europlanet 2024 Research Infrastructure

Updated heliophysics services in VESPA: science products, service design and capabilities

Baptiste CECCONI
Chloé AZRIA

LESIA l'Observatoire de Paris | PSL
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

PADC Paris Astronomical Data Centre

Heliophysics services in VESPA
(Virtual European Solar and Planetary Access)

- Heliophysics:
  Solar physics, interplanetary medium, planetary magnetospheres/plasma

- Remote sensing: images, spectra, dynamic spectra, events, cubes
  In-situ: time-series, dynamic spectra, events
  Modelled: images, spectra, time-series, dynamic spectra, events, cubes

- VESPA: TAP table compliant with EPNcore dictionary

- Updated services from ObsParis and Nançay

Updated Heliophysics Services

- BASS2000: daily solar images at various wavelengths
- HFC1AR/HFC1T3: heliophysics features: active regions, radio bursts
- NRH: Nançay Radio Heliograph: images, movies
- ORFEES: Nançay solar monitoring antenna: dynamic spectra
- NDA: Nançay Decameter Array: dynamic spectra (Sun and Jupiter)
- MASER (Voyager/PRA, Cassini/Kronos, Wind/Waves, Juno/Waves…): dynamic spectra, events, times-series (Sun and planets)
- ExPRES: modelled dynamic spectra (Jupiter)

- **Update:** Move to DaCHS 2.5, add datalink support, new metadata ingestion methods…

Updates and new features

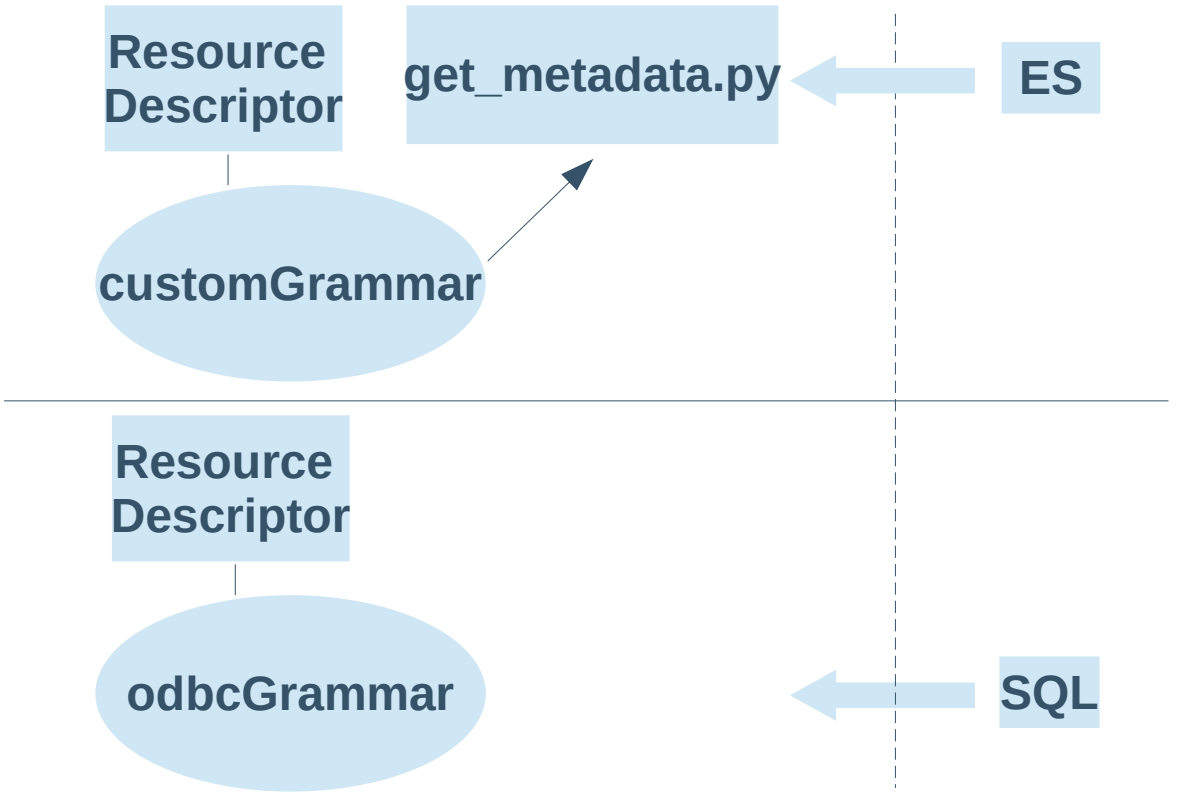
- Using new metadata ingestion method.
- Using datalink when possible, to link with:
  - progenitor data
  - quicklooks (often several formats available)
  - documentation / metadata
  - data access API (see below)
- Data access API for time-series and dynamic spectra (data streaming):
  - HAPI (Heliophysics API: https://github.com/hapi-server)
  - Das2 (temporal resampling on the y: http://das2.org)
- Serve catalogues of spectral-temporal features (using TFCat format)
- Serve collections and datasets associated to recent publications (e.g., supplementary material)

EPN-TAP Helio Services
Technical aspects

DaCHS

EPN-TAP standard

- NRH / ORFEES
  - CustomGrammar
    - Elasticsearch
- bass2000, hfc1ar, hfc1t3
  - odbcGrammar



NRH/ORFEES CustomGrammar

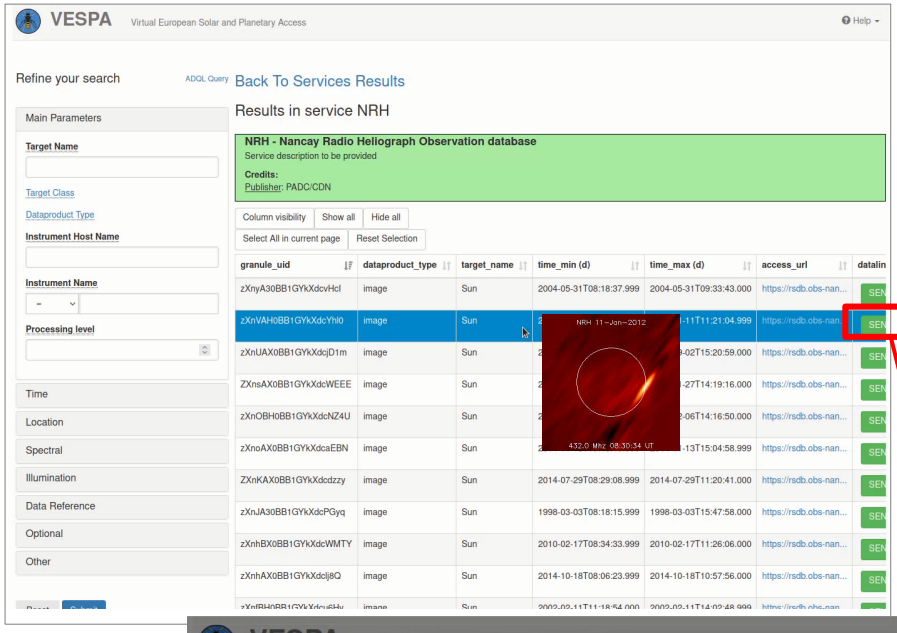
Gathering metadata

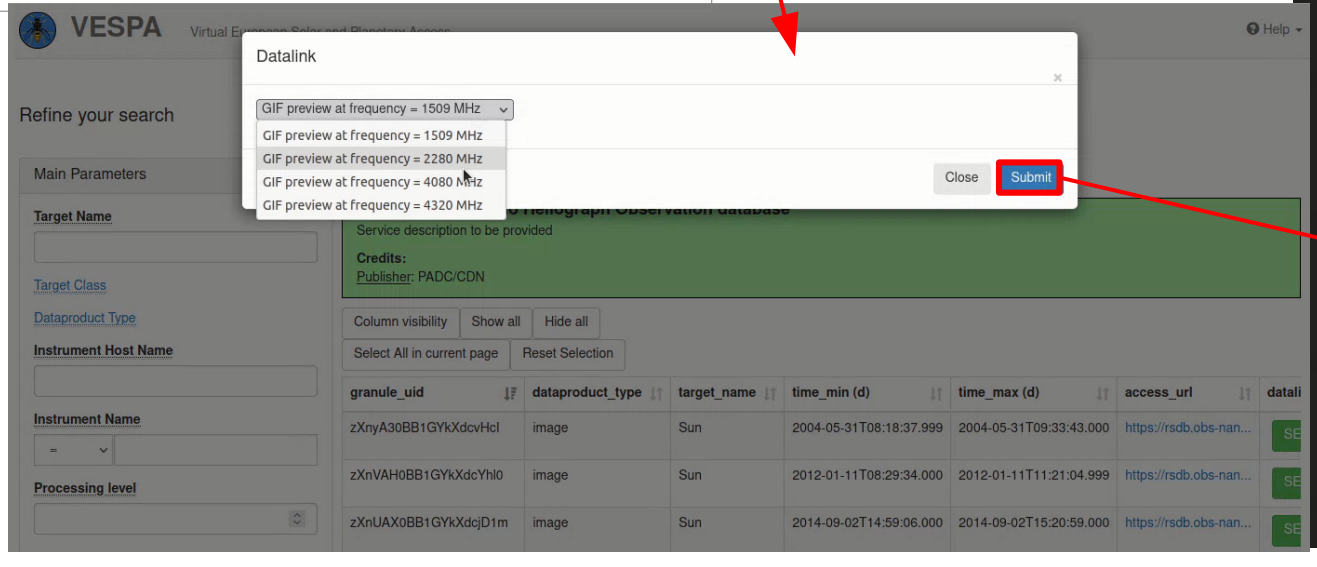
- CustomGrammar calls the get_metadata.py
- Json containing query parameters
- Elsaticsearch.helpers.scan
  - To iterate over query result

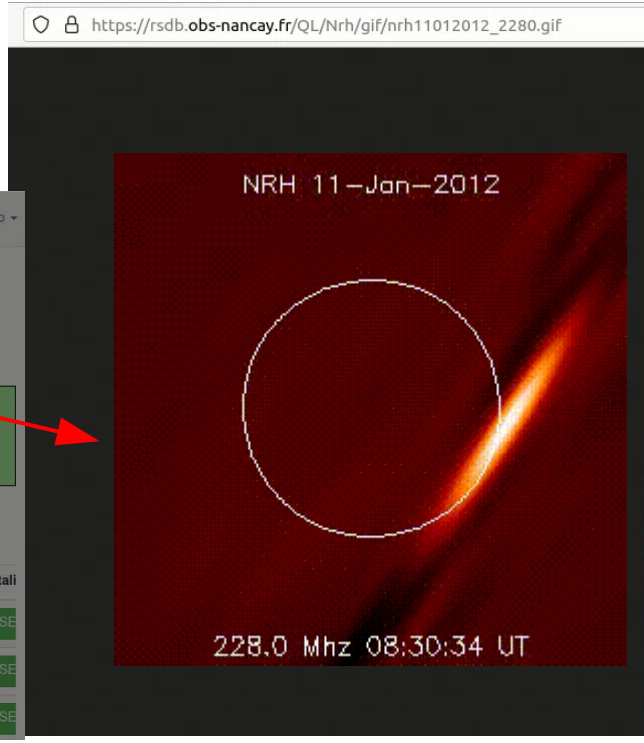
Extract of get_metadata.py

```
class RowIterator(CustomRowIterator):
    def _iterRows(self):
        with open(self.sourceToken, 'r') as f:

            rsdb_search_params = json.load(f)
            headers={'Content-type' :'application/json'}

            es = Elasticsearch(rsdb_search_params['query_url'],verify_certs=False)


            results = elasticsearch.helpers.scan(es,
                index="nrh",
                query= rsdb_search_params['query_data'])


            for item in results:
                #print(item["_source"])
                md = my_metadata(item['_id'], item['_source'])

                yield md
```

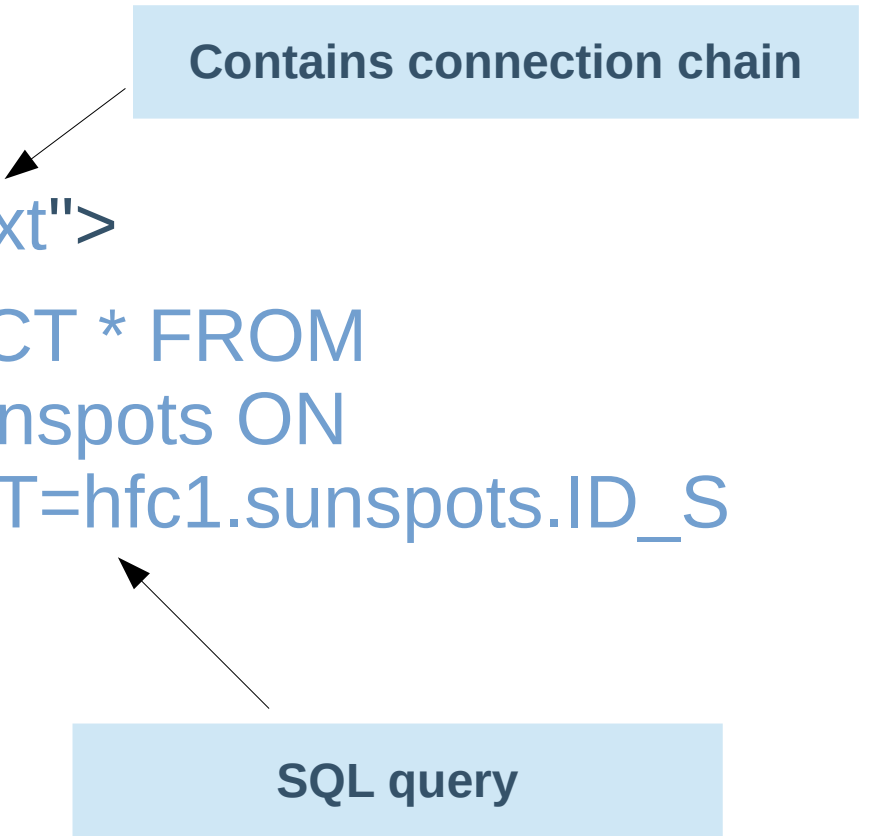
NRH thumbnails / datalinks







Bass2000, hfc1ar, hfc1t3
odbcGrammar



Hfc1ar : s_region active regions

- From Chaincode to s_region
  - Chaincode : contour of the active region
    - Each number in the chain defines the location of next pixel
  - Coordinate conversions :
    - Pixels , original record referential
      - chaincode
    - Helioprojective : center of the sun, angular radius
      - Pairs of coordinates (x,y)
    - Carrington : spherical coordinates in an absolute referential
      - Pairs of coordinates (lon,lat)

`sunpy.net.helio.chaincode`

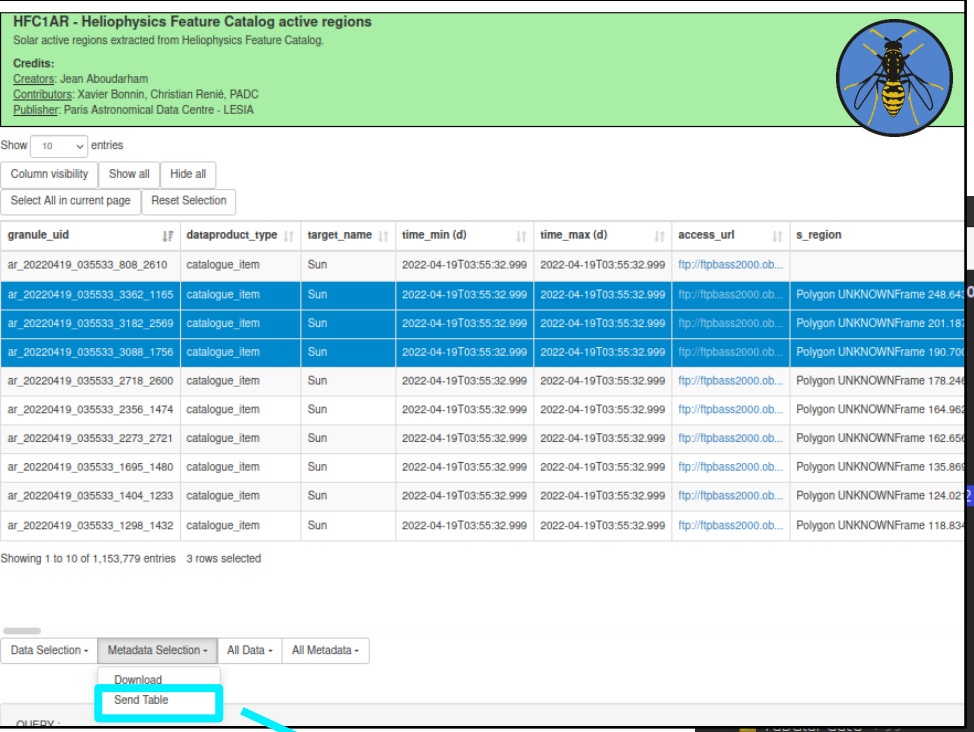
`astropy.coordinates.SkyCoord`
`transform_to`
`sunpy.coordinates frames`

s_region

- Transform (lon,lat) pairs into s_region polygons
- Only simple polygons allowed

  shapely.is_simple funtion
  - Simple polygons
    - « Polygon lon1 lat1 lon2 lat2 … »
  - Multi polygons
    - Shapely.make_simple function returns a list of shapes
      - Several polygons : regroup in one, small circle to make the union
      - Other shapes : (eg : line, multipolygon) solutions to be found

s_region

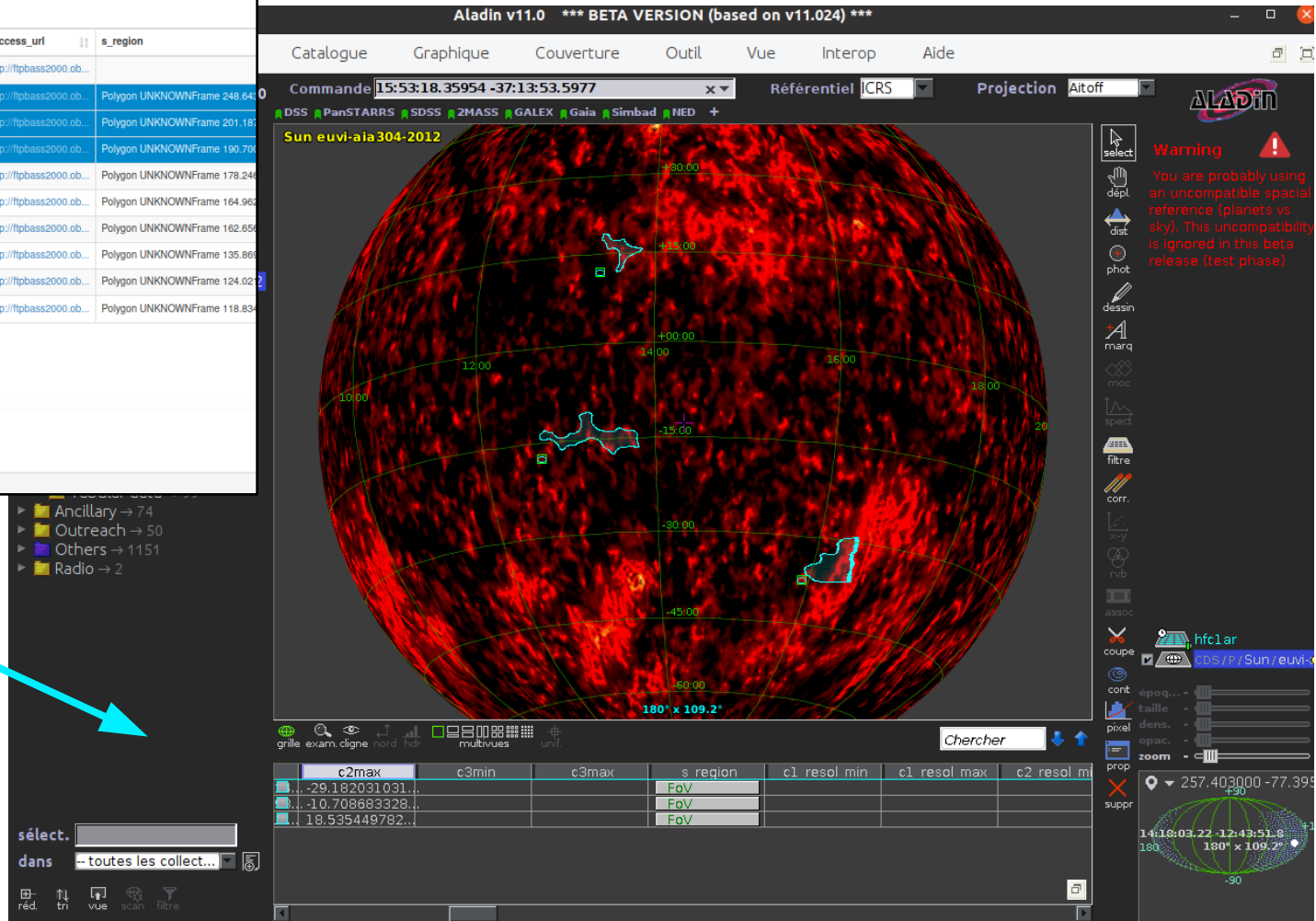


SAMP

VESPA

↓

Aladin Beta



hfc1ar: update

Heavy database
odbcGrammar with **update** : new **<makeQuery>** element (DaCHS 2.5.5)

```
<table id="epn_core" onDisk="True" adql="True" primary="id_ar" dupePolicy="dropOld" >
```

```
<data id="import" updating="True">
        <sources pattern="data/driver.txt"/>
        <odbcGrammar>
                <makeQuery>
                        <code>
                                try:
                                        with base.getTableConn() as conn:
                                                print(next(conn.query("SELECT MAX(time_min) FROM \schema.epn_core"))[0])

                                                localMax = next(conn.query("SELECT MAX(time_min) FROM \schema.epn_core"))[0]
                                                fragment = " WHERE (jdint + jdfrac) >= {}".format(escapeSQL(localMax))
                                                fragment=str(fragment)
                                                print(fragment)
                                except base.DBError as msg:
                                        base.ui.notifyWarning(f"{msg} while harvesting: full re-harvest")
                                        fragment = ""


                                return f"SELECT * FROM hfc1.view_ar_hqi JOIN hfc1.activeregions USING (id_ar) {fragment}"

                        </code>
                </makeQuery>
```

Access the posgtgres database of the service – previously imported

Build a fragment of the odbc query

Return the final odbc query
Default : query without fragment

Daily cron with service import

EUROPLANET 2024 Research Infrastructure

Updated heliophysics services in VESPA: science products, service design and capabilities

support.epntap@obspm.fr

Baptiste CECCONI
Chloé AZRIA

LESIA l'Observatoire de Paris | PSL
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

PADC Paris Astronomical Data Centre

Updated heliophysics services in VESPA: science products, service design and capabilities

Baptiste CECCONI
Chloé AZRIA

LESIA l'Observatoire de Paris | PSL
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

PADC
Paris Astronomical Data Centre

Heliophysics services in VESPA
(Virtual European Solar and Planetary Access)

- Heliophysics:
  Solar physics, interplanetary medium, planetary magnetospheres/plasma
- Remote sensing: images, spectra, dynamic spectra, events, cubes
  In-situ: time-series, dynamic spectra, events
  Modelled: images, spectra, time-series, dynamic spectra, events, cubes
- VESPA: TAP table compliant with EPNcore dictionary
- Updated services from ObsParis and Nançay

Updated Heliophysics Services

- BASS2000: daily solar images at various wavelengths
- HFC1AR/HFC1T3: heliophysics features: active regions, radio bursts
- NRH: Nançay Radio Heliograph: images, movies
- ORFEES: Nançay solar monitoring antenna: dynamic spectra
- NDA: Nançay Decameter Array: dynamic spectra (Sun and Jupiter)
- MASER (Voyager/PRA, Cassini/Kronos, Wind/Waves, Juno/Waves…): dynamic spectra, events, times-series (Sun and planets)
- ExPRES: modelled dynamic spectra (Jupiter)

- **Update:** Move to DaCHS 2.5, add datalink support, new metadata ingestion methods…

Updates and new features

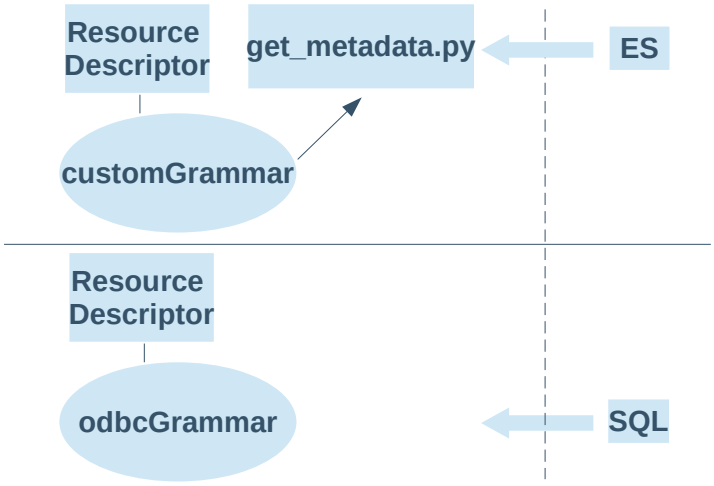
- Using new metadata ingestion method.
- Using datalink when possible, to link with:
  - progenitor data
  - quicklooks (often several formats available)
  - documentation / metadata
  - data access API (see below)
- Data access API for time-series and dynamic spectra (data streaming):
  - HAPI (Heliophysics API: https://github.com/hapi-server)
  - Das2 (temporal resampling on the y: http://das2.org)
- Serve catalogues of spectral-temporal features (using TFCat format)
- Serve collections and datasets associated to recent publications (e.g., supplementary material)

EPN-TAP Helio Services
Technical aspects

DaCHS

EPN-TAP standard

- NRH / ORFEES
  - CustomGrammar
    - Elasticsearch
- bass2000, hfc1ar, hfc1t3
  - odbcGrammar



5

I will talk about technical aspects of other helophysics services we have updated.

These services are set up with DaCHS and following EPN-TAP standards.

The services NRH and ORFEES, provides data from nancay radio observations. It is deployed using the customgrammar method of dachs. Which means the resource descriptor calls an external python script. This python script interrogates the elasticsearch database to extract metadata and fill the table.

The services bass2000, hfc1 active regions and hfc type 3 uses odbc grammar method to fill the epn_core table from SQL databases.

NRH/ORFEES
CustomGrammar

Gathering metadata

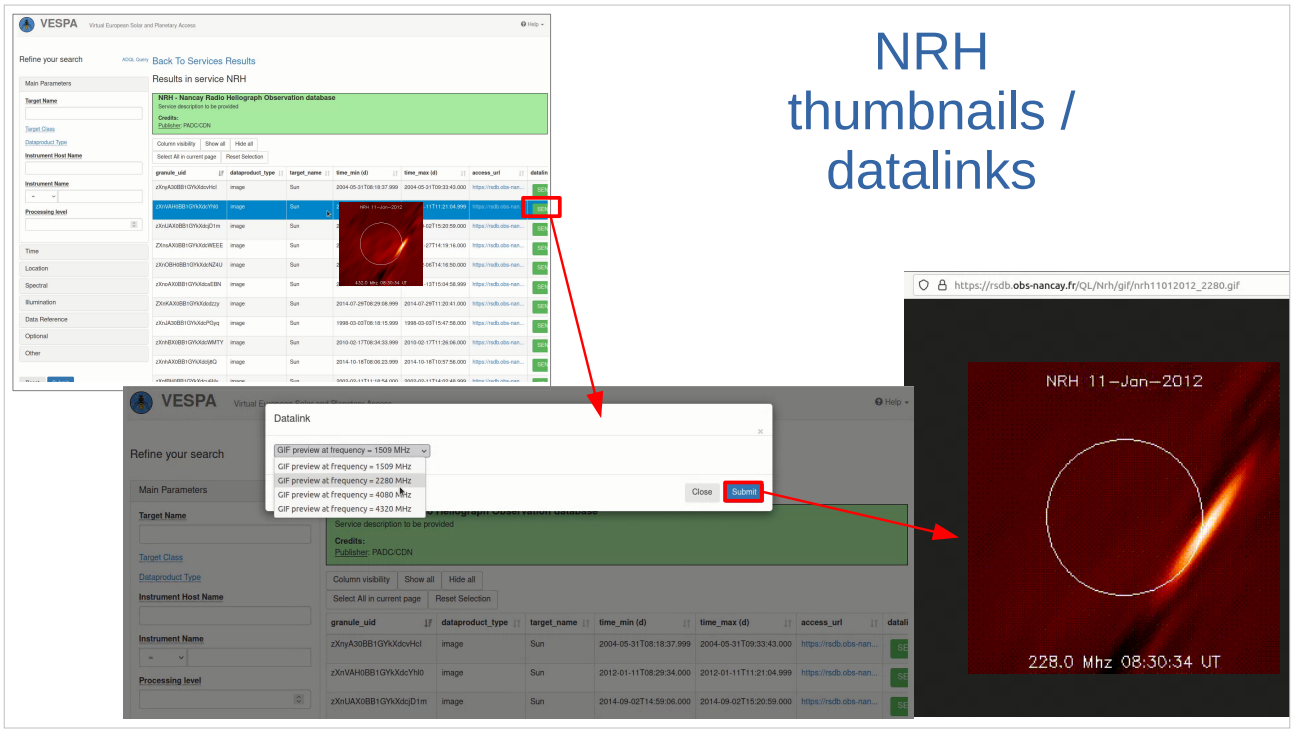
- CustomGrammar calls the get_metadata.py
- Json containing query parameters
- Elsaticsearch.helpers.scan
  - To iterate over query result

Extract of get_metadata.py

```
class RowIterator(CustomRowIterator):
    def _iterRows(self):
        with open(self.sourceToken, 'r') as f:

            rsdb_search_params = json.load(f)
            headers={'Content-type' :'application/json'}

            es = Elasticsearch(rsdb_search_params['query_url'],verify_certs=False)


            results = elasticsearch.helpers.scan(es,
                index="nrh",
                query= rsdb_search_params['query_data'])


            for item in results:
                #print(item["_source"])
                md = my_metadata(item['_id'], item['_source'])

                yield md
```

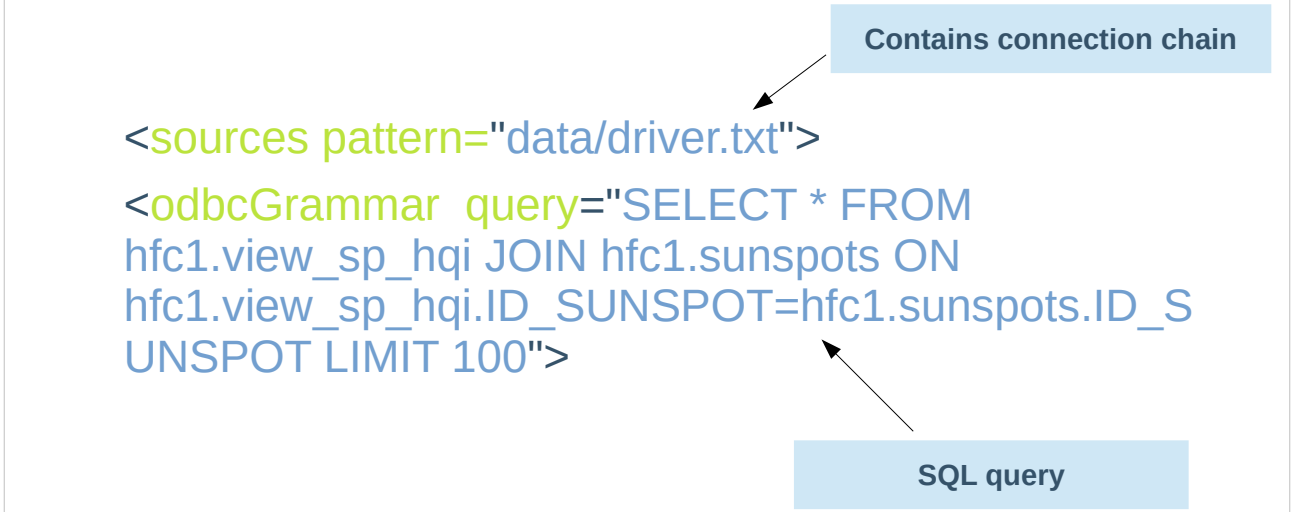

The python script here called by the customgrammar method of dachs must contain the iterow method. It uses the files given in the sources element of the resource descriptor to make the import. Here the only file needed is a json file containing the query applied to the elasticsearch database.
As we want to obtain all the results matching this query and not just a sample like elasticsearch provides natively, we use elasticsearch helpers scan that iterates over query results.

Then, the function my_metadata written somewhere else in the python script makes a mapping.



Here you can see the quicklooks of the Nancay radioHeliograph service which are gifs video-plots. The thumbnail is chosen as the higher-frequency available video-plot.
In datalinks we provide a list of alternative quicklooks at different frequencies. (excluding some frequencies where there is too much noise).
This is done by making a post query to the database in the datalink element to obtain links to the gifs with the matching unique_id.

Bass2000, hfc1ar, hfc1t3
odbcGrammar



For the services bass 2000, hfc1 active regions and hfc1 type 3 that uses odbcgrammar. In the resource descriptor, the sources links to the driver (the remote database connection information)
And the SQL source database is interrogated with a query into the odbcgrammar element.

Hfc1ar : s_region active regions

- From Chaincode to s_region
  - Chaincode : contour of the active region
    - Each number in the chain defines the location of next pixel
  - Coordinate conversions :
    - Pixels , original record referential
      - chaincode
    - Helioprojective : center of the sun, angular radius
      - Pairs of coordinates (x,y)
    - Carrington : spherical coordinates in an absolute referential
      - Pairs of coordinates (lon,lat)

`sunpy.net.helio.chaincode`

`astropy.coordinates.SkyCoord`
`transform_to`
`sunpy.coordinates frames`

One of the difficulties of this update was to add s_regions on the active regions service.
The information provided initially was chaincodes of the contour. A chaincode is a string of numbers where each number represents a shift to the next pixel.
So we had to make the conversion between the original chaincode in the original record pixel referential to helioprojective pairs of x,y positions. The helioprojective referential is a projected referential centered on the sun and the x and y axis increment are the angular radius. We do that using sunpy.net.helio.chaincode and the informations given in the database.

Then, the next step is to convert these projected coodinates on carrington heliographic coordinates, which is a spherical referential centered on the center of the sun that does not depends on the observator location. The astropy.coordinates.SkyCoord allows to create an instance of a position in a given referential defined thanks to sunpy.coordinates.frames so we can use the transnform_to method to obtain the pairs of coordinates in carrington.

s_region

- Transform (lon,lat) pairs into s_region polygons
- Only simple polygons allowed

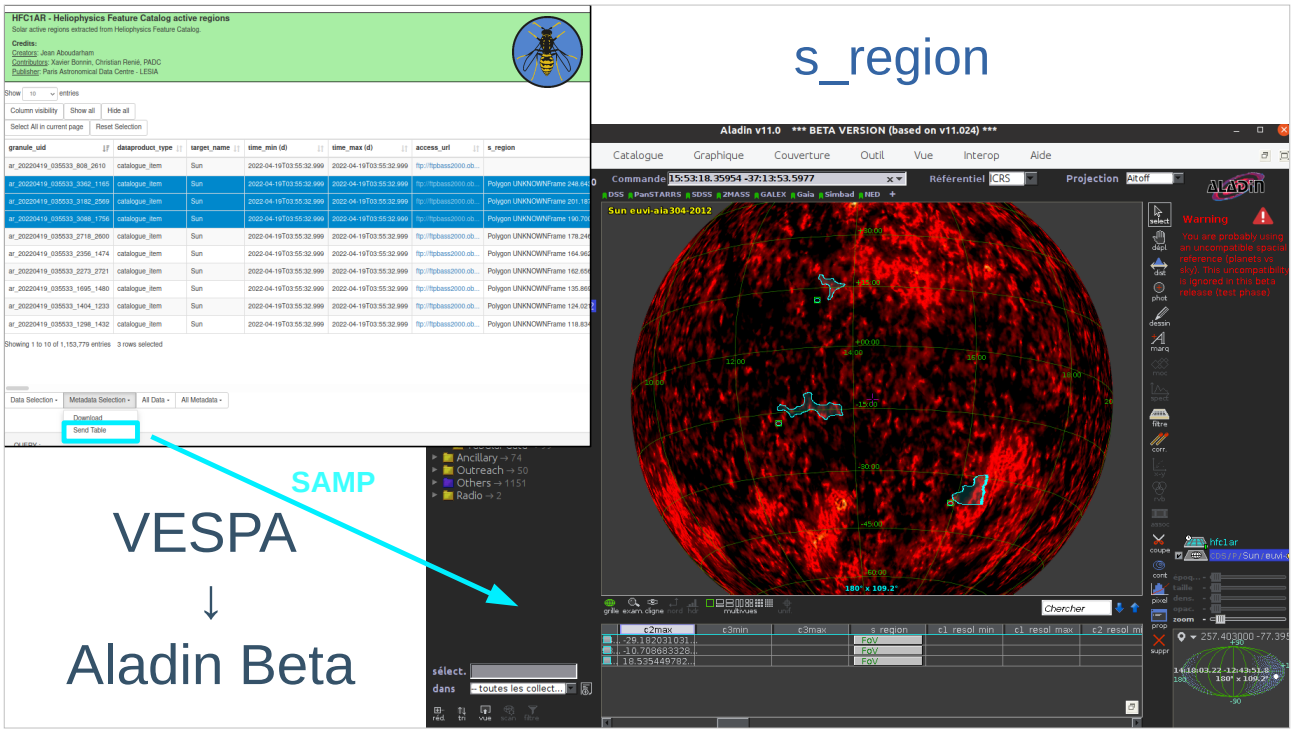
  shapely.is_simple funtion
  - Simple polygons
    - « Polygon lon1 lat1 lon2 lat2 … »
  - Multi polygons
    - Shapely.make_simple function returns a list of shapes
      - Several polygons : regroup in one, small circle to make the union
      - Other shapes : (eg : line, multipolygon) solutions to be found

Antother difficulty of the s_region build is that DaCHS only handle simple polygons. So, the contour must not cross itself.

For the already-simple polygons, we basically write the s_region with the pairs of longitude and latitude.

For the other, we try to use the make_simple function that returns a list of shapely shapes. If evereything happens like we want, it returns a list of polygons and we make the union using a small circle that makes the original polygons a simple one.

There still some cases where the make_simple does not return a list of polygons, but some other types of shapes.
In this case, we do not yet have a solution.
This case happens mostly when the active regions is located at limbs on the observation



As a result, on the VESPA portal, we can select granules we want, send it using SAMP and plot immediately the s_regions on Aladin bêta in clincking on these buttons.

hfc1ar: update

Heavy database
odbcGrammar with **update** : new **<makeQuery>** element (DaCHS 2.5.5)

```
<table id="epn_core" onDisk="True" adql="True" primary="id_ar" dupePolicy="dropOld" >
<data id="import" updating="True">
        <sources pattern="data/driver.txt"/>
        <odbcGrammar>
                <makeQuery>
                        <code>
                                try:
                                        with base.getTableConn() as conn:
                                                print(next(conn.query("SELECT MAX(time_min) FROM \schema.epn_core"))[0])

                                                localMax = next(conn.query("SELECT MAX(time_min) FROM \schema.epn_core"))[0]
                                                fragment = " WHERE (jdint + jdfrac) >= {}".format(escapeSQL(localMax))
                                                fragment=str(fragment)
                                                print(fragment)
                                except base.DBError as msg:
                                        base.ui.notifyWarning(f"{msg} while harvesting: full re-harvest")
                                        fragment = ""


                                return f"SELECT * FROM hfc1.view_ar_hqi JOIN hfc1.activeregions USING (id_ar) {fragment}"
                        </code>
                </makeQuery>
```

Access the posgtgres database of the service – previously imported

Build a fragment of the odbc query

Return the final odbc query
Default : query without fragment

Daily cron with service import

The active region database is very heavy and updating every day, the added s_regions had made the problem even worse.
Dachs basically re-import the whole service each time we want to make an update, but in our case it takes several days for one import.
So, recently from dachs 2.5.5 (so the beta version) the new makequery element has been made available. So this element allows to adapt the SQL query with python by accessing to the already-imported service.

In the resource descriptor, the data updating must be set to true and the dupe policy in the table must be set as dropold which means that if the primary key is duplicated, it will keep the new row.

So here, we access the local postgres database, and we use a monotonously increasing parameter (here the date, time_min). And the query is made with the fragment here in selecting the part of the

Updated heliophysics services in VESPA: science products, service design and capabilities

support.epntap@obspm.fr

Baptiste CECCONI
Chloé AZRIA

LESIA l'Observatoire de Paris | PSL
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

Thank you for your attention,

Feel free to ask questions or send e mails to support epntap