



Astro-COLIBRI - The API

COincidence LIBrary for Real-time Inquiry for multi-messenger astrophysics



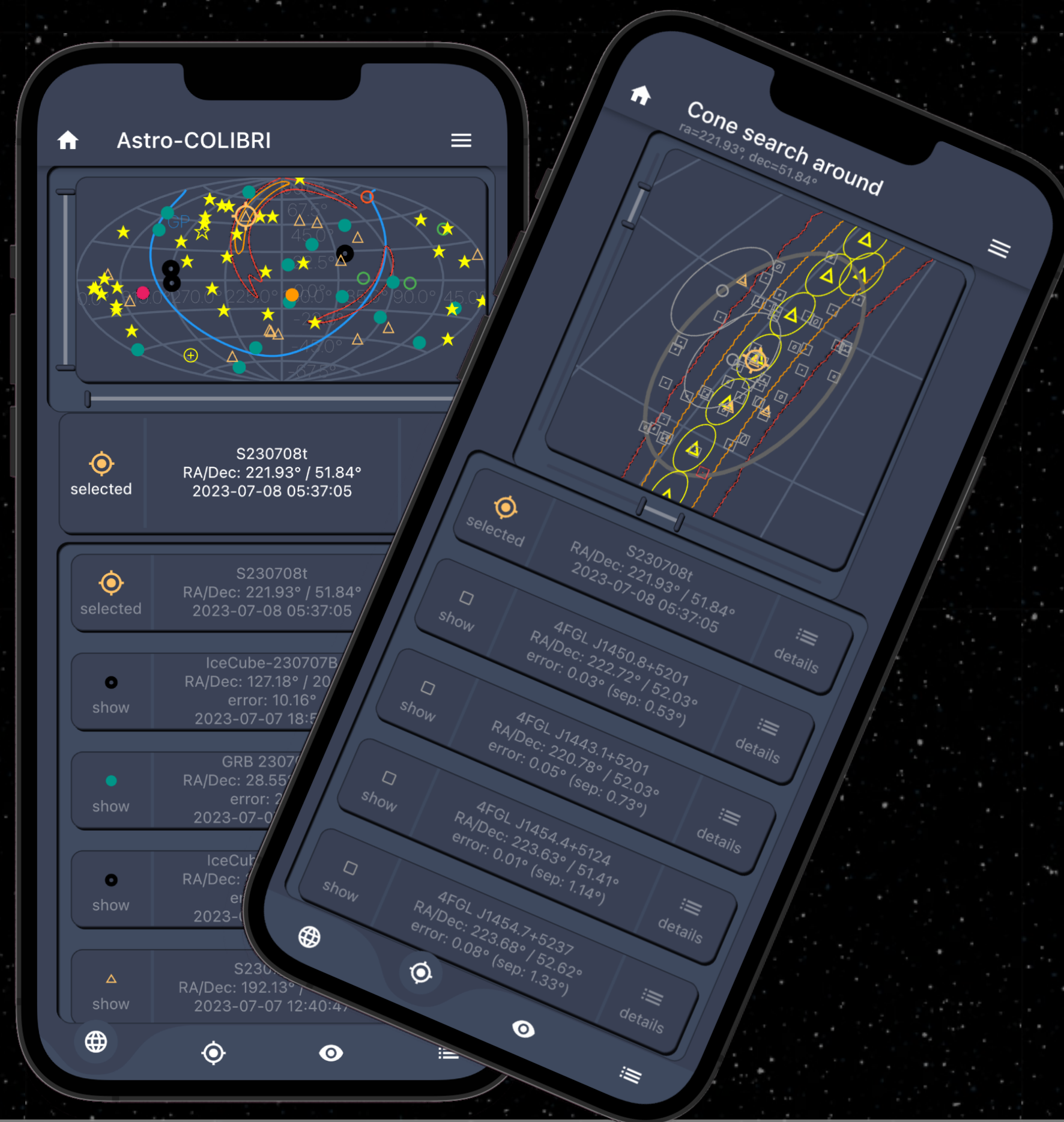
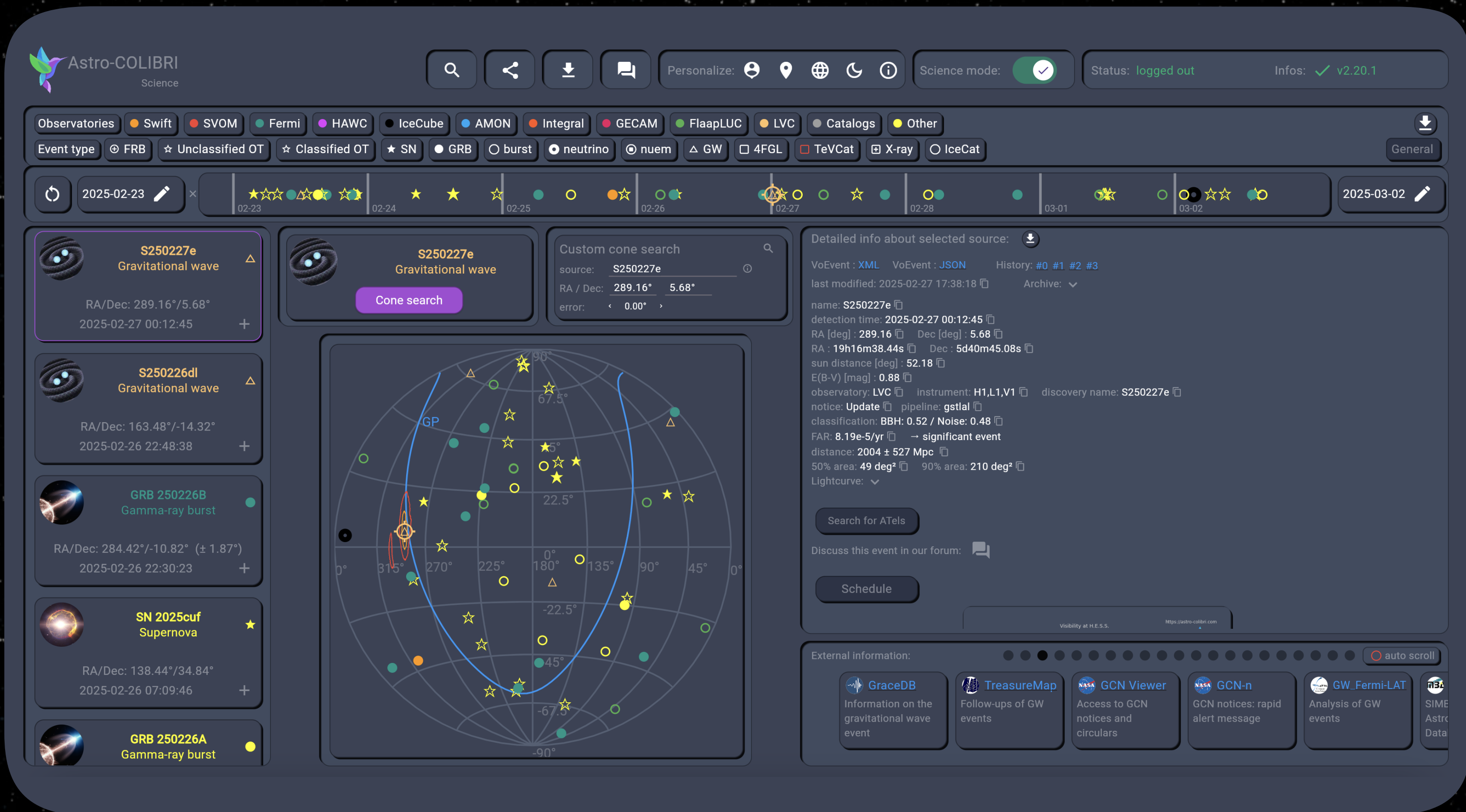
Bernardo Cornejo, Fabian Schüssler, Ilja Jaroschewski, Mickäel Costa, Weizmann Kiendrebeogo

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~~

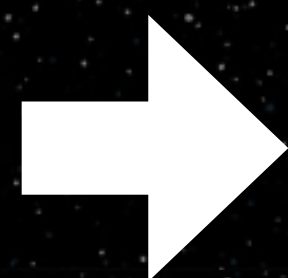


The API for real-time follow-up of
transient sources



How does the API work?

Application
Programming
Interface

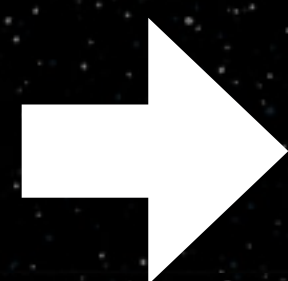


Allows you to use Astro-COLIBRI functionalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!



How does the API work?

Application
Programming
Interface



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

GET	/visibility_plots	SHOW/HIDE
GET	/visibility_plots_detailed	SHOW/HIDE
GET	/multi_observatory_visibility_plot	SHOW/HIDE
POST	/cone_search	SHOW/HIDE
POST	/latest_transients	SHOW/HIDE
GET & POST	/tevcats	SHOW/HIDE
GET & POST	/fermi	SHOW/HIDE

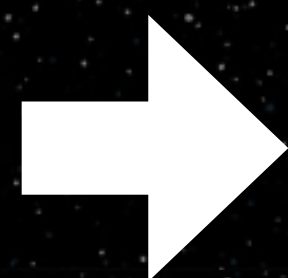
GET	/known_sources	SHOW/HIDE
GET	/celestial_position	SHOW/HIDE
GET	/atels	SHOW/HIDE
GET	/lightcurve	SHOW/HIDE
GET	/swift_xrt_lightcurve	SHOW/HIDE
GET	/get_obs_list	SHOW/HIDE

GET	/event	SHOW/HIDE
GET	/watchlist	SHOW/HIDE
GET	/archive	SHOW/HIDE
GET	/voevent	SHOW/HIDE



How does the API work?

Application
Programming
Interface



Allows you to use Astro-COLIBRI functionalities 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/tevc

```
[
  {
    "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://tevc.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
```

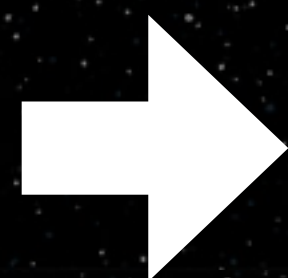



```
[1] import requests
import json
import pprint
```

How does the API work?



Application
Programming
Interface



Allows you to use Astro-COLIBRI functionalities 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/tevcats

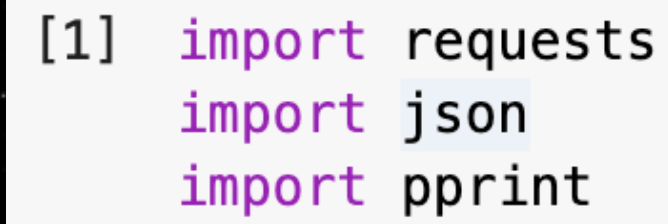
```
[3] ### You can also use this URL in directly in your code

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

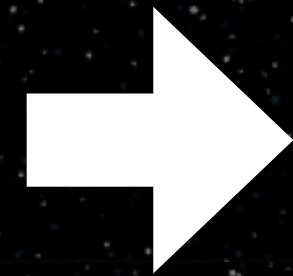
response = requests.get(url)

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://tevcats.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
```

Application Programming Interface

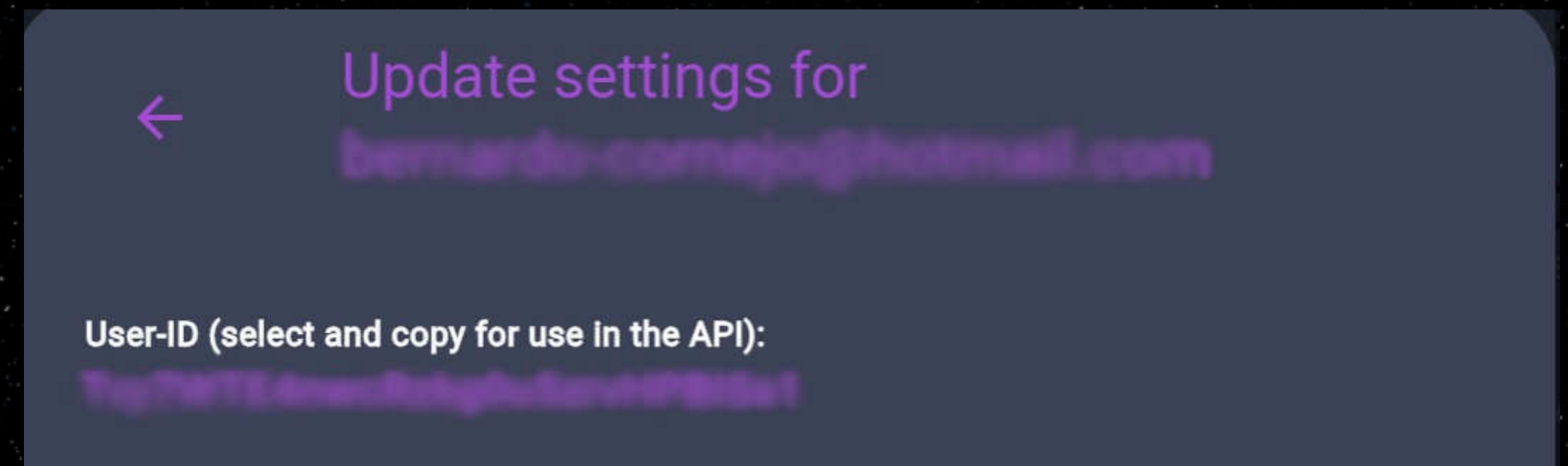


Allows you to use Astro-COLIBRI functionalities and navigate our database using simple url **endpoints** (like /cone_search) or doing a post request in your own script!

POST Request

```
[ ] uid = "0bLAEe0fGaf6QeZBpVGaB6XSYuX2"
```

c.f. Ilja's talk



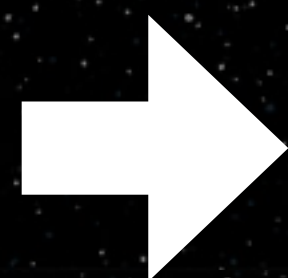


```
[1] import requests
import json
import pprint
```

How does the API work?



Application
Programming
Interface



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

POST Request

```
[ ] uid = "0bLAEe0fGaf6QeZBpVGaB6XSYuX2"
```

```
[ ] URL_TEVCAT = 'http://astro-colibri.science/tevcats'
# 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://tevcats.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
```

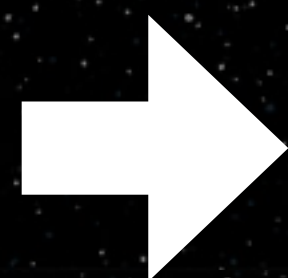



```
[1] import requests
import json
import pprint
```

How does the API work?



Application
Programming
Interface



Allows you to use Astro-COLIBRI functionalities 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/tevcats

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

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

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://tevcats.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": 201010,
  "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://tevcats.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."
}
```

POST Request

[] uid = "0bLAEe0fGaf6QeZBpVGaB6XSYuX2"

```
[ ] URL_TEVCAT = 'http://astro-colibri.science/tevcats'
# 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://tevcats.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": 201010,
  "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://tevcats.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."
}
```




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?

ASTRO-COLIBRI  [HOME](#) [API DOC](#) [TEAM](#) [Documentation](#) [FORUM](#) [Social](#) [MERCHANDISE](#) [GAMMA-CATCHER](#)

Discover the possibilities of our API by following along with our [interactive notebook demonstration](#). Give it a try!

Visibility Plots

GET	/visibility_plots	SHOW/HIDE
GET	/visibility_plots_detailed	SHOW/HIDE
	/multi-observatory-visibility-plot	SHOW/HIDE

www.astro-colibri.science/apidoc



API and Documentation

Discover the possibilities of our API by following along with our [interactive notebook demonstration](#). Give it a try!

Try it
yourself!

Plenty of
notebooks
with **detailed
examples**

co astro-colibri-use-case.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text Copy to Drive

Table of contents

Astro-COLIBRI 2.0 : API Use-case with Python

<>

{x}

Known sources

Source details

Event endpoint

VoEvent endpoint

Final remarks

+ Section

▼ Astro-COLIBRI 2.0 : API Use-case with Python

Hey there! Are you ready to dive into the exciting world of astronomical data analysis with the Astro-COLIBRI API? Great!

First things first, let's make sure we have all the necessary tools by importing the "requests" and "json" libraries. These will help us easily send requests to the Astro-COLIBRI server and understand the responses we receive. TEST

(Don't forget, the API documentation can be found at <https://astro-colibri.science/apidoc>). Feel free to modify and play with this notebook as you go along. If you want to keep your changes, you'll have to save the notebook to your own GoogleDrive area or download it as jupyter notebook.

Let's get started!

[] import requests
import json

Just a friendly reminder, all requests to the Astro-COLIBRI API are made through the following URL: <https://astro-colibri.science>.

Let's make things easier by storing this information in a variable.

[] URL = 'https://astro-colibri.science'

▼ How is a URL constructed?

In this tutorial, we will always use the same URL construction. We use the variable "URL" which stores the base of the URL and we add the parameters by concatenating a string using formatted string literals.

[] # Example
f"{URL}/myEndpoint"



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!

astro-colibri-use-case.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Copy to Drive

Table of contents

- Astro-COLIBRI 2.0 : API Use-case with Python
- How is a URL constructed?
- Function for API calls
- Known sources
- Source details
- Event endpoint
- VoEvent endpoint
- Final remarks
- + Section

Astro-COLIBRI 2.0 : API Use-case with Python

Hey there! Are you ready to dive into the exciting world of astronomical data analysis with the Astro-COLIBRI API? Great!

First things first, let's make sure we have all the necessary tools by importing the "requests" and "json" libraries. These will help us easily send requests to the Astro-COLIBRI server and understand the responses we receive. TEST

(Don't forget, the API documentation can be found at <https://astro-colibri.science/apidoc>). Feel free to modify and play with this notebook as you go along. If you want to keep your changes, you'll have to save the notebook to your own GoogleDrive area or download it as jupyter notebook.

Let's get started!

```
[ ] import requests
import json
```

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

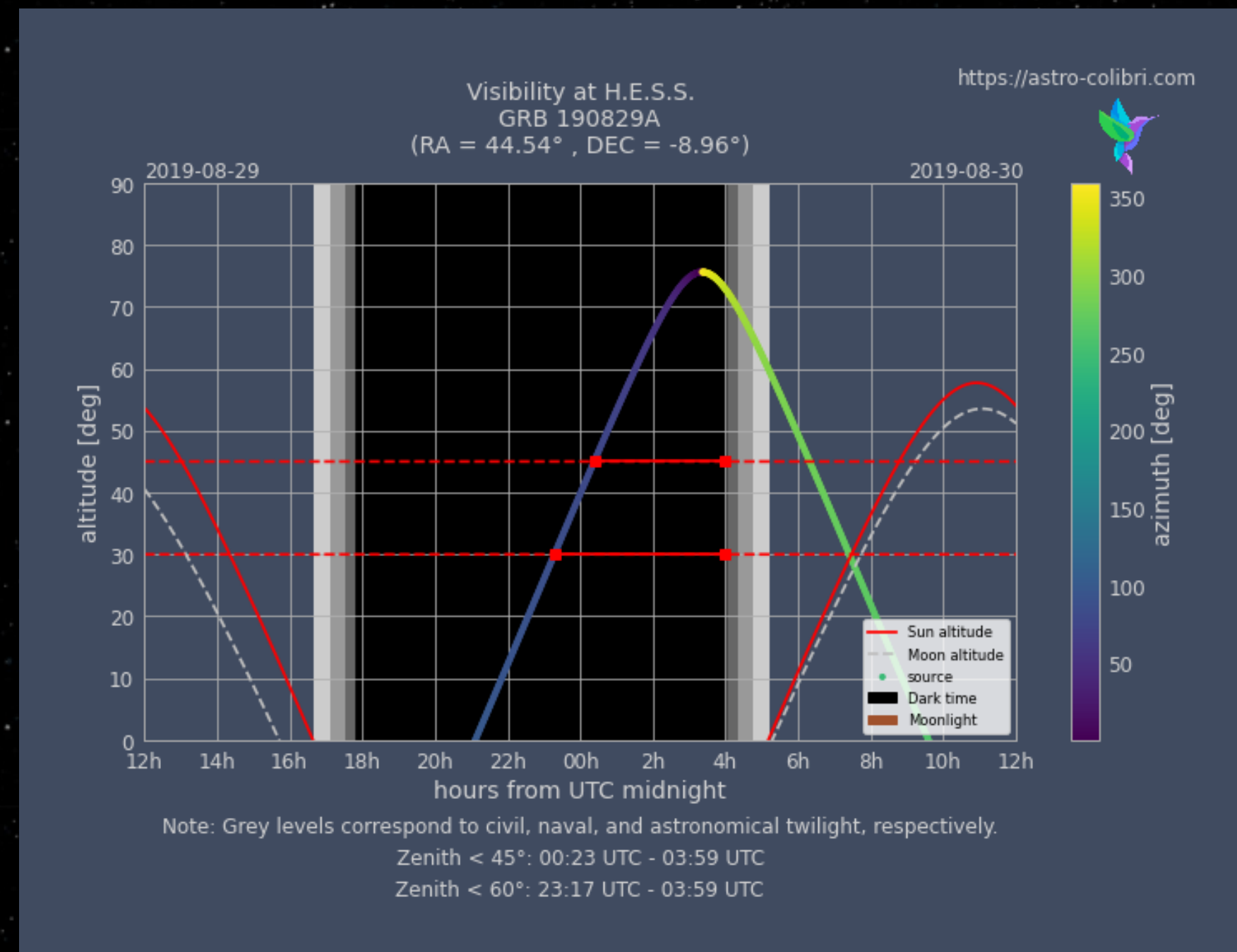
Visibility Plots

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

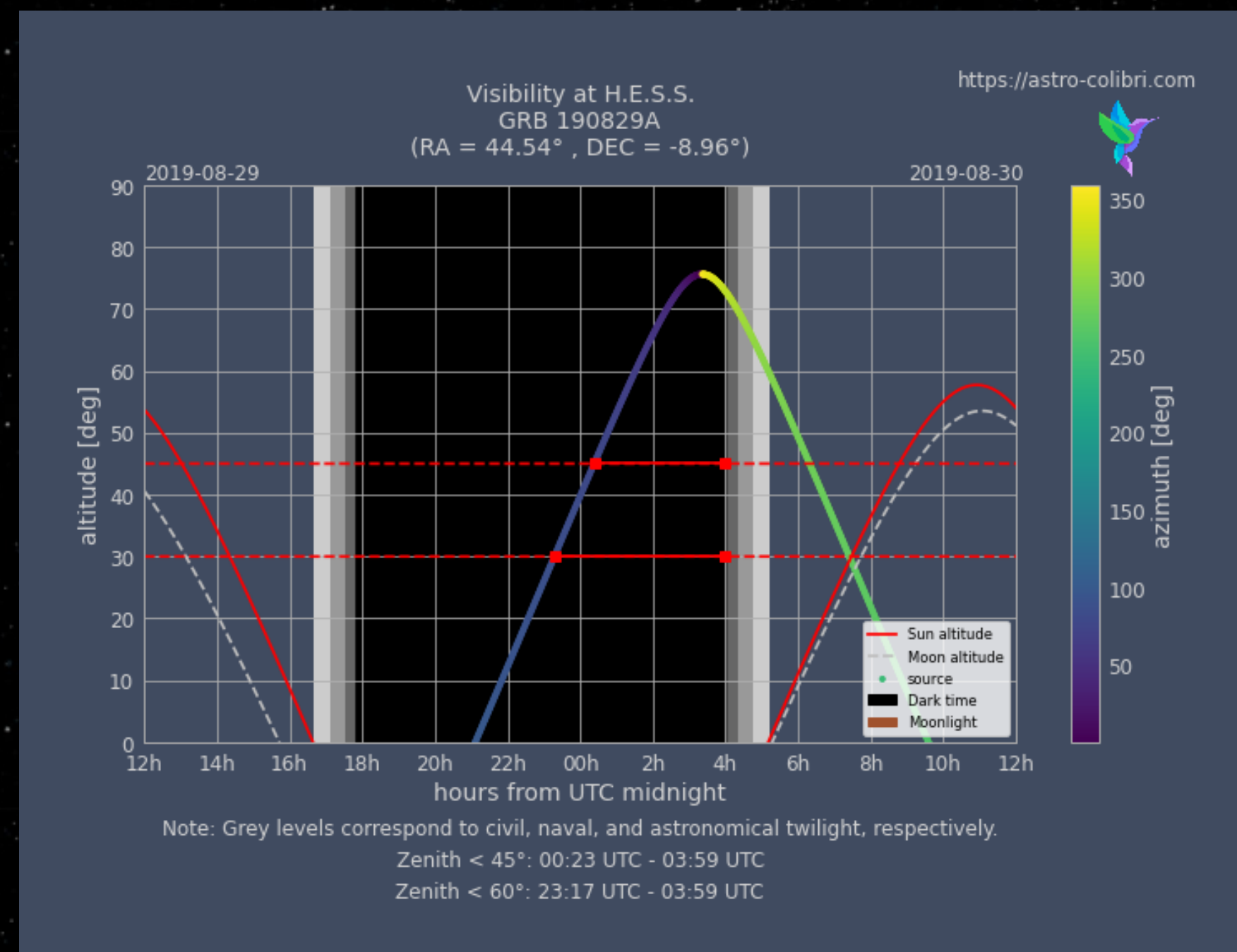
Visibility Plots

GET Requests

/visibility_plot_detailed

Query Parameters:

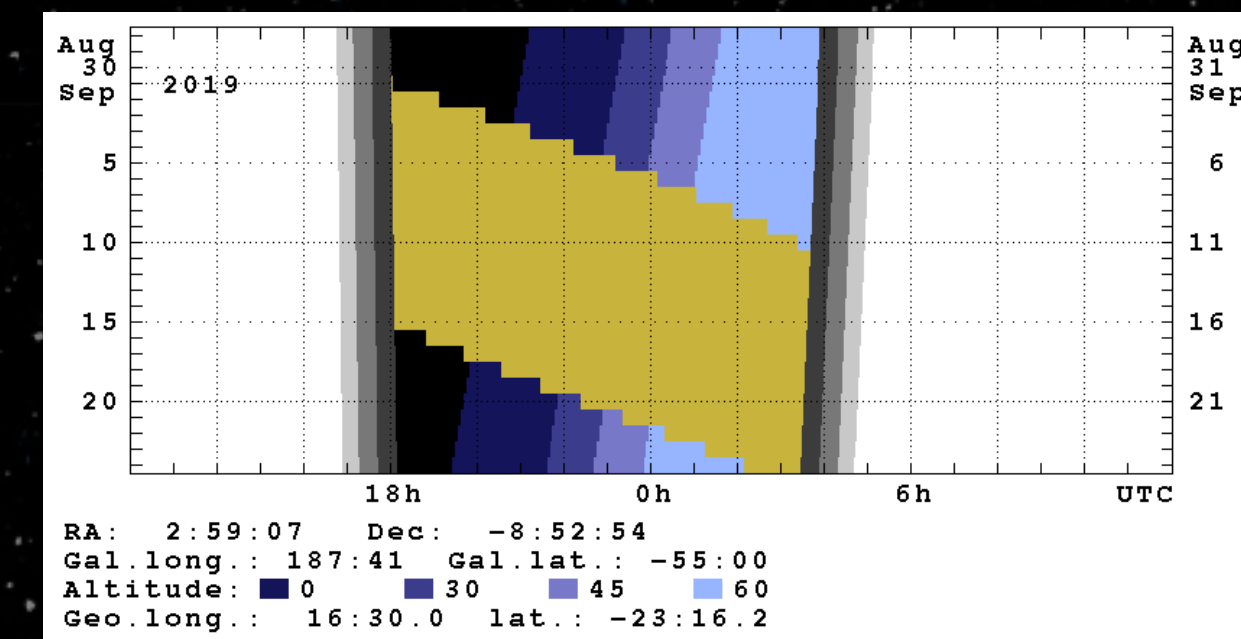
- Source position
- Observation position
- Twilight level
- Observation date
- Source name



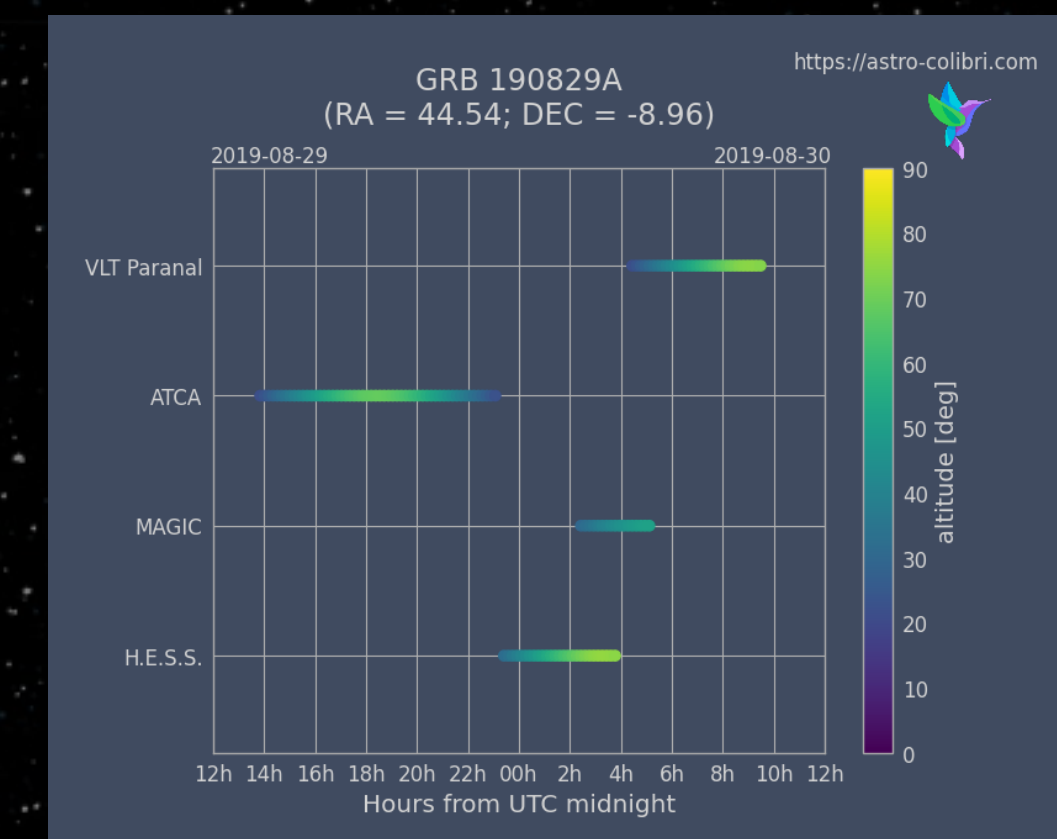
+

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



Response 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

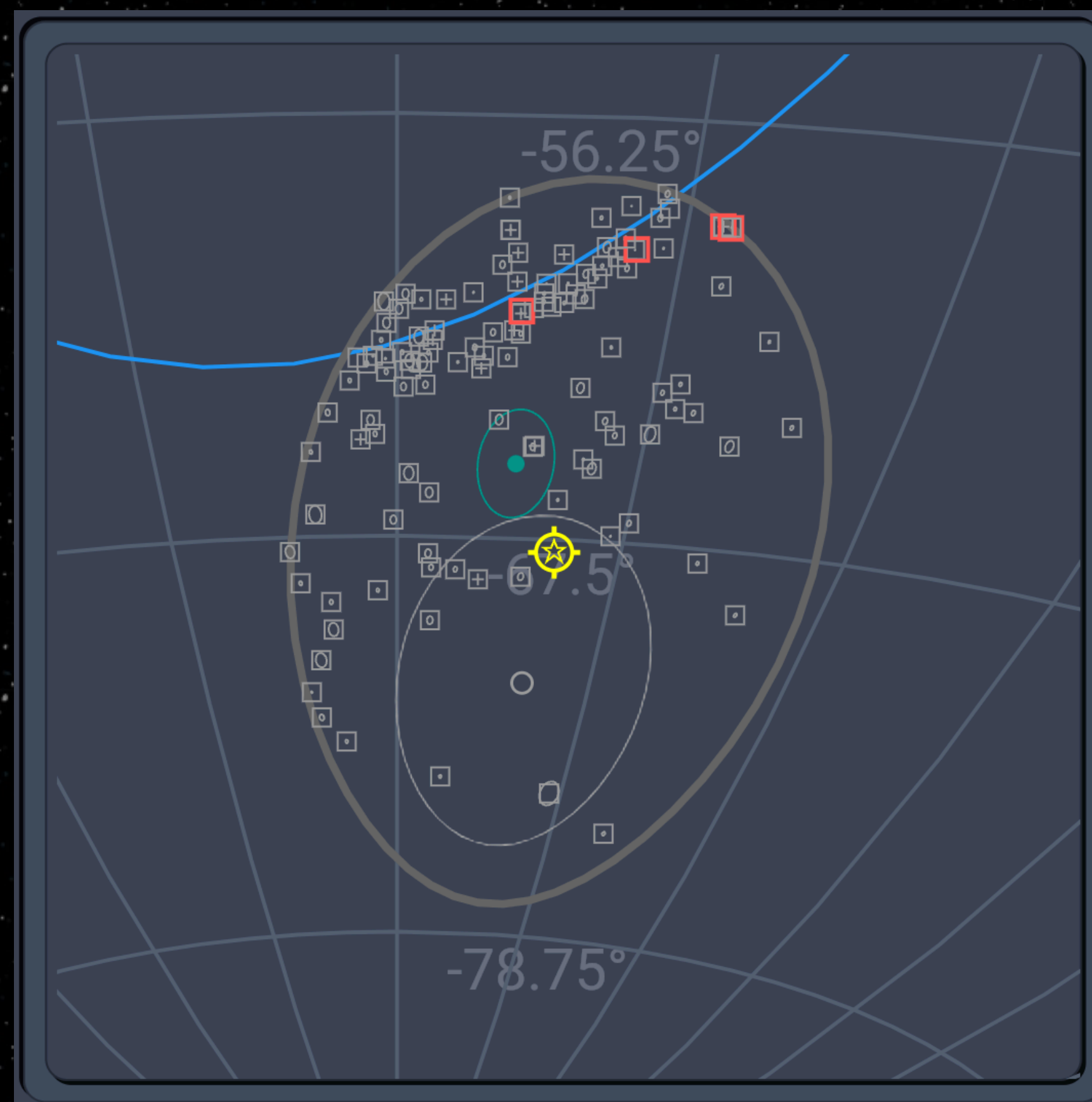
Cone search

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



```
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



```
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

/tevcats & /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...



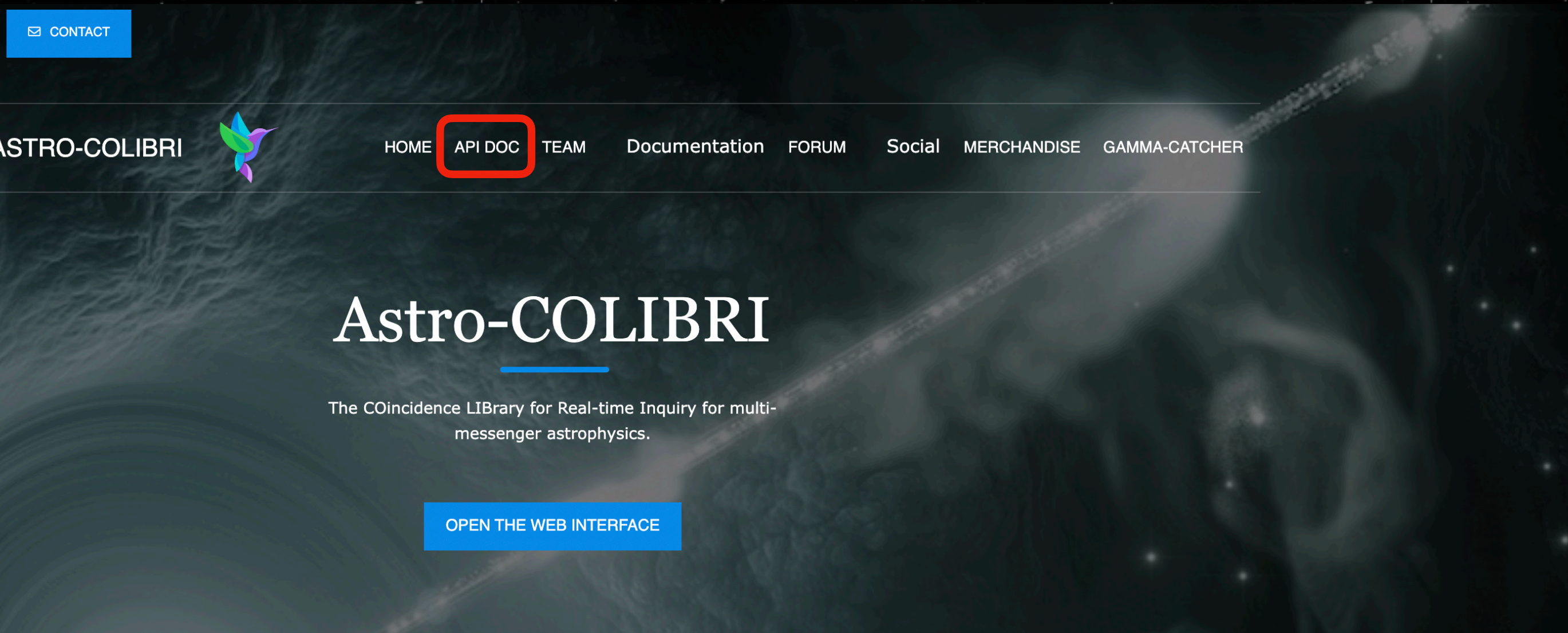
API and Documentation

www.astro-colibri.science

Read our documentation

Or

Follow our tutorials on
Youtube



What Is Astro-COLIBRI ?

A NOVEL PLATFORM FOR MULTI-MESSENGER ASTROPHYSICS

Astro-COLIBRI provides real-time insights into the most extreme astronomical phenomena, helping users quickly identify and observe events across various timescales. Our state-of-the-art architecture and user-friendly interface ensure a seamless experience for astronomers worldwide. Whether you are an experienced astronomer or simply curious about the mysteries of our universe, Astro-COLIBRI provides a powerful yet



API and Documentation

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

forum.astro-colibri.com

Topics

More

CATEGORIES

General

Latest transients

Helpdesk + feature re...

Astronomical Equipm...

Tilepy

Astrophotography

Random astronomical...

All categories

TAGS

astro-colibri

release

grb

gw

neutrino

All tags

categories tags Latest Hot Categories

Topic	Replies	Views	Activity
Welcome to the Astro-COLIBRI forum! 🙌			
General We are so glad you joined us. Astro-COLIBRI The platform for multi-messenger astrophysics in real-time Here are some things you can do to get started: 🗨️ Introduce yourself by adding your picture and in... read more	0	464	Feb 2024
AstroColibri Plugin for N.I.N.A available Astronomical Equipment and Tools	9	78	2h
Postdoc position: Multi-messenger astroparticle physics with Astro-COLIBRI General jobs	0	5	3d
RAPAS observation list starting 2025-06-06 RAPAS	0	14	3d
RAPAS observation list starting 2025-05-30 RAPAS	2	26	5d
RAPAS observation list starting 2025-05-16 RAPAS	2	47	7d
The end of astrophysics as we know it General nasa	1	35	9d
AT 2025mka is AT 2018ltl Latest transients astro-colibri	0	10	10d



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

