HAPI Developments and a New Java Framework for HAPI Servers and Its Use at ESAC

Jeremy Faden, dba Cottage Systems, University of Iowa
Jon Vandegriff, Applied Physics Laboratory
Larry Brown, Applied Physics Laboratory
Anastasia Andres, ESAC
Hector Perez, ESAC

What I'll talk about

- the HAPI data transport protocol
- new features for HAPI this past year
- introduce a new Java framework for creating servers

The HAPI Protocol

- Many web sites have servers which send data. Each has its own protocol, and new client software is needed for each one.
- This means we're always coding, and writing code that works just well enough to get the job done.
- This also means code is buggy, and the service is only useful to those who can code. Interactions with the server staff are often required as well.
- The HAPI project started in 2016, where we hoped to create a single, simple API which time-series data providers could implement.
- We would also provide clients for common programming languages.

The HAPI Protocol

- HAPI (Heliophysics API) is a data discovery and access protocol.
- data is discovered using "catalog" and "info" requests:
  - https://vires.services/hapi/catalog → list of dataset ids
  - https://vires.services/hapi/info?id=CS_OPER_MAG → description of dataset and its parameters
- data is accessed using "data" requests:
  - https://vires.services/hapi/data?id=CS_OPER_MAG
    &start=2022-03-30T20:00Z
    &stop=2022-03-31T23:59Z
    &parameters=Timestamp,Latitude,Longitude,F → data file
- HAPI servers themselves are discovered using a registry at https://github.com/hapi-server/servers/blob/master/all.txt

The HAPI Protocol

- requests are REST-style queries
- responses are JSON and CSV (and optionally Binary)
- data responses are streamed to client,
  so that data can be transmitted as soon as it is read,
  and processed as it is received.
- JSON schemas on the responses constrain
  what is allowed, and x_* tags can be used to
  insert arbitrary metadata.

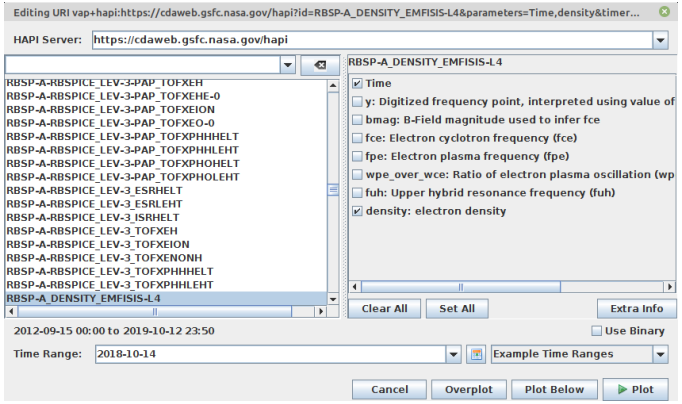
vires.services/hapi/info?id=CS_OPER_MAG

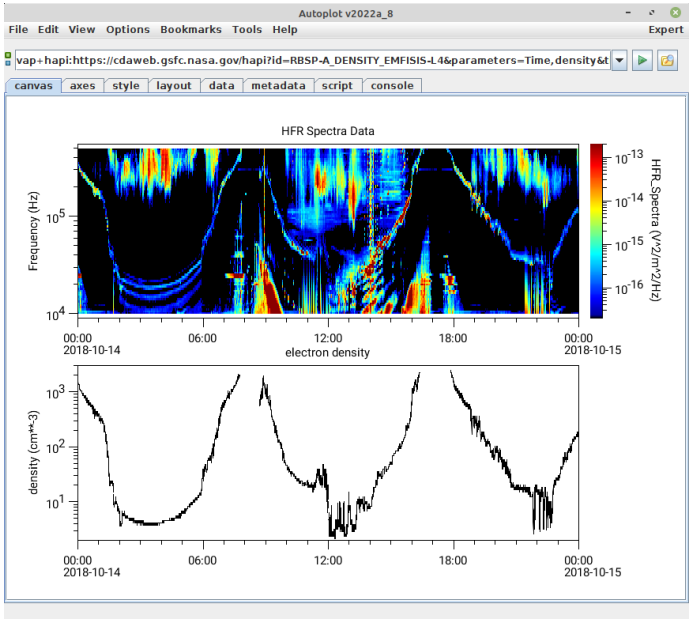
Apps timekeeping ihdea/agu fun ct project autoplot


```
{
  "HAPI": "3.0",
  "status": {
    "code": 1200,
    "message": "OK"
  },
  "x_dataset": "CS_OPER_MAG",
  "x_datasetType": "CS_MAG",
  "startDate": "2010-04-08T17:03:26.254008Z",
  "stopDate": "2022-03-31T23:59:52.266008Z",
  "cadence": "PT4S",
  "x_maxTimeSelection": "P20D",
  "modificationDate": "2022-05-19T09:21:03.928Z",
  "description": "Cryosat-2 Platform magnetometer measurements",
  "x_dataTerms": "This dataset is provided by the European Space Ag
  "resourceURL": "https://doi.org/10.1186/s40623-020-01171-9",
  "parameters": [
    {
      "name": "Timestamp",
      "description": "Time stamp",
      "units": "UTC",
      "type": "isotime",
      "length": 24,
      "x_standard": "UTC",
      "x_epoch": "1970-01-01T00:00:00Z",
      "x_unit": "ms",
      "x_type": "int64",
      "fill": null
    },
```

The HAPI Protocol

- many clients exist already
  - Java
  - IDL
  - Matlab
  - Python
  - JavaScript
- use in applications
  - Autoplot
  - SPEDAS
  - LASP Space Weather Portal
  - ViRES server
  - Python client used in ad-hoc analyses
  - wget/curl ("read my data" feature)

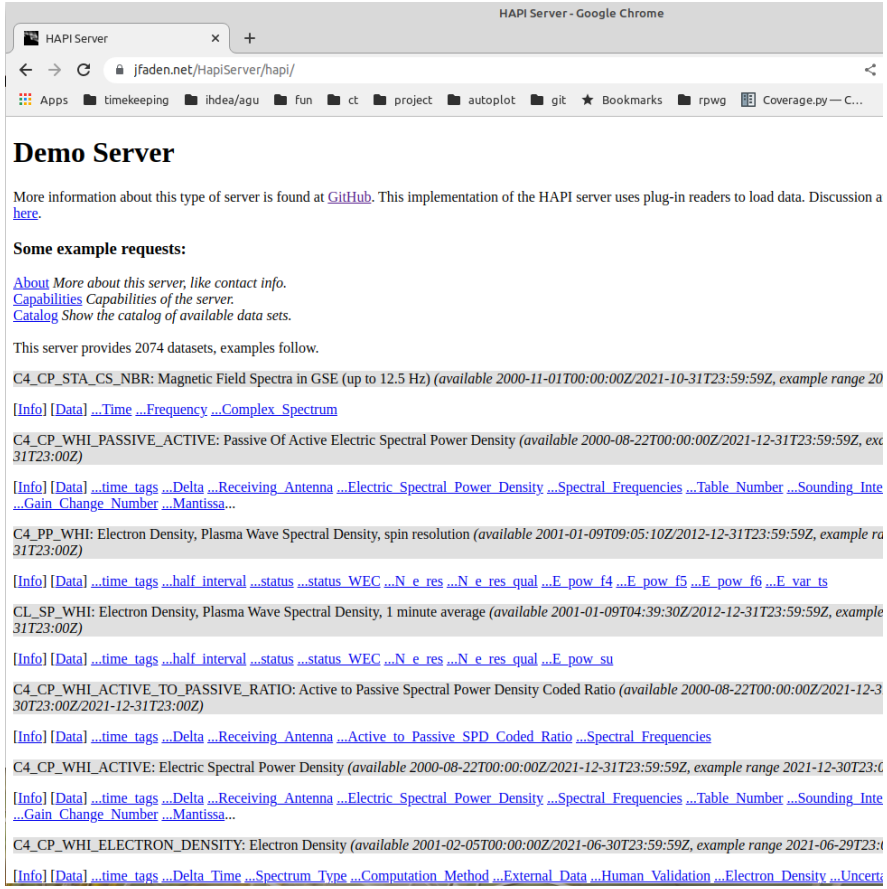




A Python script that uses a HAPI server works anywhere, unlike with data from downloaded files. This makes communication with collaborators easier.

The HAPI Protocol

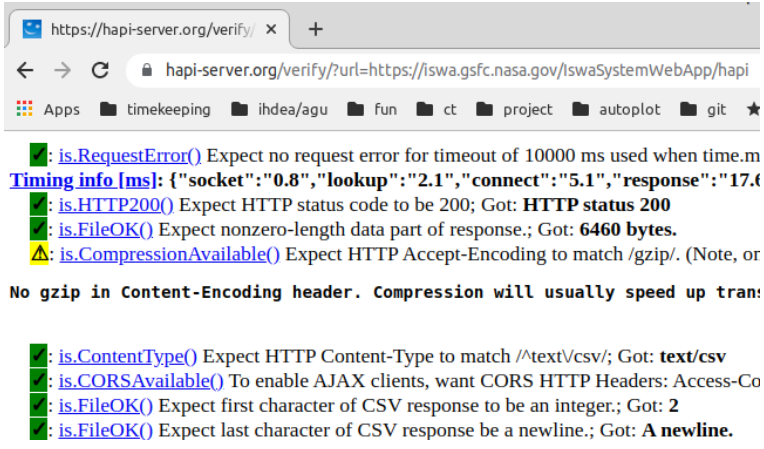
- many server codes exist already
  - Python
  - Node JavaScript
  - Scala
  - Java - several independently-written codes
- implementations include
  - CDAWeb https://cdaweb.gsfc.nasa.gov/hapi
  - SSCWeb http://hapi-server.org/servers/SSCWeb/hapi
  - University of Iowa http://planet.physics.uiowa.edu/das/das2Server/hapi
  - LASP http://lasp.colorado.edu/lisird/hapi
  - AMDA http://amda.irap.omp.eu/service/hapi
  - VirES https://vires.services/hapi
  - PDS-PPI https://pds-ppi.igpp.ucla.edu/hapi/
- Note that many of these simply convert an existing service to HAPI.



The HAPI Protocol

HAPI is a brand-name we want the scientists to recognize and know what it is, and for it to mean that there's a reliable service being offered.

- Validation
  - testing service which validates a server (https://hapi-server.org/verify/)
  - removes human biases gives a reference for what "HAPI" means
- Testing
  - hourly testing of all known servers ensures reliability (https://jfaden.net/jenkins/job/test-hapi-servers-2022/)
- Indexing
  - indexing of known servers is done weekly, capturing data sets offered and descriptions
  - https://github.com/hapi-server/servers/tree/master/index



New Features this Year

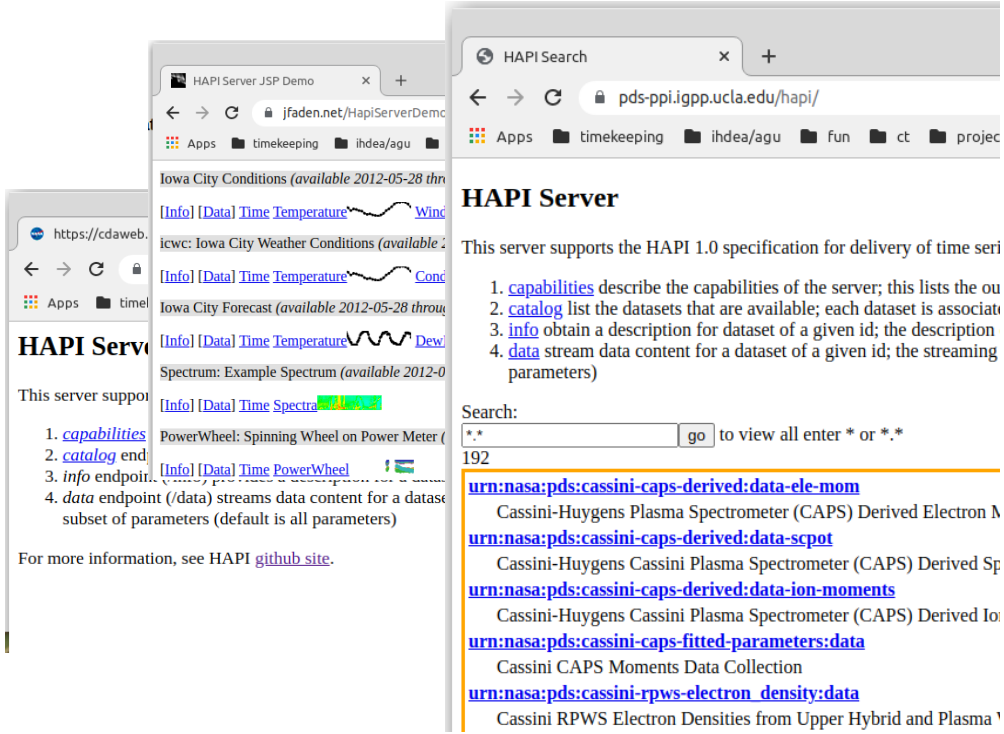
- Version 3.1 is about to be released
- trivial things like time.min→start, time.max→stop
  (Note this changes the API, version 4.0 will drop them)
- firm up the spec on IDs (recommendation that IDs should be a-z, A-Z, 0-9, [-._~] and up to 30 characters; but this is not required. Any Unicode characters are allowed, but will require URL escaping)
- maxRequestDuration allows servers to tell clients the size of a reasonable request, presumably "request too big" would be returned.
- units and units schemas, so that unit strings ("1/cm**3" vs "cm^-3" etc) are machine-ready.
- identifying coordinate frames in a machine-ready way. Vector quantities x,y,z or in theta,phi,radius or ...
- arbitrary metadata block of non-HAPI metadata (for example, all CDF metadata) with machine-ready schema identifier. For example, Autoplot understands CDF metadata, so now it can recognize and use it.
- and a new logo!

Help is available

- Help is available for teams wanting to set up servers.
- We have weekly Zoom meetings to discuss the technical details. ("Should ticket 141 be resolved for release 4.0?")
- Monthly Zoom community meetings to showcase new servers and to understand what's needed outside our group. ("Here's my new server, but my vector components don't have labels in Autoplot, why?")
- This is a very loose federation of institutions with common interests, and new people and ideas are welcome!

Server-Java

- Many Java implementations of HAPI servers exist already:
  - I have one which I use with Autoplot to test new HAPI features
  - CDAWeb
  - PDS-PPI planetary node at UCLA
- This confirms that the servers are easy enough to implement, but...



Server-Java

- I'd argue each is under-implemented
  - do bugs exist?
  - does it match performance goals of HAPI brand (response within 300ms, etc)?
  - does it provide all HAPI features (for example Binary as well as CSV)?
  - I have yet to see a server which completely validates with no warnings.
- There should be one common Java code for HAPI servers, and specific site's developer should only worry about the code to make their data available to the server.

Server-Java

- Server-Java is intended to be a framework with which new servers (specific-servers) are implemented.
- data providers will use its source code with their specific-server code to interface site data.
- it is configured to call into their specific-server code appropriately.
- Server-Java cleans up and validates responses from specific-server code, speeding up implementation.

Server-Java — configuration

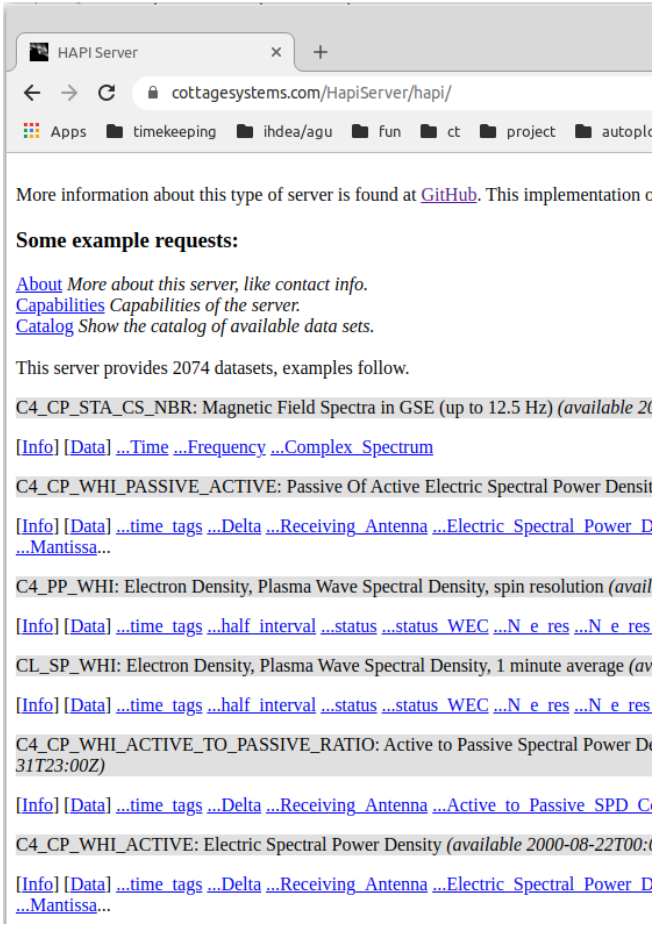
- For each data set, a configuration controls
  - identifies the "data reader", for example it should:
    - call into Java code (a special reader or adapter to existing API),
    - or execute a Unix command and read the output,
    - or read pre-formatted files,
    - or ... (we can extend the code as needed).
  - but also tells Server-Java how to call the data reader. Should it:
    - load all data set's parameters at once, or just load a subset of the parameters?
    - load all records in the requested interval, or break it up into granules of regular cadence (i.e. $Y-$m-$d for daily granules) and Server-Java will combine them? This allows a non-streaming legacy server to feed data into the streaming HAPI server.

Server-Java — configuration

- This means a data reader can be as simple as executing "cat my_formatted_file.csv" on Unix,
- and as complex as calling into a reader which opens each CDF file and extracts just the needed data.
- Servers are configured using .json files, which can contain the pre-formatted info responses, or they can specify a code to call which will output the info and catalog responses.

Use Case at ESAC

- Larry visited ESAC and set up a server with their service staff.
- The server-specific software interfaces with the existing TAP server.
- He wrote Java code which knows how to request data from the TAP server and provide it to server-java.
- The TAP responses are streaming, so server-specific code calls it once per data request. The TAP server responds with all parameters for a data set.
- The Server-Java code will trim the whole-day response to precisely the time range requested, and extract only the parameters which were requested by the scientist.
- The ESAC server should be available to the public soon.



What's Coming

- servers ought to be responsive, so caching will enable servers to cache responses. This will reduce server load at the expense of storage.
- right now the configuration is done with JSON and familiarity with HAPI is needed to form JSON HAPI responses, and a tool for generating responses is needed (interview form).
- we plan to implement new CDAWeb and SSCWeb servers using this framework in the next year.

Thanks!

https://hapi-server.org

https://github.com/hapi-server

https://github.com/hapi-server/data-specification (documentation for specification)

https://github.com/hapi-server/server-java (server-java code)

https://github.com/hapi-server/server-java/wiki (server-java documentation)

https://cottagesystems.com/HapiServer/hapi/ (temporary home for ESAC server, will change)

Jeremy Faden

faden@cottagesystems.com

Server-Java

actors and nouns

- data reader is the code which provides data
- parameter is an individual time series (Density or GSMPosition or FluxSpectrum)
- data id identifies a group of parameters with common time tags. (OMNI_HRO_5MIN)
- data request is a request for a set of parameters for a block of time.
- info response is the JSON response describing the parameters available and time span of a data id.
- data set is the data for a group of parameters for a block of time.
- server home directory is the directory containing configurations and cached data.
- json reader configs are the files which configure the server.
- server-java is the core part which is common to all server implementations.
- specific-server is a particular implementation, like ESAC or CDAWeb.
- scientist is the person receiving data.
- server developer is the person/people implementing the server.
- service staff are the people maintaining and adding data to the server.
- site data is the data which will be served to the public.