Feedback on the implementation of HAPI in CDPP/AMDA

Benjamin Renard, Vincent Génot and all CDPP team

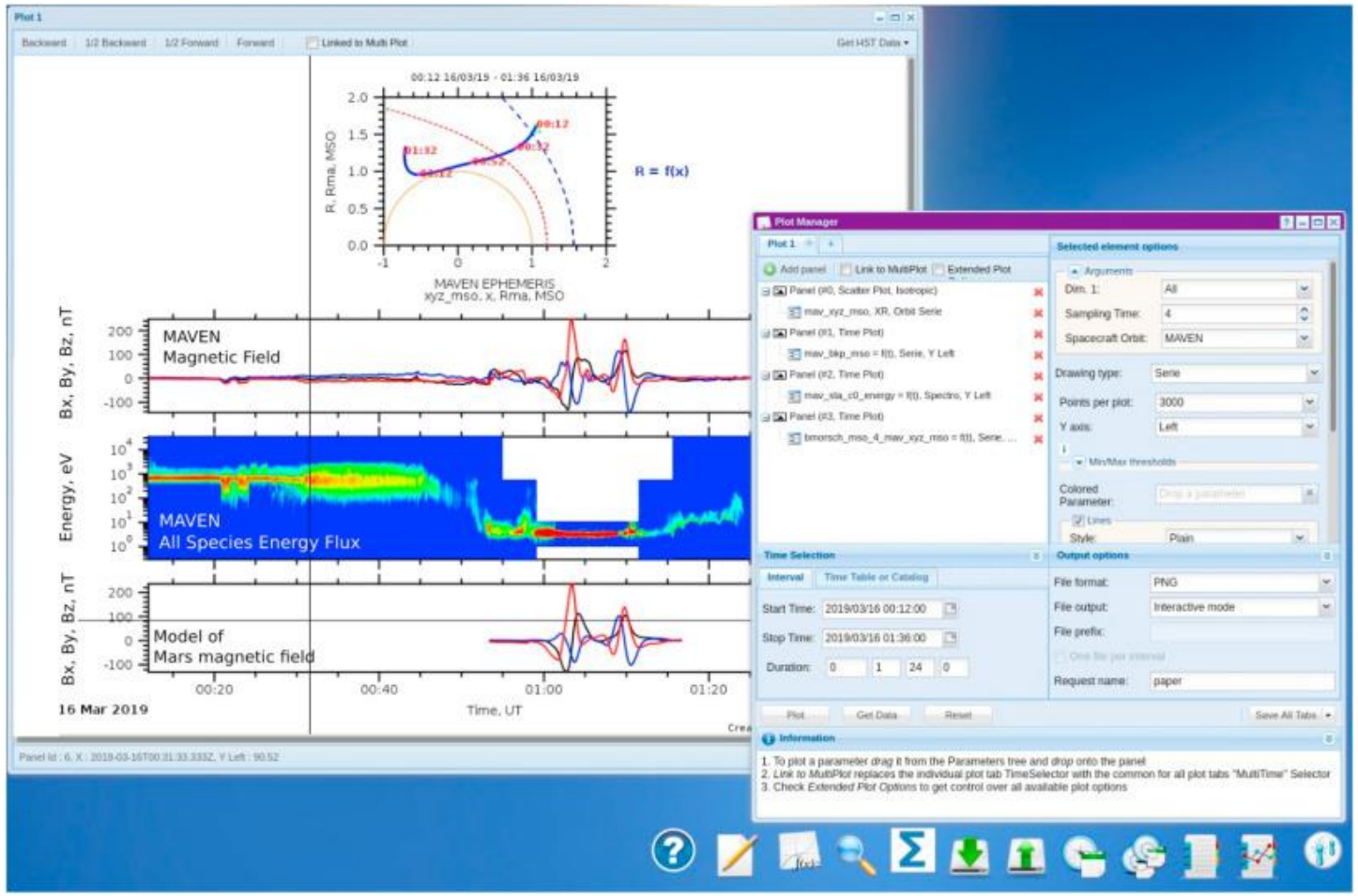
CDPP/AMDA overview – 1/3

- CDPP : Centre de Données de la Physique des Plasmas :
  - French national data centre for natural plasmas of the solar system
  - http://www.cdpp.eu
- AMDA : Automated Multi-Dataset Analysis :
  - Web-based facility : http://amda.cdpp.eu
  - Distributes space physics data for 15 years:
    - Wide variety of space physics data (heliosphere, magnetospheres, planetary environments) in a homogeneous way
    - More than 800 datasets
    - Public data from NASA/CDAWeb, NASA/PDS, ESA/PSA, ESA/CSA, ESA/SOAR, … + simulations and models
    - Data distribution center for plasmas data of Rosetta mission
  - More than 500 unique connections per month

CDPP/AMDA overview – 2/3

AMDA key features :

- Plot data
- Constructing derived parameter from a mathematical expression
- Data mining
- Statistics calculation on data
- Operations on TimeTables and catalogues
- Upload and download data
- Machine Learning predictions (coming soon)



More information about CDPP/AMDA (with some use cases) : https://doi.org/10.1016/j.pss.2021.105214

Automated Multi-Dataset Analysis (AMDA): An on-line database and analysis tool for heliospheric and planetary plasma data, Planetary and Space Science – 2021 - V. Génot et al.

CDPP/AMDA overview – 3/3

The CDPP is an actor of the interoperability:

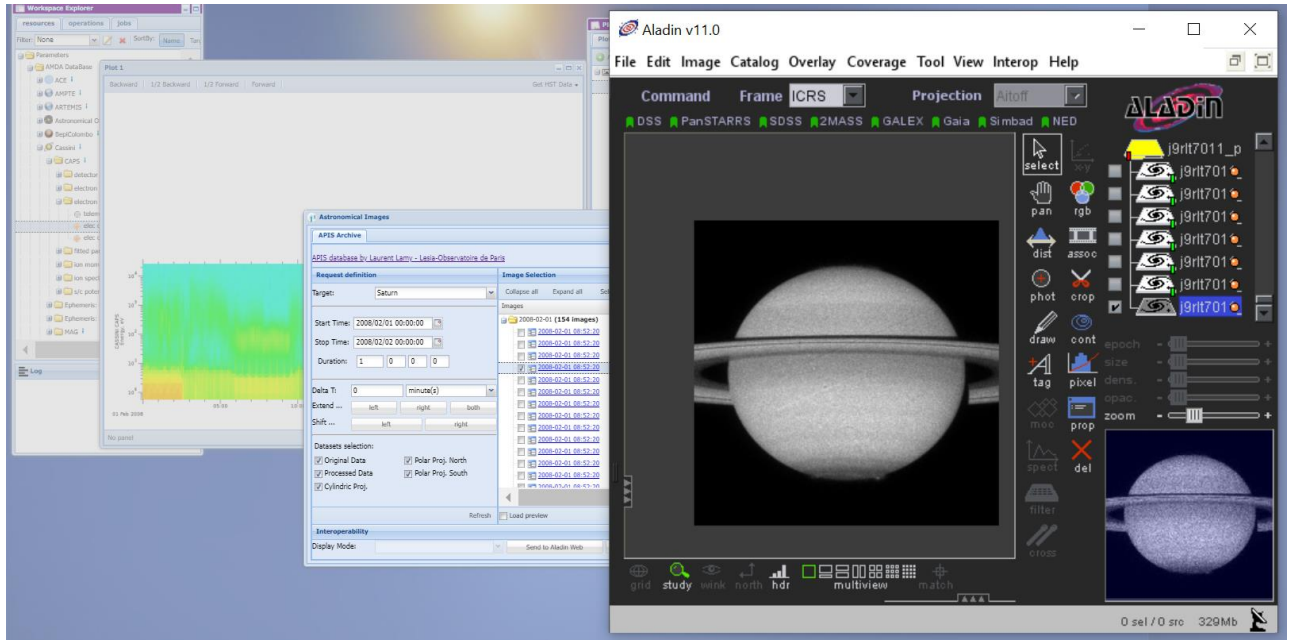
- Join SPASE group very quickly after if was created (1998)
- Participate to the definition of the “Simulation Extensions” for the SPASE data model

In AMDA:

- Use SPASE data model for the proper description of data
- “HPEvent” format to export catalogues
- Implements an EPN-TAP server
- IVOA/SAMP integration
- **And, of course, implements a HAPI server**

We also have a new Python client « speasy » to access AMDA data:

- https://pypi.org/project/speasy/
- Developped with our colleagues of the LPP (A. Jeandet et N. Aunai)



HAPI Server Front-End – Server-side software

- Use official HAPI NodeJS server : https://github.com/hapi-server/server-nodejs
- ⚠ Version 0.9.5 (last release: Version 1.0.11) ⚠
- Support HAPI 2.0 (last stable release: Version 3.0.0)

The intended use for this server-side software is a data provider wants to serve data through a HAPI API. With this software, the data provider only needs

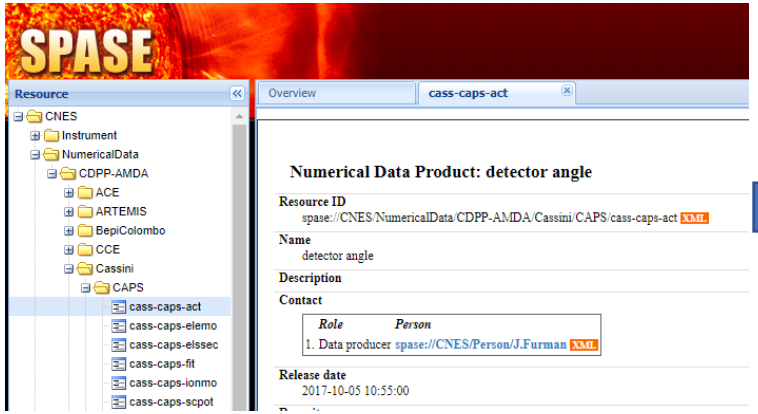
1. HAPI metadata, in one of a variety of forms, for a collection of datasets and
2. a command-line program that returns at least headerless HAPI CSV for all parameters in the dataset over the full time range of available data. Optionally, the command line program can take inputs of a start and stop time, a list of one or more parameters to output, and an output format

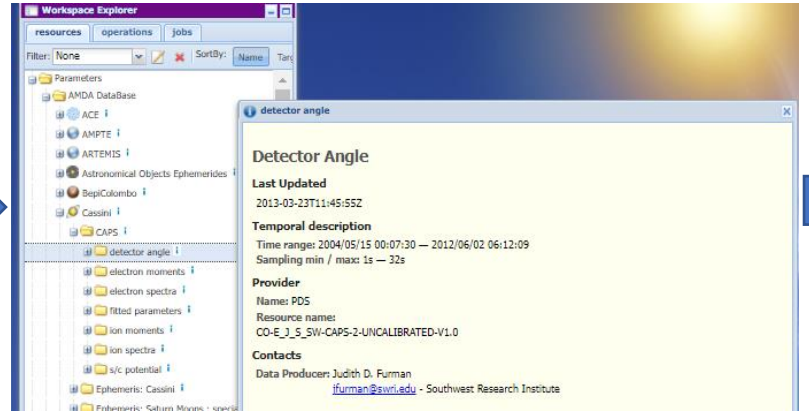
to be able to serve data from a HAPI API from their server. This software handles

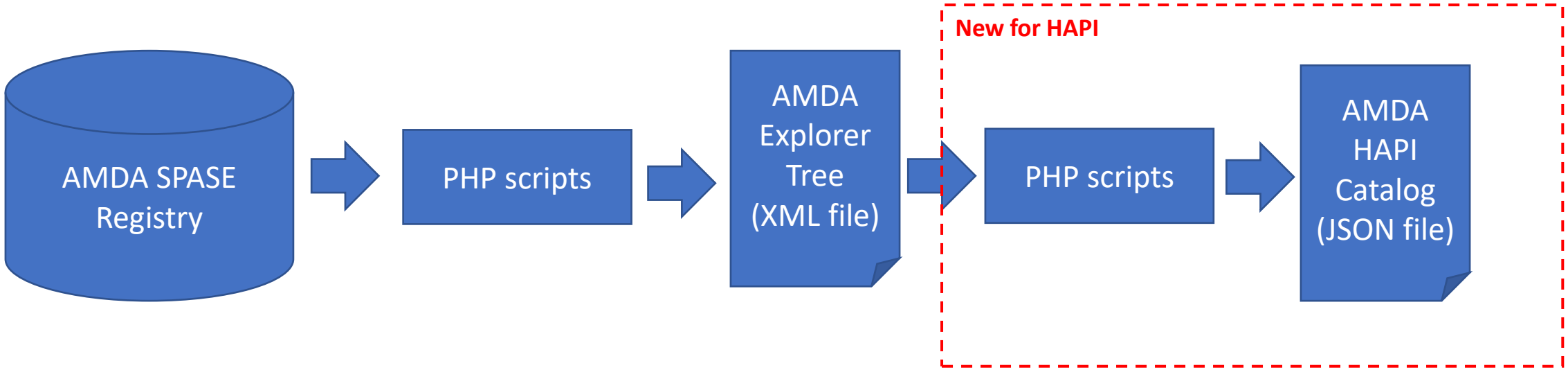
1. HAPI metadata validation,
2. request validation and error responses,
3. logging and alerts,
4. time and parameter subsetting (as needed), and
5. generation of HAPI JSON or HAPI binary (as needed).

A list of catalogs that are served using this software is given at http://hapi-server.org/servers.

HAPI Server Front-End – Metadata generation 1/2







HAPI Server Front-End – Metadata generation 2/2

Restrictions / Limitations :

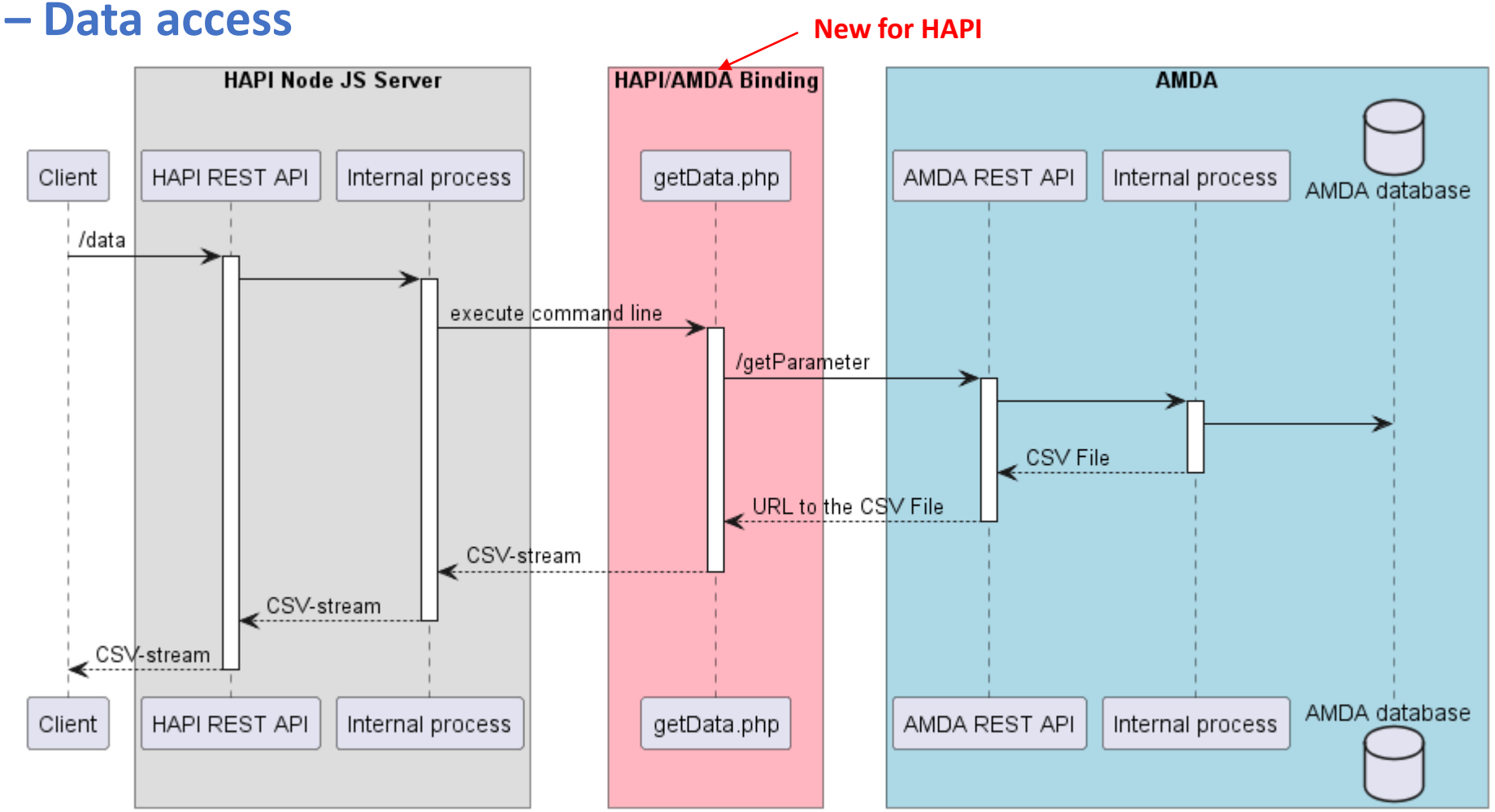
- All parameters (except “Time”) are considered as “double”
- “Mission dependent” parameters cannot be added
- Datasets with an access restriction cannot be added
- Variable bins not supported (HAPI 2.0)

⚠ Our AMDA SPASE registry is currently <u>not public</u> :

- Not fully compliant with the SPASE Data Model ☹:
  - Some additional tags for internal purpose
  - Some missing tags required by the specification
- Work in progress to be able to publish our registry in hpde.io (coming soon in 2022)


```
{
  "id": "cass-caps-elssec",
  "title": "electron spectra",
  "info": {
    "startDate": "2004-05-15T00:07:30Z",
    "stopDate": "2012-06-02T06:12:09Z",
    "cadence": "PT4S",
    "description": "electron spectra : incalibrated : 8 anodes<br/> MinSampling: 4S; MaxSampling: 32S<br/> Provider: PDS",
    "resourceID": "spase://CNES/NumericalData/CDPP-AMDA/Cassini/CAPS/cass-caps-elssec",
    "parameters": [
      {
        "name": "Time",
        "type": "isotime",
        "length": 24,
        "units": "UTC",
        "fill": null
      },
      {
        "name": "cass_caps_elssectel",
        "type": "double",
        "units": "",
        "fill": "-1e31"
      },
      {
        "name": "cass_caps_els",
        "type": "double",
        "size": [
          63
        ],
        "bins": [
          {
            "name": "Energy",
            "units": "eV",
            "ranges": [
              [
                24133,
                26040
              ],
              [
                20609,
                24133
              ],
                [...]
              [
                0.58,
                0.87
              ]
            ]
          }
        ],
        "units": "cnts.s-1",
        "fill": "-1e31",
        "description": "energy spectrogram of electron count rates"
      },
      {
        "name": "cass_caps_els_sum",
        "type": "double",
        "size": [
          63
        ],
      [...]
```

HAPI server – Data access



To serve data, we have just to implement a binding (« getData.php » script) between HAPI NodeJS Server and the AMDA REST WebServices:

- http://amda.irap.omp.eu/help/apidoc/
- Initially developed during the “EU/FP7 IMPEx” project, and continue to be improved

HAPI server – Configuration and integration

1. Server configuration

⚠ Version 0.9.5 ⚠

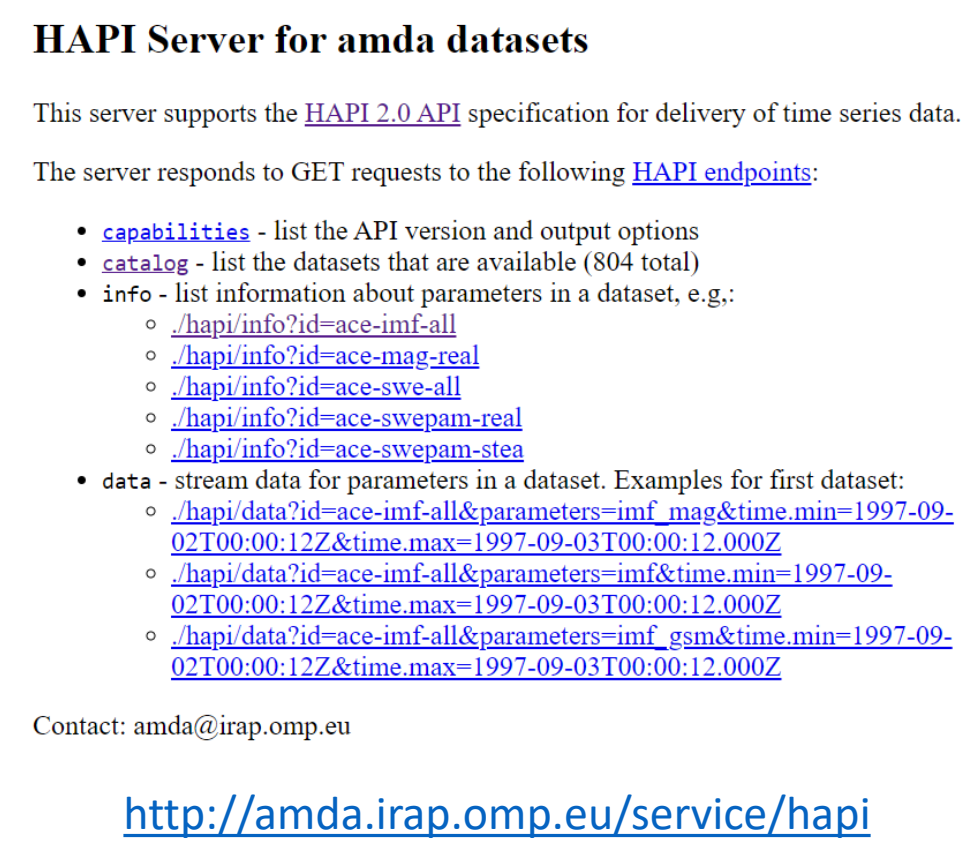
```
{
        "data": {
                "command": "php ../php/hapi/getData.php --id ${id} --parameters \"${parameters}\" --start ${start} --stop ${stop}",
                "contact": "amda@irap.omp.eu"
        },
        "catalog" : "../generic_data/HAPI/metadata/amda-catalog.json"
}
```

2. Server installation

- Install NodeJS
- Uncompress hapi-server release
- Add « ProxyPass » definition in Apache configuration file
- Restart Apache

3. Update metadata

- We already have a script to automatically update AMDA local tree
- We just add a post-process:
  - To automatically call the script to generate HAPI metadata
  - To restart HAPI server for catalog reloading



HAPI Server for amda datasets

This server supports the HAPI 2.0 API specification for delivery of time series data.

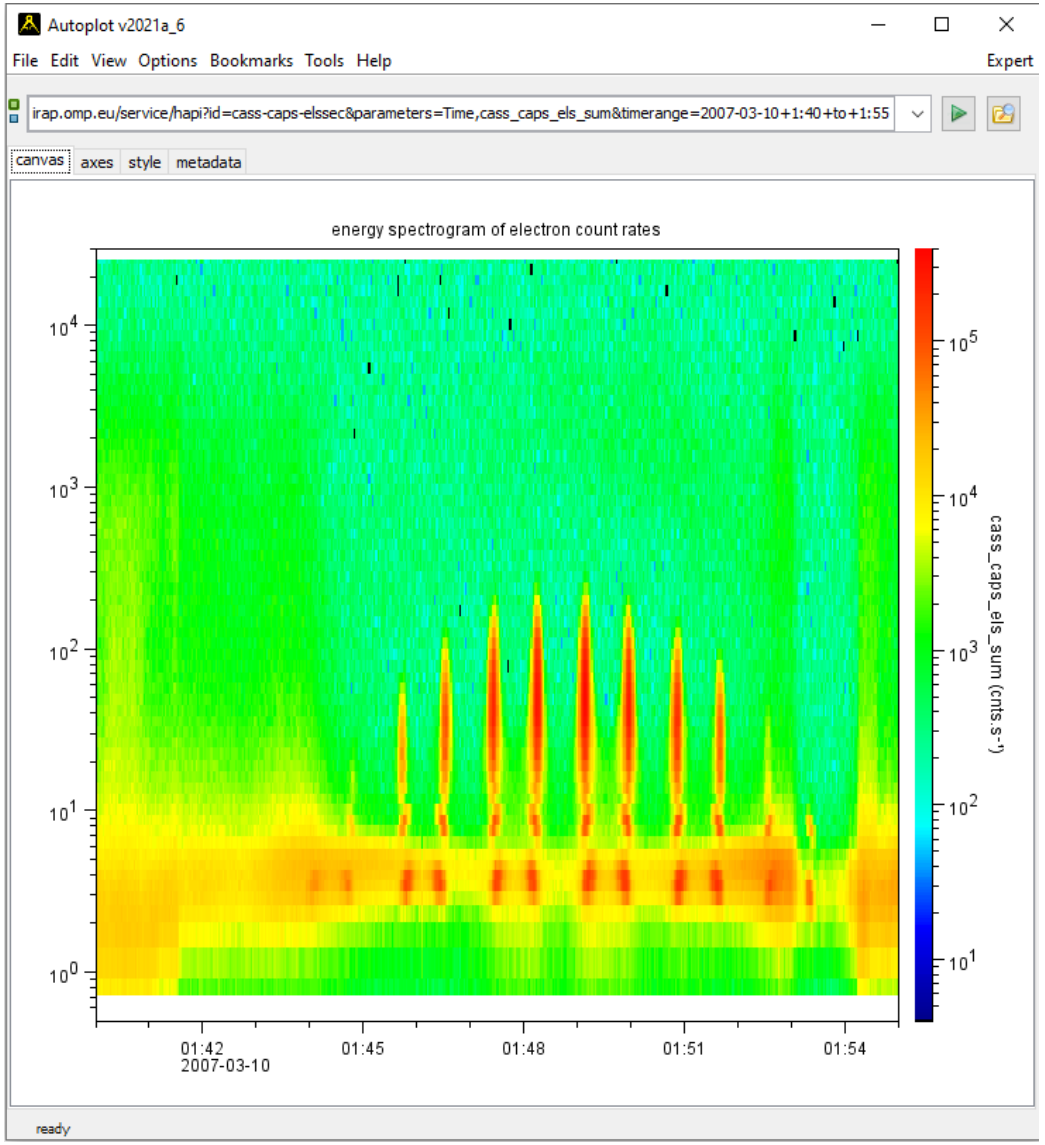
The server responds to GET requests to the following HAPI endpoints:

- `capabilities` - list the API version and output options
- `catalog` - list the datasets that are available (804 total)
- `info` - list information about parameters in a dataset, e.g,:
  - ./hapi/info?id=ace-imf-all
  - ./hapi/info?id=ace-mag-real
  - ./hapi/info?id=ace-swe-all
  - ./hapi/info?id=ace-swepam-real
  - ./hapi/info?id=ace-swepam-stea
- `data` - stream data for parameters in a dataset. Examples for first dataset:
  - ./hapi/data?id=ace-imf-all&parameters=imf_mag&time.min=1997-09-02T00:00:12Z&time.max=1997-09-03T00:00:12.000Z
  - ./hapi/data?id=ace-imf-all&parameters=imf&time.min=1997-09-02T00:00:12Z&time.max=1997-09-03T00:00:12.000Z
  - ./hapi/data?id=ace-imf-all&parameters=imf_gsm&time.min=1997-09-02T00:00:12Z&time.max=1997-09-03T00:00:12.000Z

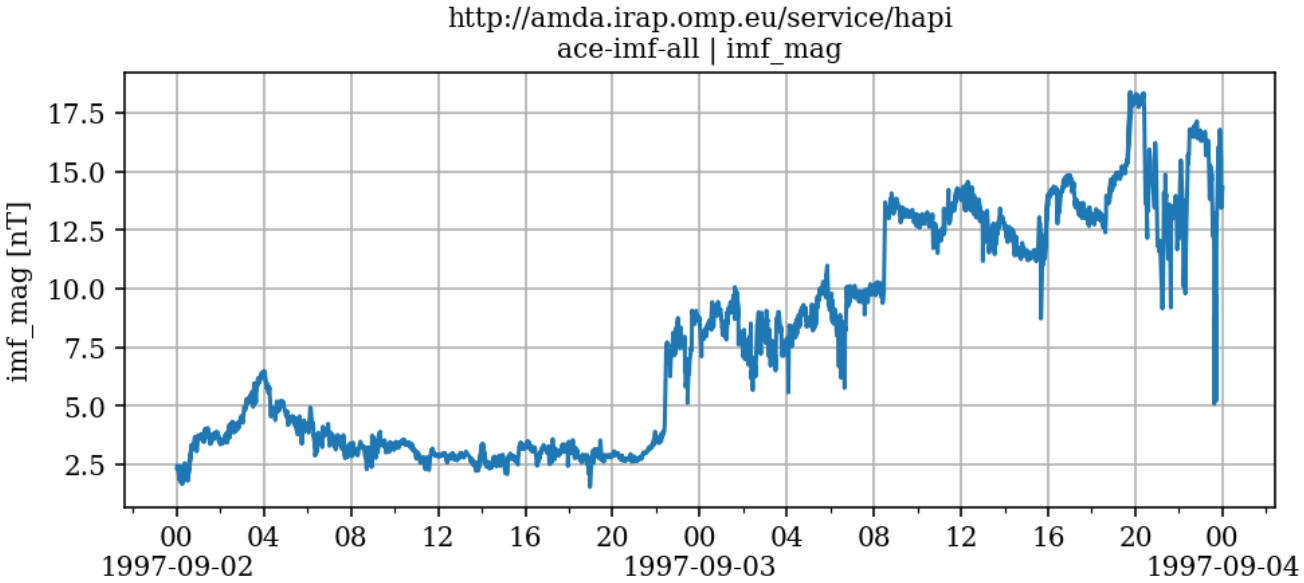
Contact: amda@irap.omp.eu

http://amda.irap.omp.eu/service/hapi

HAPI server – Some examples



http://autoplot.org/



http://hapi-server.org/servers/

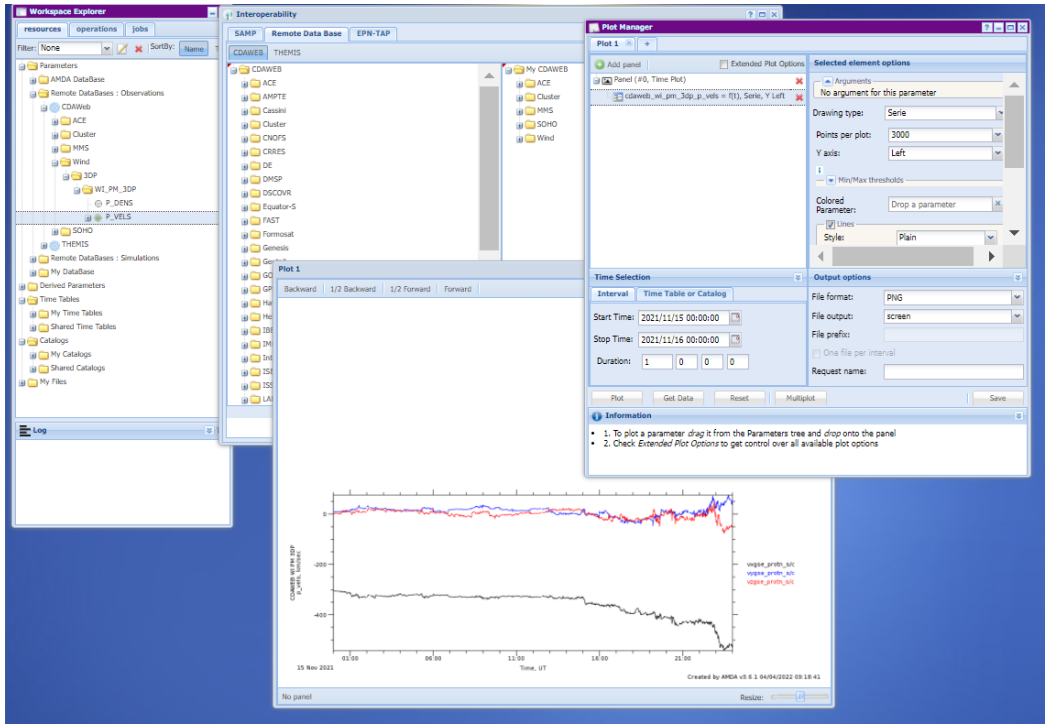
Perspectives and improvements

What we need to do quickly:

- Upgrade to the last hapi-server release (probably with the Docker image)
- Add parameters with time-varying bins (new in HAPI 3)

What we expect to do as soon as possible:

- Use our Python client « speasy » to access data for HAPI API:
  - https://pypi.org/project/speasy/
  - Implements a caching system
- Implement a HAPI client in AMDA



Conclusion

HAPI implementation has been very easy in AMDA:

- Due to the minimum set of capabilities define by the specification
- Thanks to the HAPI Server Front-End implementation (https://github.com/hapi-server/server-nodejs)
- => Less than 5 days of work
- Currently, we don’t have any usage statistics (we only have statistics for all access through our WebService)