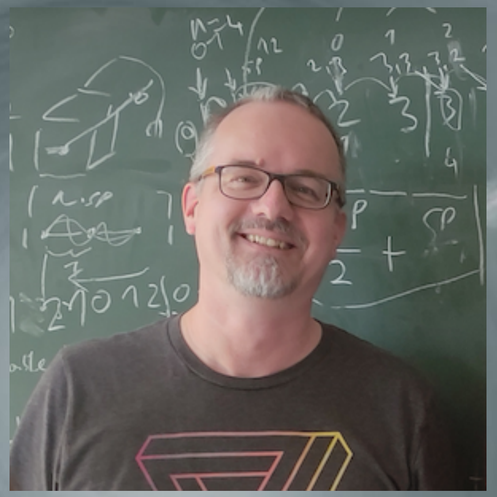
Astro-COLIBRI - The API

COincidence **LIB**rary for **R**eal-time **I**nquiry for multi-messenger astrophysics


Bernardo Cornejo, Fabian Schüssler, Ilja Jaroschewski, Mickäel Costa, Weizmann Kiendrebeogo

CEA Paris-Saclay, IRFU

The app for real-time follow-up of transient sources

How to interact with Astro-COLIBRI without an interface?

The app for real-time follow-up of transient sources

~~The app for real-time follow-up of transient sources~~

The API for real-time follow-up of transient sources

How does the API work?

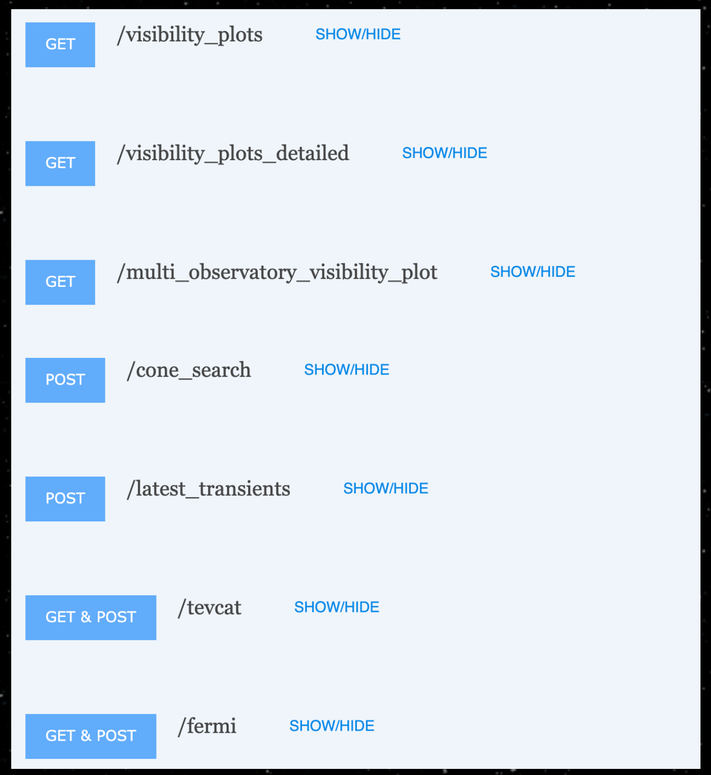
Application Programming Interface

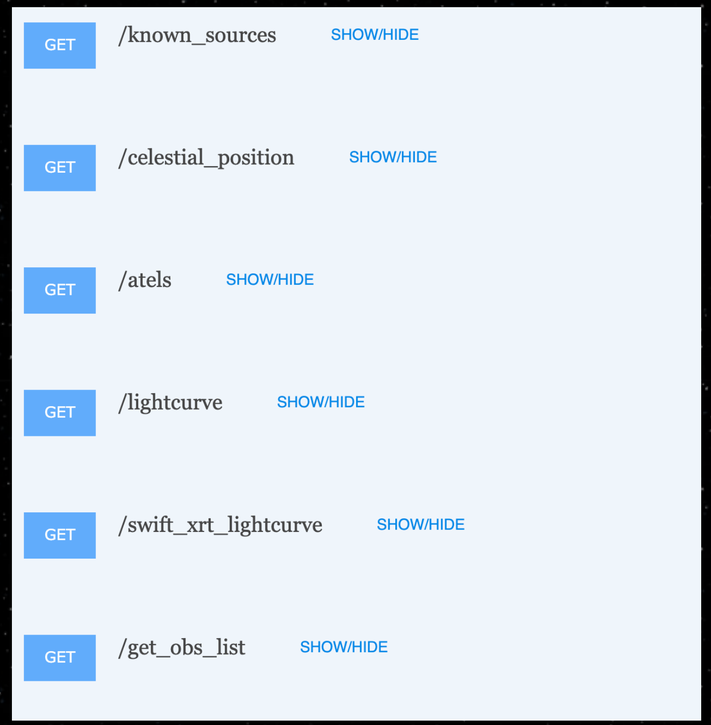
Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

How does the API work?

Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!



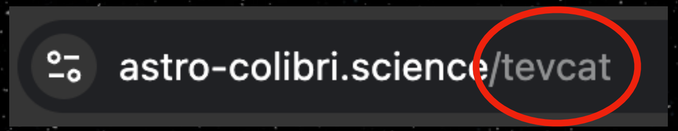


How does the API work?

Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

GET Request




```
[
  {
    "DEC": "-42 35 49",
    "Discovery Date": 201012,
    "Distance": 0.105,
    "Distance Units": "z",
    "Extended": 0,
    "Extent Major Axis": "None",
    "Extent Minor Axis": "None",
    "Flux": 0.005,
    "Galactic Latitude": 20.0640990675711,
    "Galactic Longitude": 307.540262148442,
    "ID": 119,
    "Link": "https://tevcat.org/?mode=1&showsrc=216",
    "RA": "13 14 58.5",
    "Source Name": "1ES 1312-423",
    "Source Type": "HBL",
    "Spectral Index": 2.85,
    "Sub-Catalog": 1,
    "Telescope": "H.E.S.S."
  },
  {
    "DEC": "-40 39 36.0",
    "Discovery Date": 201810
```


```
[1]  import requests
     import json
     import pprint
```

How does the API work?

Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

GET Request

astro-colibri.science/tevcat

```
[3]  ### You can also use this URL in directly in your code

     url = "https://astro-colibri.science/tevcat"

     response = requests.get(url)

     if response.status_code == 200:
         print("Response successfully received.")
         events = response.json()
         print('number of events: ' + str(len(events)))
         pprint.pprint(events)
     else:
         print("Request did NOT succeed : ", response.status_code)
         print("Error message : ", response.content.decode('UTF-8'))
```


```
[
  {
    "DEC": "-42 35 49",
    "Discovery Date": 201012,
    "Distance": 0.105,
    "Distance Units": "z",
    "Extended": 0,
    "Extent Major Axis": "None",
    "Extent Minor Axis": "None",
    "Flux": 0.005,
    "Galactic Latitude": 20.0640990675711,
    "Galactic Longitude": 307.540262148442,
    "ID": 119,
    "Link": "https://tevcat.org/?mode=1&showsrc=216",
    "RA": "13 14 58.5",
    "Source Name": "1ES 1312-423",
    "Source Type": "HBL",
    "Spectral Index": 2.85,
    "Sub-Catalog": 1,
    "Telescope": "H.E.S.S."
  },
  {
    "DEC": "-40 39 36.0",
    "Discovery Date": 201810
```


```
[1]  import requests
     import json
     import pprint
```

How does the API work?

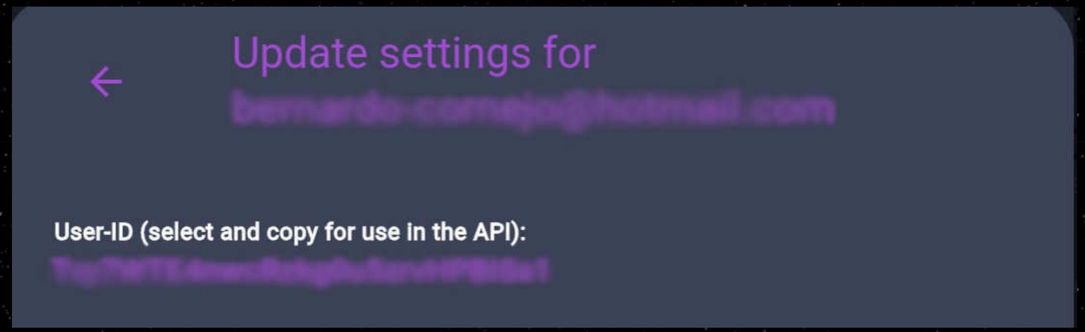
Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

POST Request

```
[ ]  uid = "0bLAEeOfGaf6QeZBpVGaB6XSYuX2"
```

c.f. Ilja's talk




```
[1]  import requests
     import json
     import pprint
```

How does the API work?

Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

POST Request

```
[ ]  uid = "0bLAEeOfGaf6QeZBpVGaB6XSYuX2"
```

```
[ ]  URL_TEVCAT = 'http://astro-colibri.science/tevcat'
     # Request parameters (headers, body)
     headers = {"Content-Type": "application/json"}
     body = {
         "uid": uid,
     }

     # Perform the POST request
     response = requests.post(URL_TEVCAT, headers=headers, data=json.dumps(body))

     # Process the response
     if response.status_code == 200:
         print("Response successfully received.")
         events = response.json()
         print('number of events: ' + str(len(events)))
         pprint.pprint(events)
     else:
         print("Request did NOT succeed : ", response.status_code)
         print("Error message : ", response.content.decode('UTF-8'))
```


```
[
  {
    "DEC": "-42 35 49",
    "Discovery Date": 201012,
    "Distance": 0.105,
    "Distance Units": "z",
    "Extended": 0,
    "Extent Major Axis": "None",
    "Extent Minor Axis": "None",
    "Flux": 0.005,
    "Galactic Latitude": 20.0640990675711,
    "Galactic Longitude": 307.540262148442,
    "ID": 119,
    "Link": "https://tevcat.org/?mode=1&showsrc=216",
    "RA": "13 14 58.5",
    "Source Name": "1ES 1312-423",
    "Source Type": "HBL",
    "Spectral Index": 2.85,
    "Sub-Catalog": 1,
    "Telescope": "H.E.S.S."
  },
  {
    "DEC": "-40 39 36.0",
    "Discovery Date": 201810
```


```
[1]  import requests
     import json
     import pprint
```

How does the API work?

Application Programming Interface

Allows you to use Astro-COLIBRI functionnalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

GET Request

astro-colibri.science/tevcat

```
[3]  ### You can also use this URL in directly in your code

     url = "https://astro-colibri.science/tevcat"

     response = requests.get(url)

     if response.status_code == 200:
         print("Response successfully received.")
         events = response.json()
         print('number of events: ' + str(len(events)))
         pprint.pprint(events)
     else:
         print("Request did NOT succeed : ", response.status_code)
         print("Error message : ", response.content.decode('UTF-8'))
```


```
[
  {
    "DEC": "-42 35 49",
    "Discovery Date": 201012,
    "Distance": 0.105,
    "Distance Units": "z",
    "Extended": 0,
    "Extent Major Axis": "None",
    "Extent Minor Axis": "None",
    "Flux": 0.005,
    "Galactic Latitude": 20.0640990675711,
    "Galactic Longitude": 307.540262148442,
    "ID": 119,
    "Link": "https://tevcat.org/?mode=1&showsrc=216",
    "RA": "13 14 58.5",
    "Source Name": "1ES 1312-423",
    "Source Type": "HBL",
    "Spectral Index": 2.85,
    "Sub-Catalog": 1,
    "Telescope": "H.E.S.S."
  },
  {
    "DEC": "-40 39 36.0",
    "Discovery Date": 201810
```


POST Request

```
[ ]  uid = "0bLAEeOfGaf6QeZBpVGaB6XSYuX2"
```

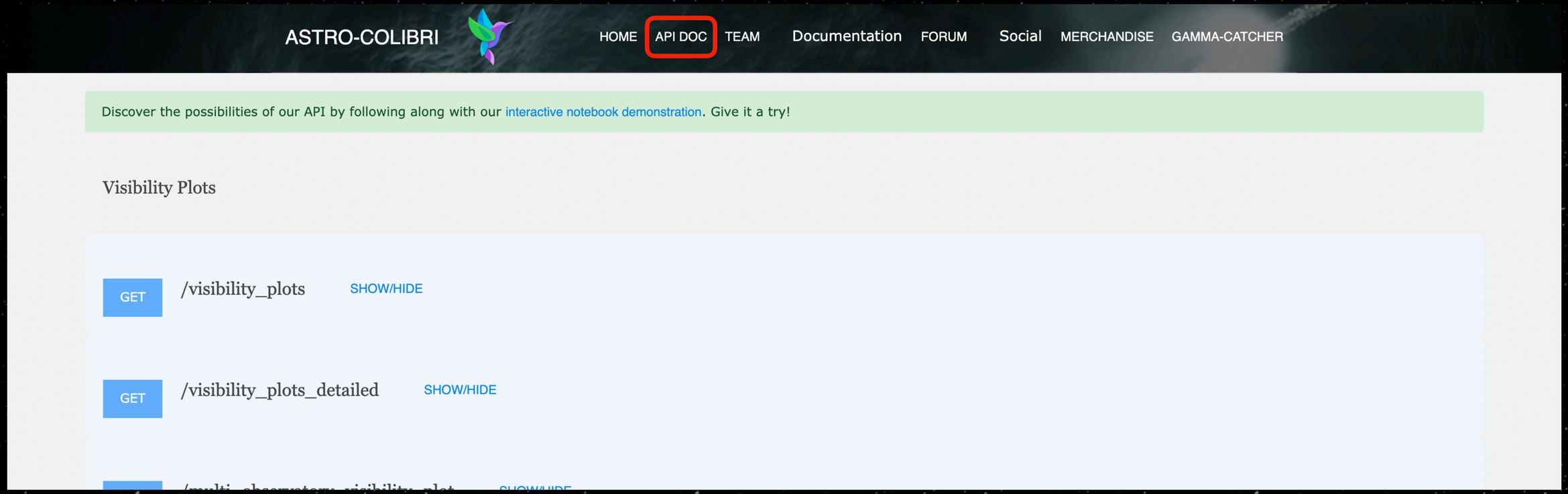
```
[ ]  URL_TEVCAT = 'http://astro-colibri.science/tevcat'
     # Request parameters (headers, body)
     headers = {"Content-Type": "application/json"}
     body = {
         "uid": uid,
     }

     # Perform the POST request
     response = requests.post(URL_TEVCAT, headers=headers, data=json.dumps(body))

     # Process the response
     if response.status_code == 200:
         print("Response successfully received.")
         events = response.json()
         print('number of events: ' + str(len(events)))
         pprint.pprint(events)
     else:
         print("Request did NOT succeed : ", response.status_code)
         print("Error message : ", response.content.decode('UTF-8'))
```


```
[
  {
    "DEC": "-42 35 49",
    "Discovery Date": 201012,
    "Distance": 0.105,
    "Distance Units": "z",
    "Extended": 0,
    "Extent Major Axis": "None",
    "Extent Minor Axis": "None",
    "Flux": 0.005,
    "Galactic Latitude": 20.0640990675711,
    "Galactic Longitude": 307.540262148442,
    "ID": 119,
    "Link": "https://tevcat.org/?mode=1&showsrc=216",
    "RA": "13 14 58.5",
    "Source Name": "1ES 1312-423",
    "Source Type": "HBL",
    "Spectral Index": 2.85,
    "Sub-Catalog": 1,
    "Telescope": "H.E.S.S."
  },
  {
    "DEC": "-40 39 36.0",
    "Discovery Date": 201810
```

API and Documentation

What to give as query parameters for GET requests? Body parameters for POST requests?

What do we get as response fields and parameters?



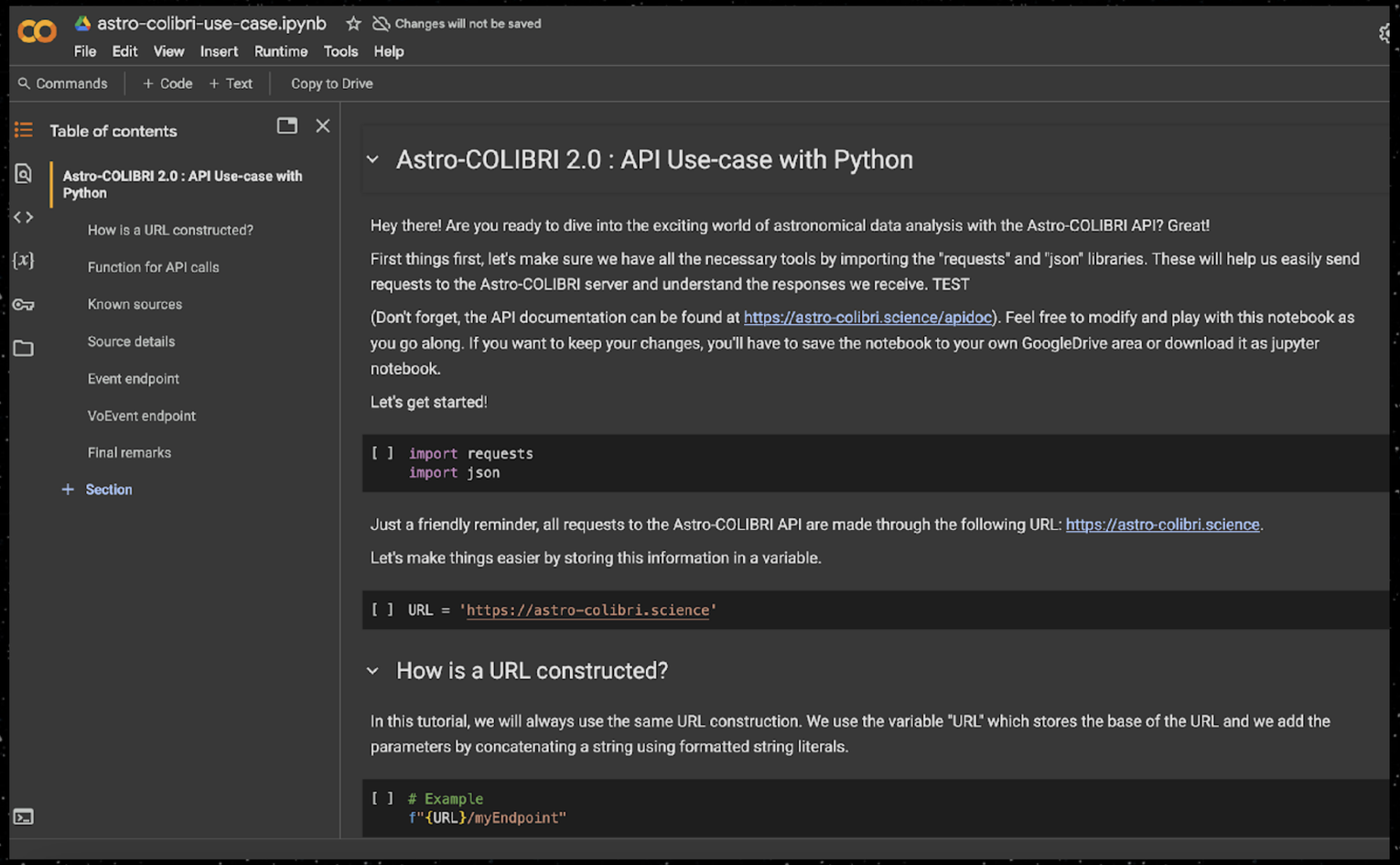
www.astro-colibri.science/apidoc

API and Documentation

Try it yourself!

Plenty of notebooks with **detailed examples**

Discover the possibilities of our API by following along with our interactive notebook demonstration. Give it a try!

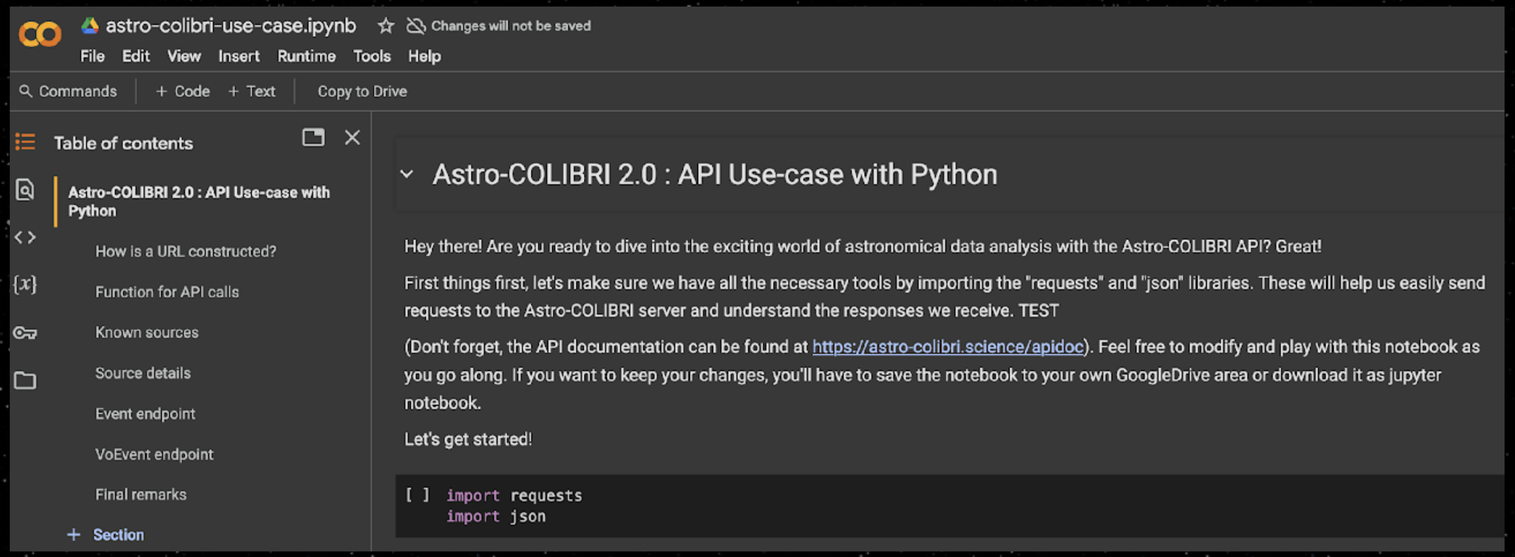


API and Documentation

Try it yourself!

Plenty of notebooks with **detailed examples**

Discover the possibilities of our API by following along with our interactive notebook demonstration. Give it a try!



Input:

- User ID
- Position
- Filters
- Time Window

Output:

Event list in JSON format


```
{
  "DEC": "-42 35 49",
  "Discovery Date": 201012,
  "Distance": 0.105,
  "Distance Units": "z",
  "Extended": 0,
  "Extent Major Axis": "None",
  "Extent Minor Axis": "None",
  "Flux": 0.005,
  "Galactic Latitude": 20.0640990675711,
  "Galactic Longitude": 307.540262148442,
  "ID": 119,
  "Link": "https://tevcat.org/?mode=1&showsrc=216",
  "RA": "13 14 58.5",
  "Source Name": "1ES 1312-423",
  "Source Type": "HBL",
  "Spectral Index": 2.85,
  "Sub-Catalog": 1,
  "Telescope": "H.E.S.S."
},
```


How to do a cone search and filter your events?

How to get the transient events from a given time window?

How to navigate through 4FGL and TeVCat catalogs?

Usage examples
Visibility Plots

GET Requests

/visibility_plot_detailed

Query Parameters:

- Source position
- Observation position
- Twilight level
- Observation date
- Source name

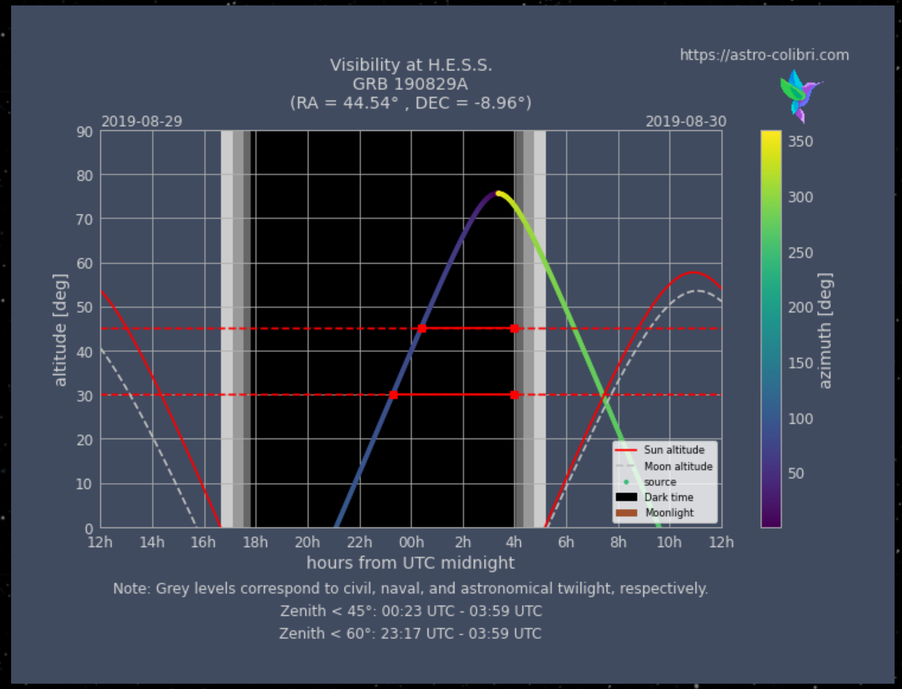
Usage examples
Visibility Plots

GET Requests

/visibility_plot_detailed

Query Parameters:

- Source position
- Observation position
- Twilight level
- Observation date
- Source name



+

Metadata in JSON format
(observation windows)

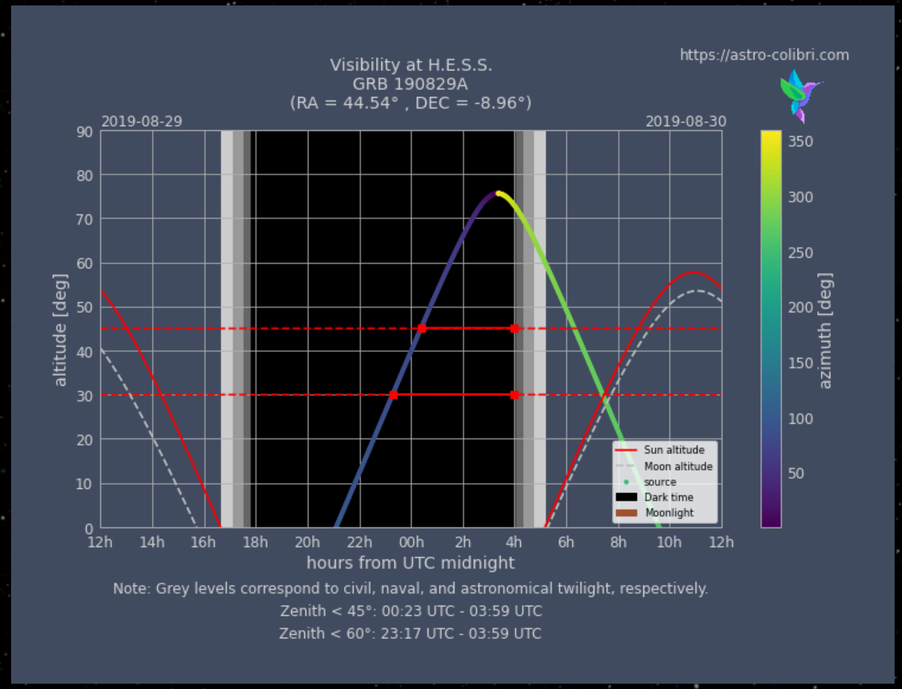
Usage examples
Visibility Plots

GET Requests

/visibility_plot_detailed

Query Parameters:

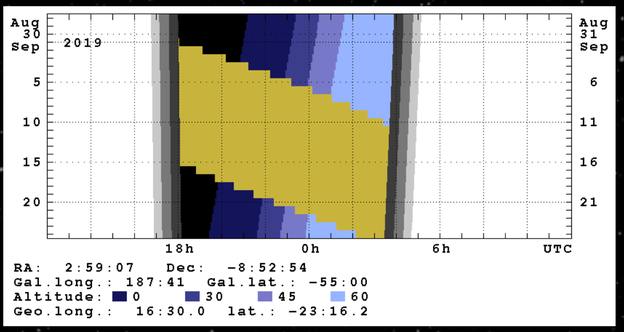
- Source position
- Observation position
- Twilight level
- Observation date
- Source name



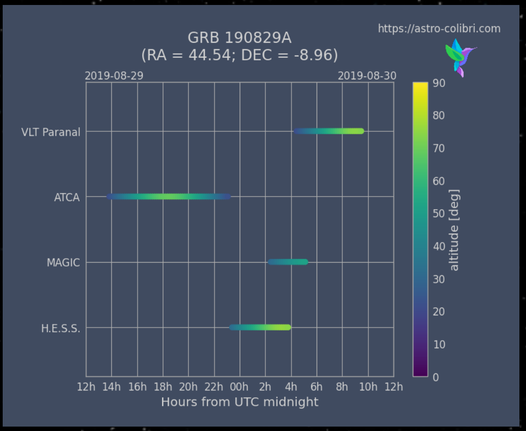
\+

Metadata in JSON format
(observation windows)

/visibility_plot



/multi_observatory_
visibility_plot



Usage examples
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format

Usage examples

Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- **Filter**
- User ID
- Return format

```
body = {
    "uid": uid,
}
```

‘AndSpecification’

‘OrSpecification’

‘NotSpecification’

```
{
    'type': 'AndSpecification',
    'specs': {
        'type': 'OrSpecification',
        'specs': [
            {
                'type': 'NotSpecification',
                'specs': {}
            },
            {
                'type': 'AndSpecification',
                'specs': {}
            }
        ]
    }
}
```

```
my_filters = {
    'type': 'OrSpecification',
    'specs': [
      {
        'type': 'FieldSpecification',
        'field': 'transient_flux',
        'value': 22,
        'operation': '<',
        'typeField': 'float'
      },
      {
        'type': 'FieldSpecification',
        'field': 'observatory',
        'value': 'icecube',
        'operation': '==',
        'typeField': 'string'
      },
      {
        'type': 'FieldSpecification',
        'field': 'observatory',
        'value': 'fermi',
        'operation': '==',
        'typeField': 'string'
      }
    ],
}
```

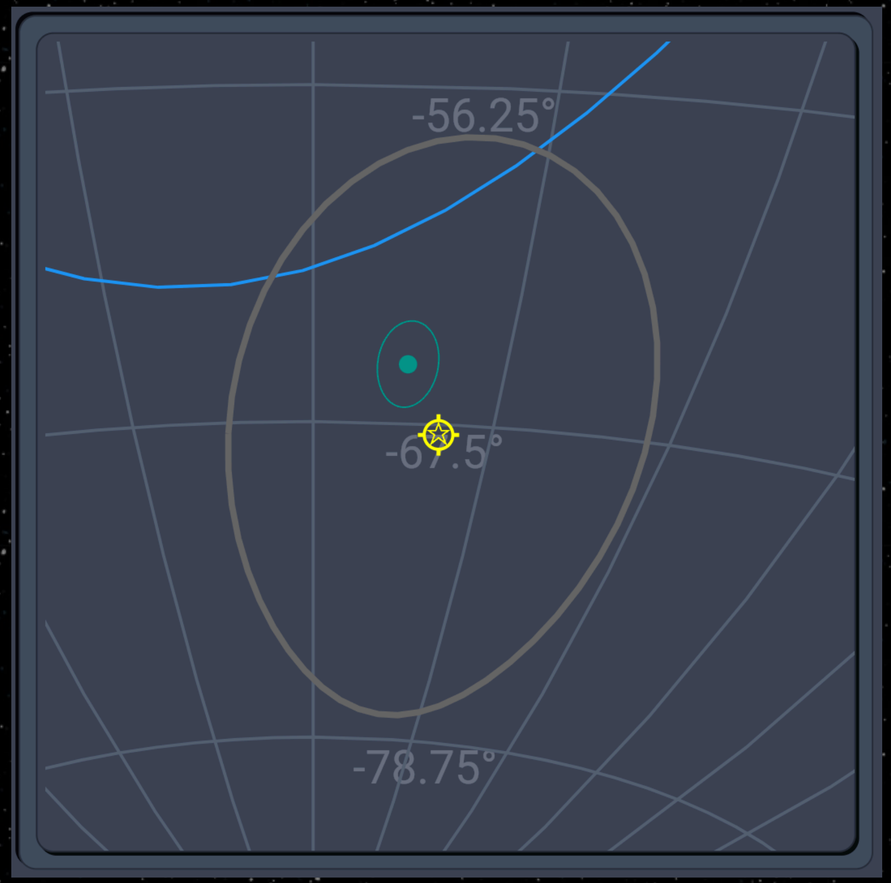
Usage examples
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format



Reponse fields:

- Voevents

Usage examples

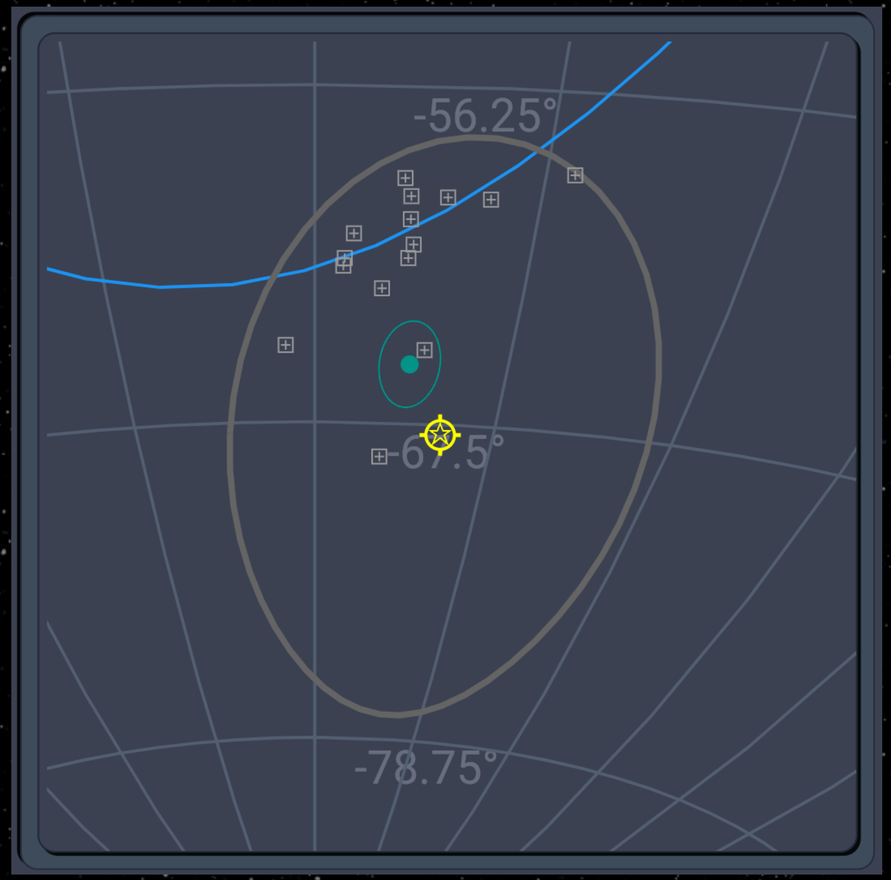
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format



Reponse fields:

- Voevents
- Xsources

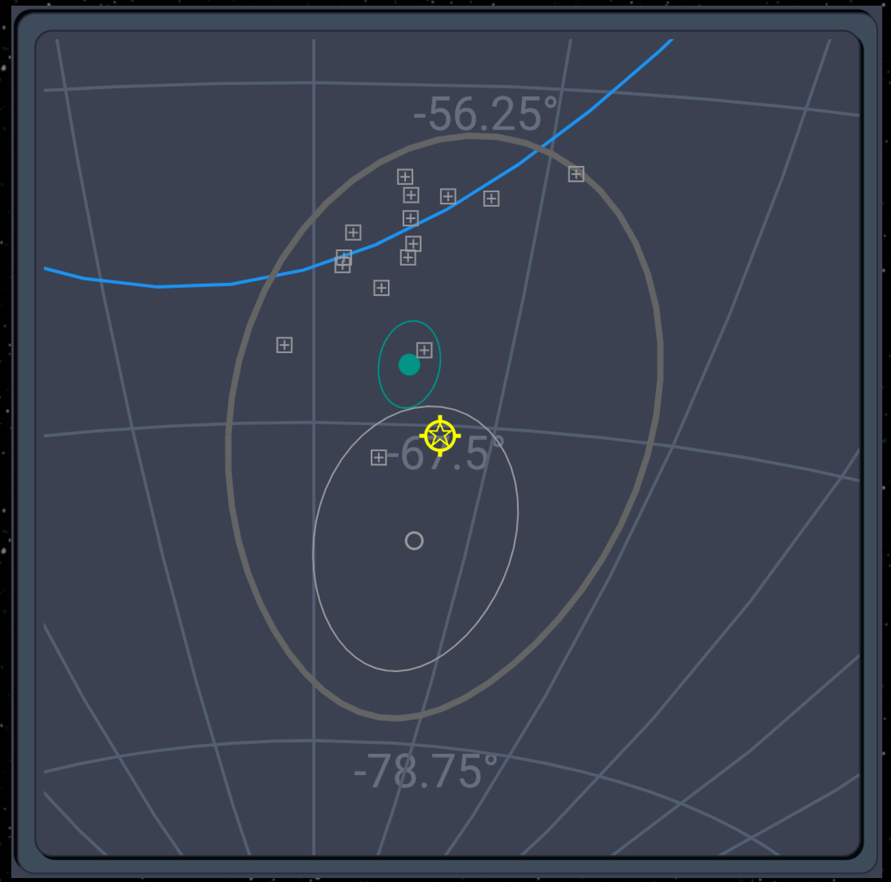
Usage examples
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format



Reponse fields:

- Voevents
- Xsources
- Icecat

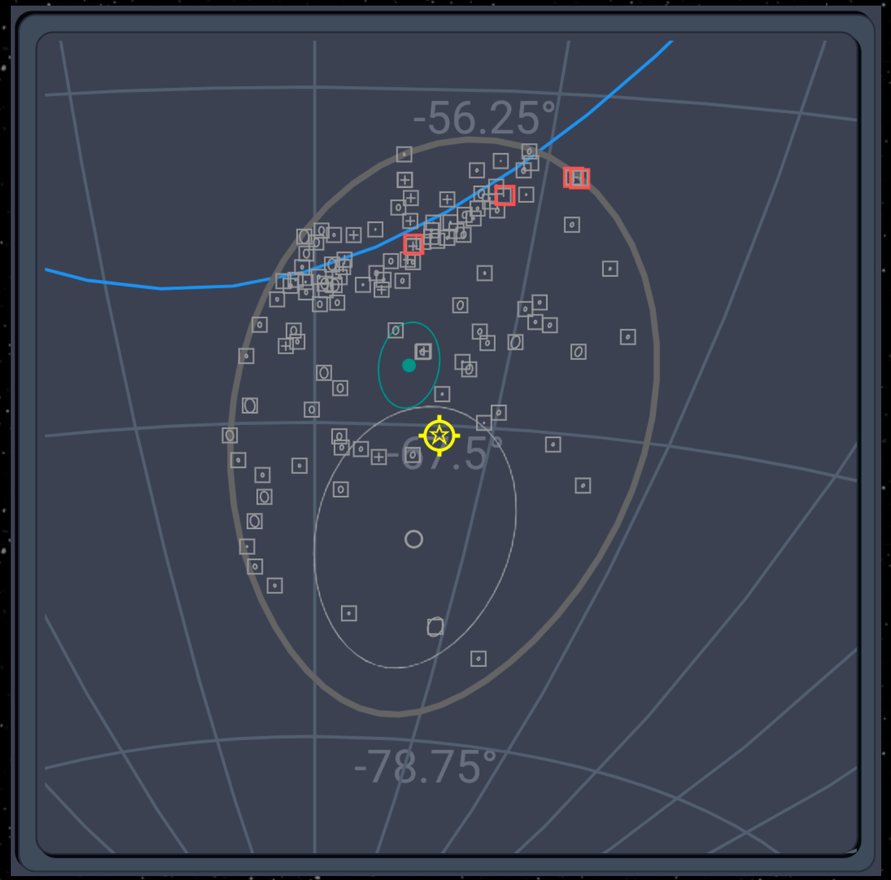
Usage examples
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format



Reponse fields:

- Voevents
- Xsources
- Icecat
- Sources

Usage examples
Cone search

POST Request

/cone_search

Body Parameters:

- Position
- Radius
- Time range
- Filter
- User ID
- Return format

```python
import requests
import json

# Base URL of the API
url = 'https://astro-colibri.science/cone_search'

# Request parameters (headers, body)
headers = {"Content-Type": "application/json"}
body = {
    "uid": uid,
    "time_range": {
        "max": date_max,
        "min": date_min,
    },
    "properties": {
        "type": "cone",
        "position": {"ra": ra, "dec": dec},
        "radius": radius,
    }
}

# Perform the POST request
response = requests.post(url, headers=headers, data=json.dumps(body))

# Process the response
if response.status_code == 200:
    print("Response successfully received.")
    events = response.json()['voevents']
    print('number of events: ' + str(len(events)))
    print(events)  # only show transient events. You can also access catalog sources
else:
    print("Request did NOT succeed : ", response.status_code)
    print("Error message : ", response.content.decode('UTF-8'))
```

Output:

List of sources, voevents, xsources and/or icecat events in a given time range in a given space area

```
Response successfully received.
number of events: 2
[{'alert_type': 'Gnd_Pos', 'arch
```

Usage examples

Latest transients

POST Request

/latest_transients

Body Parameters:

- Time range
- Filter
- User ID
- Return format

```python
import requests
import json

# Base URL of the API
url = 'https://astro-colibri.science/latest_transients'

# Request parameters (headers, body)
headers = {"Content-Type": "application/json"}
body = {
    "uid": uid,
    "filter": my_filters,
    "time_range": {
        "max": date_max,
        "min": date_min,
    },
}

# Perform the POST request
response = requests.post(url, headers=headers, data=json.dumps(body))

# Process the response
if response.status_code == 200:
    print("Response successfully received.")
    events = response.json()['voevents']
    print('number of events: ' + str(len(events)))
    print(json.dumps(events, indent=4))
else:
    print("Request did NOT succeed : ", response.status_code)
    print("Error message : ", response.content.decode('UTF-8'))
```

Output:

List of transient events within a given time window. (With their full list of parameters)

Reponse Field:

- voevents

Usage examples
Catalog search

POST Request

/tevcat & /fermi

Body Parameters:

- Position
- Radius
- Filter
- User ID
- AstroColibriParams
- Return format

```
# Request parameters (headers, body)
headers = {"Content-Type": "application/json"}
body = {
    "uid": uid,
    "AstroColibriParams": True,
    "properties": {
        "type": "cone",
        "position": {"ra": ra, "dec": dec},
        "radius": radius,
    }
}

# Perform the POST request
response = requests.post(URL_FERMI, headers=headers, data=json.dumps(body))

# Process the response
if response.status_code == 200:
    print("Response successfully received.")
    events = response.json()
    print('number of events: ' + str(len(events)))
    pprint.pprint(events)
else:
    print("Request did NOT succeed : ", response.status_code)
    print("Error message : ", response.content.decode('UTF-8'))
```

Output:

List of Fermi or TeVCat cataloged sources (with their full original parameters) filtered by your own parameters

Usage examples
Others

GET Requests

/known_sources

List of the names of all known sources in the Astro-COLIBRI databases

/atels

Astronomers Telegrams associated with a given source

/event

Retrieve the full set of parameters we store for a given transient event

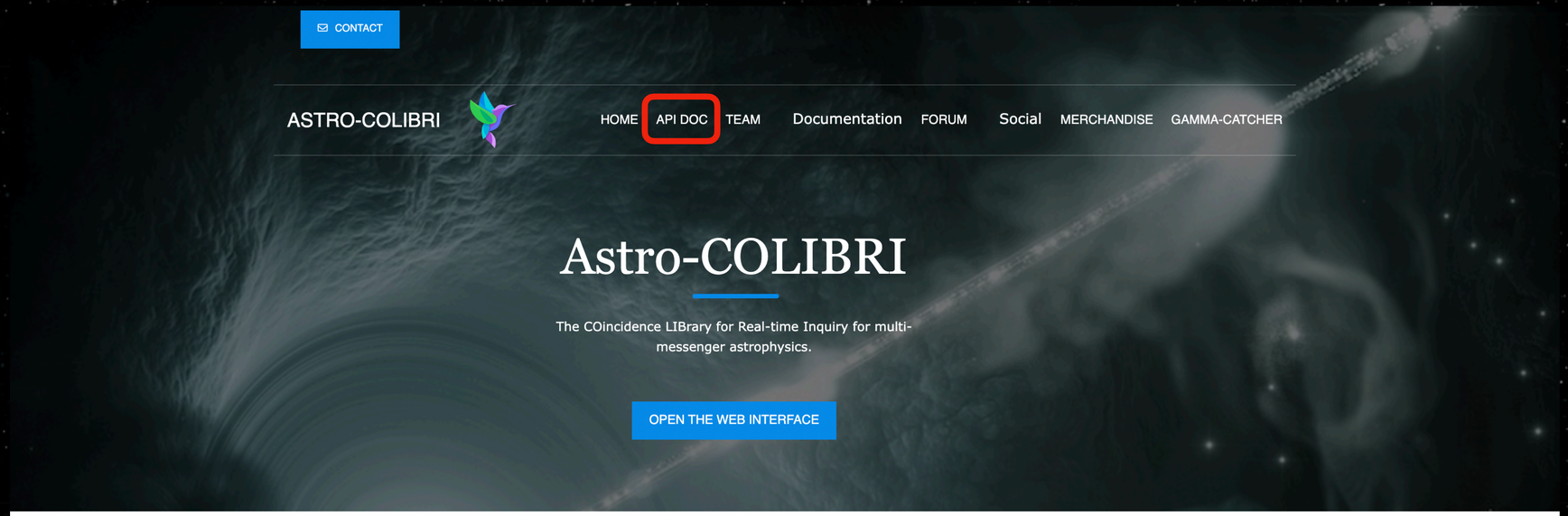
/watchlist

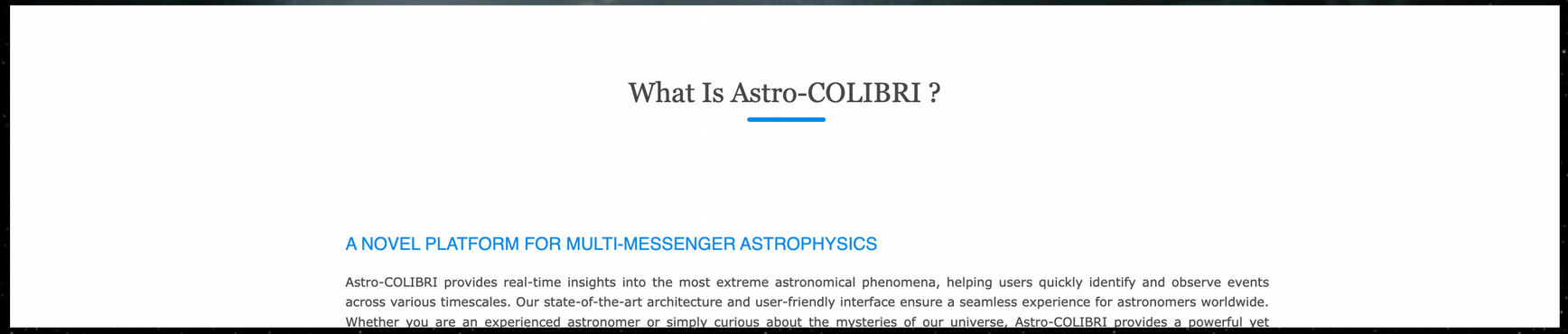
Retrieve the events on your watchlist

And others...

API and Documentation

www.astro-colibri.science





Read our documentation

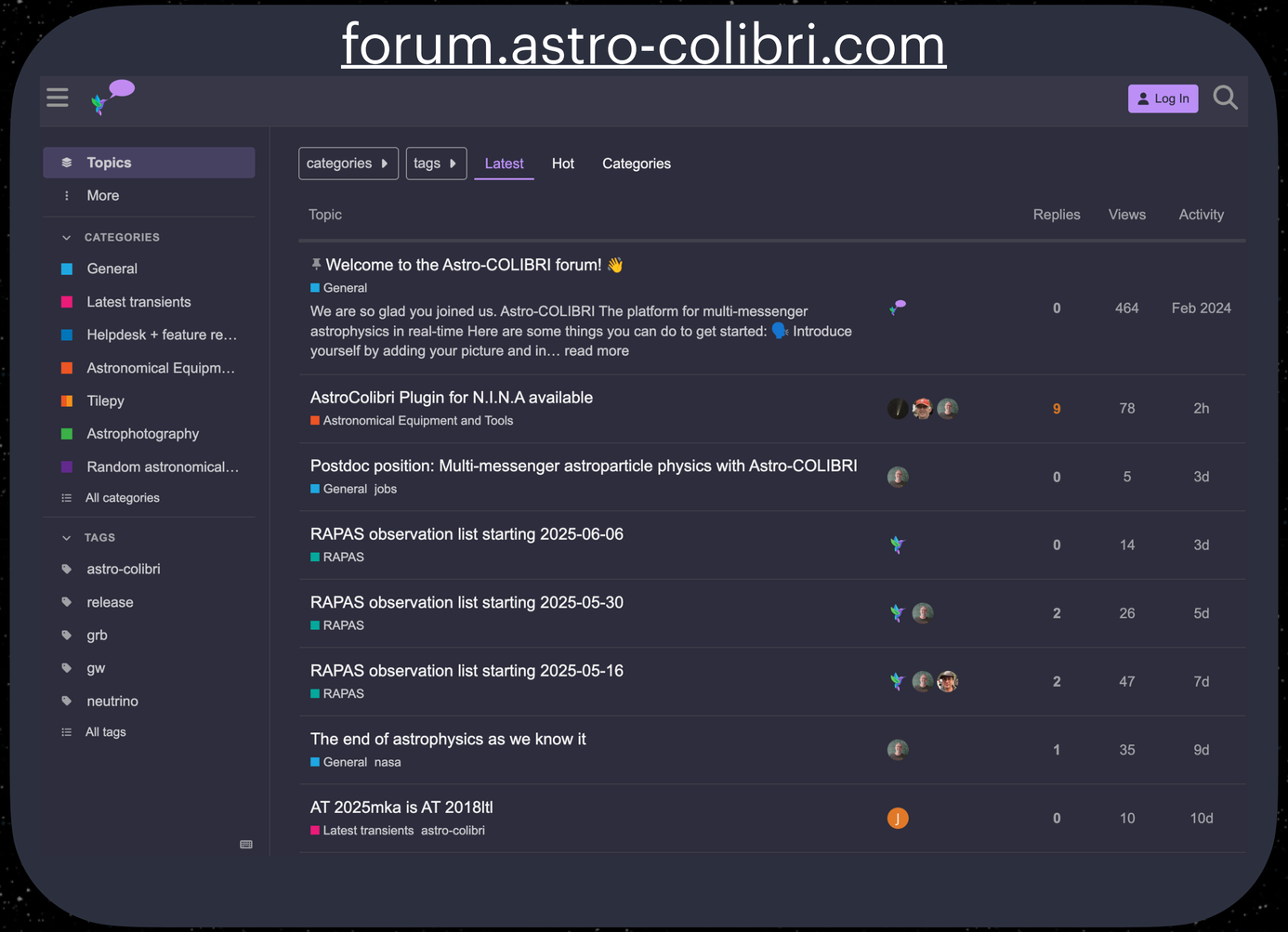
Or

Follow our tutorials on Youtube

API and Documentation

Ask any question in the Astro-COLIBRI forum or by email (astro.colibri@gmail.com)!

forum.astro-colibri.com



Try it yourself and implement Astro-COLIBRI in your devices or scripts to use it for your own interests!

We already have community driven integrations into the PRISM and NINA softwares!

Try it yourself and implement Astro-COLIBRI in your devices or scripts to use it for your own interests!

The Astro-COLIBRI team is always attentive for your needs and suggestions!

Thank you!

Merci!

Contact: astro.colibri@gmail.com