approach, a scientist programming in, for example, IDL, MATLAB, or Python can use a line of code to bring data into a processing or analysis routine from a diverse set of data providers and for a wide array of data – from uniform cadence ground magnetometer measurements to multi-dimensional distribution functions of ions from a spacecraft instrument. This paper provides an overview of the HAPI design and specification, a summary of its API, and a discussion of plans for future development. The API is mature, and the current effort involves broadening the adoption of the standard by data providers and increasing the use of the API and the HAPI software clients by scientists.

1.1 Related Works

As HP projects have primarily converged on CDF as the preferred data file format, the next barrier in interoperability is the diversity of APIs needed to access various HP data centers. Tables 1 lists common access mechanisms and an accounting of the methods used for several well-known data providers listed in Table 2.

Table 1. Common Heliophysics data access methods

Method	Description
A.	A FTP or HTTP directory of files (usually one day of data per file).
B.	A web page with link that indicates when processing is complete, often with the ability to display the data and download it.
C.	A web page with link to an archive of files (zip or tgz) that is available when the processing of a request is complete.
D.	A REST API that returns a data stream or file directly to a user's application.

Table 2. Heliophyiscs data providers, links to their top-level web site, and the data access methods listed in Table 1 that they natively support. An asterisk "*" indicates that data are already available through a HAPI API (*HAPI Server List*, 2021). A plus "+" indicates that HAPI support is under development. Not all of the above data providers that use methods A, B, and C use the same file formats (when files provided); none of the data providers using method D use the same API. However, they all provide data that could be served without loss information through a HAPI server.

Provider	URL	Access Methods
AMDA* (Génot et al., 2021)	http://amda.irap.omp.eu/service/hapi	B, D
CAIO+	https://csa.esac.esa.int/csa/aio/	B, C, D
CARISMA	http://www.carisma.ca/	C
CCMC/iSWA*	https://iswa.gsfc.nasa.gov/	D
CDAWeb*	https://cdaweb.gsfc.nasa.gov/	A, B, C, D
Das2*	http://das2.org/	D
FTEC*	http://hapi.ftecs.com/hapi	A
IMAGE	http://space.fmi.fi/image	C
INTERMAGNET+	http://intermagnet.org/	A, C
ISAS/JAXA	https://darts.isas.jaxa.jp.	A
LiSIRD*	http://lasp.colorado.edu/lisird/	A, D
OMNIWeb*	https://omniweb.gsfc.nasa.gov/	A, B, D
NGDC	https://ngdc.noaa.gov/stp/satellite/	A, C, D
PDS+	https://pds-ppi.igpp.ucla.edu/hapi	A, B
SSCWeb*	https://sscweb.gsfc.nasa.gov/	B, D
SuperMAG	http://supermag.jhuapl.edu/	D
SWPC	https://www.swpc.noaa.gov/	B
MADRIGAL	https://openmadrigal.org/	A, D

1.2 Design Constraints and Associated Motivations

The HAPI specification (*HAPI Project Overview Page*, 2021) was developed by Heliophysics researchers and data scientists as a grass-roots effort to improve interoperability in Heliophysics data. Experience with related efforts such as TSDS (Weigel et al., 2010), a service that mapped data available by methods A to D in Table 1 through a single API, led to the choice of REST (Representational state transfer; Richardson and Ruby (2007)) style services. The REST conventions allow for simple, stateless servers and the streaming of data that naturally hides any file boundaries between data files.

The primary design considerations for the HAPI specification were:

1. The minimum requirements for HAPI compliance should be easy to implement for existing, well-structured data collections.

   Because ease of implementation increases the likelihood of adoption, the HAPI requirements closely resemble the implementations of many existing time series data providers. They do not include any data operations – only delivering existing content; faced with a new specification, many data providers will likely only ever implement the minimum elements needed for compliance.

2. The required metadata is the minimum amount needed to make the data scientifically usable.

   A common way to evaluate the science usability of structured data is whether enough information is present to automatically create a labeled, scientifically comprehensible plot of the data. To reduce the effort required to serve data through a HAPI API, the required HAPI metadata is limited to that needed to do this and allow common content access and usage scenarios rather than search and discovery usecases. However, HAPI metadata can point to other richer metadata or explanatory information about a dataset, and metadata for search and discovery can point to HAPI endpoints and metadata. For example, in Heliophysics, the Space Physics Archive, Search and Extract (SPASE) metadata model (Roberts et al., 2018) has become a standard method for describing data products. (SPASE does not include a specification for the access of data.) In addition to pointing to unstructured external information, HAPI metadata can reference any existing SPASE record, which has structured information such as detailed documentation and caveats. Also, SPASE records are being updated to reference the HAPI-enabled servers indicated in Table 2 as a possible access mechanism for access to data products.

3. Data are accessible at a minimum in a simple CSV-formatted stream

   Although HAPI servers may use high–performance streaming formats that are defined in the specification, all servers must support CSV for output. CSV data is easy to read in all scientific programming languages with only a few lines of code, making it easy for scientists to start working with the data immediately.

   The HAPI streaming format requires servers to create a response stream that aggregates data if it is stored in multiple files. While this introduces complexity, in potential conflict with constraint (1), a simplifying aspect is that servers do not need to have a staging process for file aggregation nor do they need to operate asynchronously or perform user notifications when the result of a request is complete.

   The requirement that HAPI requests and responses occur synchronously adds a practical limitation on the amount of data sent in a single response. The HAPI specification does not dictate any data transfer limits. However, servers may define data volume constraints and report via a HAPI error response that a request was too large to process.

The HAPI specification and server capabilities are related to OPeNDAP (Cornillon et al., 2003) and NcML (Nativi et al., 2005), but the HAPI specification was designed primarily for time-series data, whereas OPeNDAP is primarily used for spatial data, although it can be used for time series. Several of the developers had extensive experience in attempting to adopt OPeNDAP for providing HAPI-like services, but this effort was abandoned in favor of the HAPI specification due to the complications of mapping from OPeNDAP's data model, in which space is a primary index, to HAPI, in which time is the primary index.

1.3 Data and Metadata Formats

All HAPI metadata uses JSON (Javascript Object Notation), which has near-universal support in modern scientific programming languages.

A HAPI server may stream data in CSV and optionally JSON or binary. CSV is highly accessible for scientists and is easily read by all modern programming languages. JSON is also widely supported by many languages and web-based frameworks.

The binary format is a straightforward translation of the CSV output: newlines are removed and integer parameters are written as 64-bit signed integers. Floating-point parameters are written as 64-bit IEEE 754 little-endian floats (*IEEE 754 Standard*, 2021), and string parameters are written as a sequence of 8-bit ASCII characters. HAPI metadata contains the information required to interpret the binary file, including the number of parameters, the dimensionality of each parameter, and the formats of numerical parameters (or length for string parameters).

This custom, simple, and efficient binary format was selected because typical scientific data formats, such as CDF (*NASA Common Data Format*, 2021), HDF (Folk et al., 1999), and netCDF (Rew & Davis, 1990) do not support streaming of data. Experimental code for streaming HDF and netCDF can be found online, but not as integrated parts of the code base for those formats. Protocol buffers (*Protocol Buffers*, 2021) were considered, but library support for this format in the commonly used programming languages (IDL, MATLAB, Python) is poor.

As described in the following section, few scientists should ever need to write parsers and readers for HAPI metadata or data.

1.4 Facilitating Adoption

The HAPI specification is designed to provide a standardized way to provide streaming services that many time-series data providers already offer. The use of the common aspects of existing features of HP data providers means that data providers can, with minimal effort, modify existing streaming servers to make them HAPI compliant.

In addition, several tools have been developed to facilitate adoption.

1. Although the HAPI streaming formats are simple enough that only 10-20 lines of code are needed to read a basic HAPI response, there are error conditions and other optimizations to consider (such as caching and parallelization of requests); a robust and comprehensive client implementation is beyond what should be expected of science users or even software developers who want to create tools that use HAPI data. Mature HAPI server client libraries are available from open-source HAPI projects (*HAPI GitHub Project Page*, 2021) for Java, IDL, MATLAB, and Python, so most users can begin with these libraries. These clients read a HAPI response into a data structure appropriate for the given language (e.g, in Python the response is read into a NumPy N-D array with timestamps converted to Python `datetime` objects). Work is underway to incorporate HAPI readers into existing popular anal-

ysis frameworks, such as SPEDAS in IDL and Python (*SPEDAS Space Physics Environment Data Analysis Software*, 2021), Autoplot in Java (Faden et al., 2010), SunPy in Python (Mumford et al., 2015), and SpacePy in Python (Morley et al., 2010).

2. To assist server developers with adoption, a cross-platform reference server has been developed. With this server, a data provider only needs to provide HAPI JSON metadata and a command-line program that emits HAPI CSV given inputs of a dataset identifier, and start/stop times (*HAPI Project Overview Page*, 2021). The server handles HAPI error responses, request validation, and logging.

3. A validator/verifier has been developed that runs many tests on a server (*HAPI Server Verifier*, 2021). The server executes a comprehensive suite of tests for compliance with the specification. By either downloading and running the test software locally, or by entering the URL of a HAPI server into a website, data providers can test most aspects of their HAPI implementation, with detailed results indicating warnings or errors. This has proven to be a very effective way to assist developers, and it helps ensure that new HAPI servers are fully functional. The tests include checks of JSON responses against the HAPI metadata schema, verifying that the server handles different allowed start/stop time representations (e.g, year, day-of-year and year, month day with varying amounts of additional time precision), error responses and message, timeouts, and testing that output is independent of the output format. Also, the validator/verifier makes recommendations – for example, if a server does not include cross-origin request sharing headers or respond with compressed data when gzip is specified in the Accept-Encoding request header, it emits a message noting their advantages and a link to information on how to enable or implement these features.

2 API

The five required endpoints that constitute the HAPI API are summarized in this section. In addition to defining the responses to requests to these URLs, the HAPI specification includes requirements for the format and structure of the response. These additional requirements are summarized in the description of the relevant endpoint.

2.1 `/about` endpoint

The response to this endpoint is a JSON object containing the HAPI API version implemented, the status of the response, the `id` of the server, the server `title` (a brief description of the provided data), and contact information for the server administrator. An example response is

```
{
  "HAPI": "3.0",
  "status": {"code": 1200, "message": "OK"},
  "id": "SSCWeb",
  "title": "SSCWeb Data and Metadata",
  "contact": "example@example.org"
}
```

This endpoint has no request parameters.

2.2 `/capabilities` endpoint

The response to this endpoint is a JSON object containing the HAPI API version implemented, the status of the response, and output formats supported. A client can use this endpoint to determine if the HAPI server provides additional data output formats beyond the required CSV. An example response is

```
{
  "HAPI": "3.0",
  "status": {"code": 1200, "message": "OK"},
  "outputFormats": ["csv", "binary", "json"]
}
```

Note that the minimum requirement is that `outputFormats=["csv"]`. This endpoint has no request parameters.

2.3 `/catalog` endpoint

The response to a request to this endpoint is a list of the available datasets from the HAPI server. An example is

```
{
    "HAPI": "3.0",
    "status": {"code": 1200, "message": "OK"},
    "catalog":
        [
            {"id": "dataset1", title: "dataset 1 title"},
            {"id": "dataset2"}
        ]
}
```

The minimal requirement is that the objects in the `catalog` array contain an `id`; the `title` is optional. This endpoint has no request parameters.

2.4 `/info` endpoint

This endpoint has one required request parameter, `dataset`, which corresponds to an identifier of the dataset that appears in the `/catalog` response. The response to this endpoint contains information about the parameters in a dataset. It contains the metadata needed to satisfy requirement (2) in section 1.2 that HAPI metadata should be the minimum required for a program reading the data to produce a plot with labels needed for scientific interpretation.

For example, a response to `/info?dataset=ACE_MAG` could be

```
{
"HAPI": "3.0",
"status": {"code": 1200, "message": "OK"},
"startDate": "1998-001Z",
"stopDate" : "2017-100Z",
"parameters":
[{
      "name": "Time",
      "type": "isotime",
      "units": "UTC",
      "fill": null,
      "length": 24
},
     ...
{
              "name": "mag_GSE",
              "type": "double",
              "units": "nT",
              "fill": "-1e31",
              "size" : [3],
              "description": "hourly ave magnetic field in GSE",
              "label": "B field in GSE"
         }]
}
```

In this example, the size element of the `mag_GSE` parameter indicates the dimensionality of the parameter. `size=[3]` means that this parameter will occupy three columns in the CSV data response. Higher-dimensional parameters are allowed. For example, a dataset parameter consisting of the energy of protons in one of 6 energy ranges striking a sensor at 12 different pitch angles would have `size=[12, 6]` and would occupy 72 columns in the CSV response.

2.5 `/data` endpoint

This endpoint has required request parameters of a dataset identifier and time range, and the response is a CSV stream of all parameters in the dataset. Optionally, the request may include a comma-separated subset of parameters to return. A sample response for the request

```
/data?dataset=ACE_MAG&parameters=mag_GSE&time.min=2016-01-01Z&time.max=2016-01-02Z
```

is

```
2016-01-01T00:00:00.000Z,0.05,0.08,-50.98
...
2016-01-01T23:59:59.000Z,0.09,0.03,-30.0
```

The HAPI binary representation of this file is 86400 blocks, each containing 24 ASCII-encoded bytes (representing a timestamp) followed by 3 IEEE 754 little-endian 64-bit floats.

3 Summary and Future Development

The HAPI specification captures a diverse range of time series representations used in Heliophysics, including scalars, vectors, spectra, and multi-dimensional time series commonly made by energetic particle instruments. The specification also captures many of the features of existing APIs, simplifying the transition from a non-standards-based API to a HAPI API.

In addition to developing an API, as discussed in 1.4, significant effort was made to facilitate adoption. For example, for server developers, a generic server is available – a data provider needs only to provide `/catalog` and `/info` metadata JSON files and executable that takes inputs of the dataset identifier and start/stop times and returns HAPI CSV. Usually, such programs are fewer 50 lines. For users, software has been developed for commonly used programming languages in Heliophysics, including IDL, Java, MATLAB, and Python.

The type of data described by HAPI metadata and served according to its specification is easily extended to other science domains such as the Earth Sciences and is an area of active research.

References

Cornillon, P., Gallagher, J., & Sgouros, T. (2003). OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal*, *2*, 164–174.

Faden, J. B., Weigel, R. S., Merka, J., & Friedel, R. H. (2010). Autoplot: a browser for scientific data on the web. *Earth Science Informatics*, *3*(1), 41–49.

Folk, M., McGrath, R., & Yeager, N. (1999). HDF: an update and future directions. In *International Geoscience and Remote Sensing Symposium.* IEEE. doi: 10.1109/igarss.1999.773469

Génot, V., Budnik, E., Jacquey, C., Bouchemit, M., Renard, B., Dufourg, N., … Cabrolie, F. (2021, July). Automated multi-dataset analysis (AMDA): An on-line database and analysis tool for heliospheric and planetary plasma data. *Planetary and Space Science*, *201*, 105214. doi: 10.1016/j.pss.2021.105214

HAPI Github Project Page. (2021). Retrieved from `https://github.com/hapi-server/`

HAPI Project Overview Page. (2021). Retrieved from `http://hapi-server.org/`

HAPI Server List. (2021). Retrieved from `http://hapi-server/servers/`

HAPI Server Verifier. (2021). Retrieved from `http://hapi-server.org/verify`

Heliophysics Data Environment. (2021). Retrieved from `https://hpde.gsfc.nasa.gov/`

IEEE 754 Standard. (2021). Retrieved from `https://standards.ieee.org/standard/754-2019.html`

King, T., Thieman, J., & Roberts, D. A. (2010, May). SPASE 2.0: a standard data model for Space physics. *Earth Science Informatics*, *3*(1-2), 67–73. doi: 10.1007/s12145-010-0053-4

Morley, S. K., Welling, D. T., Koller, J., Larsen, B. A., & Henderson, M. G. (2010). *Spacepy-a python-based library of tools for the space sciences* (Tech. Rep.). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Mumford, S. J., Christe, S., Pérez-Suárez, D., Ireland, J., Shih, A. Y., Inglis, A. R., … others (2015). Sunpy—python for solar physics. *Computational Science & Discovery*, *8*(1), 014009.

NASA Common Data Format. (2021). Retrieved from `https://cdf.gsfc.nasa.gov/`

Nativi, S., Caron, J., Davis, E., & Domenico, B. (2005, November). Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML). *Computers & Geosciences*, *31*(9), 1104–1118. Retrieved from `https://doi.org/10.1016/j.cageo.2004.12.006` doi: 10.1016/j.cageo.2004.12.006

Ponz, J., Thompson, R., & Munoz, J. (1994). The FITS image extension. *Astronomy and Astrophysics Supplement Series*, *105*, 53–55.

Protocol Buffers. (2021). Retrieved from `https://developers.google.com/protocol-buffers`

Rew, R., & Davis, G. (1990, July). NetCDF: an interface for scientific data access. *IEEE Computer Graphics and Applications*, *10*(4), 76–82. doi: 10.1109/38.56302

Richardson, L., & Ruby, S. (2007). *RESTful web services.* O'Reilly.

Roberts, D. A., Thieman, J., Génot, V., King, T., Gangloff, M., Perry, C., … Hess, S. (2018, December). The SPASE Data Model: A metadata standard for registering, finding, accessing, and using Heliophysics data obtained from observations and modeling. *Space Weather*, *16*(12), 1899–1911. doi: 10.1029/2018sw002038

SPEDAS Space Physics Environment Data Analysis Software. (2021). Retrieved from `https://spedas.org/`

Weigel, R. S., Lindholm, D. M., Wilson, A., & Faden, J. (2010, June). TSDS: high-performance merge, subset, and filter software for time series-like data. *Earth Science Informatics*, *3*(1-2), 29–40. doi: 10.1007/s12145-010-0059-y