Interfaces Observatoire Virtuel pour PRISM et SharpCap

WIVONA: Un Projet Pro/Am soutenu par l'Observatoire de Paris-PSL

https://proam-gemini.fr/les-nouveaux-clients-de-lobservatoire-virtuel/
https://www.observatoiredeparis.psl.eu/api-collaborations-proam.html

Renaud Savalle - PADC/Observatoire de Paris-PSL
renaud.savalle@obspm.fr

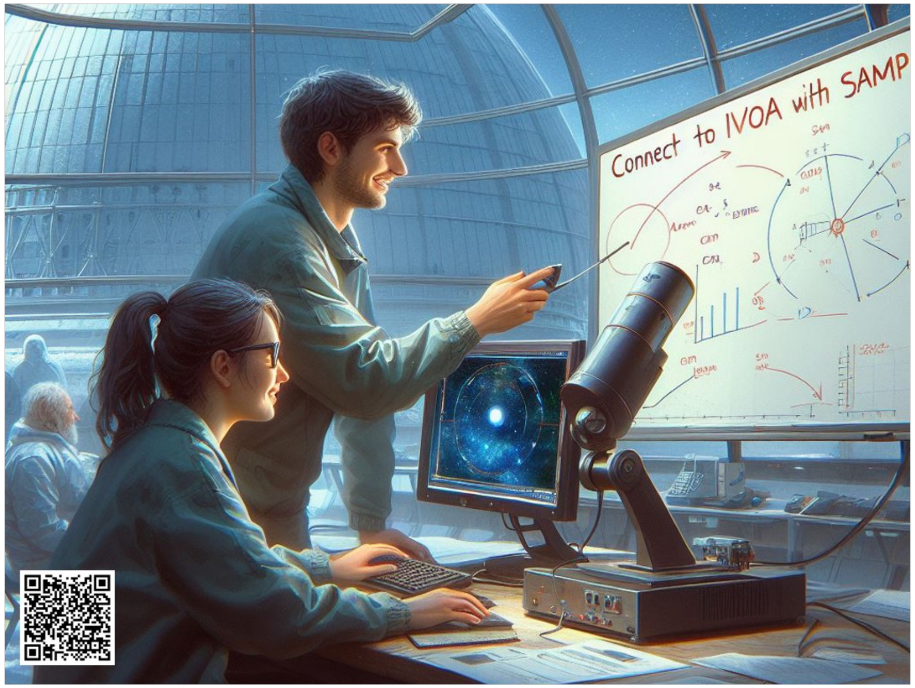
Journées SF2A – 2024-06-07 – Marseille

V2.3

WIVONA

We Implement
Virtual Observatory
Needs of Astrams

*A **Pro/Am** collaboration for the Observers Community*



- PI: **Jean-Paul GODARD**, Astronome amateur (Dev PRISM: SAMP, Astro-Colibri)
- **Renaud SAVALLE**, PADC/ Observatoire de Paris, Ingénieur de recherche CNRS, (Dev SharpCap: SAMP, Scripts Python)
- **Cyril CAVADORE**, ALCOR SYSTEM, PhD (Dev PRISM)
- **David VALLS-GABAUD**, LERMA/ Observatoire de Paris, Directeur de Recherche CNRS

X

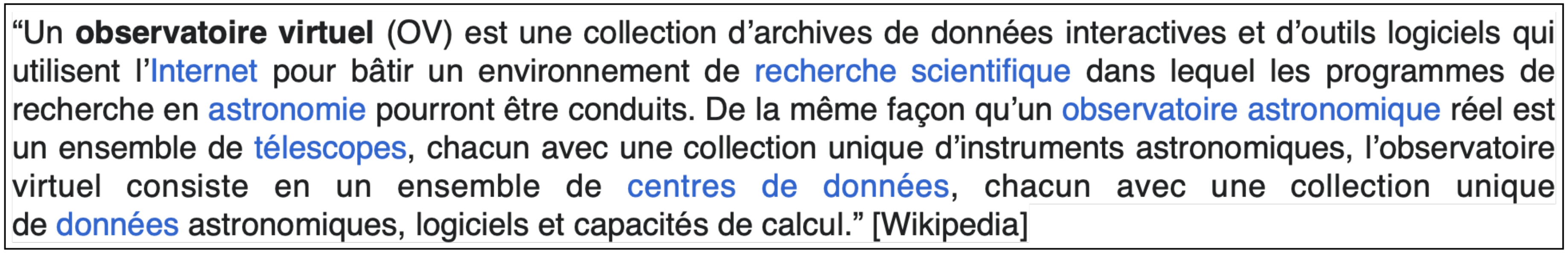
Plan

1. L'Observatoire Virtuel (OV)
2. Le protocole **SAMP** - porte d'entrée de l'OV
3. Le suivi des Alertes des Phénomènes Transitoires
4. L'accès aux données de de l'OV avec Python
5. Etat du projet

L’Observatoire Virtuel

- N’est pas:
  - un site web, ni un ensemble de sites
  - un programme

- Mais plutôt:
  - des protocoles standards de l’IVOA pour trouver, accéder, utiliser les données
  - ~50 centres de données (CDS, ESA, ESO, NASA…) dans ~20 pays
  - des opérateurs pour les services et l’infrastructure centrale (le Registre)
  - des développeurs de clients (TOPCAT, Aladin…)



“Un **observatoire virtuel** (OV) est une collection d’archives de données interactives et d’outils logiciels qui utilisent l’Internet pour bâtir un environnement de recherche scientifique dans lequel les programmes de recherche en astronomie pourront être conduits. De la même façon qu’un observatoire astronomique réel est un ensemble de télescopes, chacun avec une collection unique d’instruments astronomiques, l’observatoire virtuel consiste en un ensemble de centres de données, chacun avec une collection unique de données astronomiques, logiciels et capacités de calcul.” [Wikipedia]

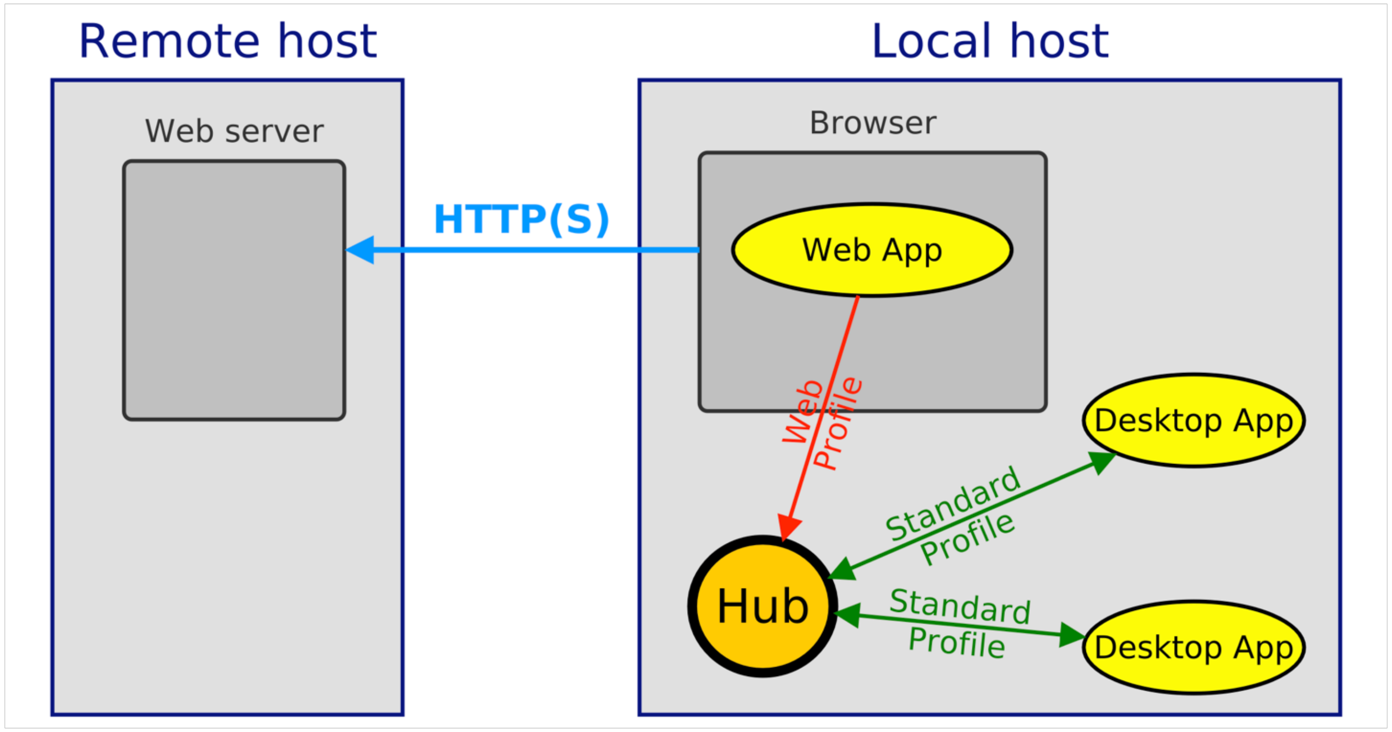
X

L’Observatoire Virtuel

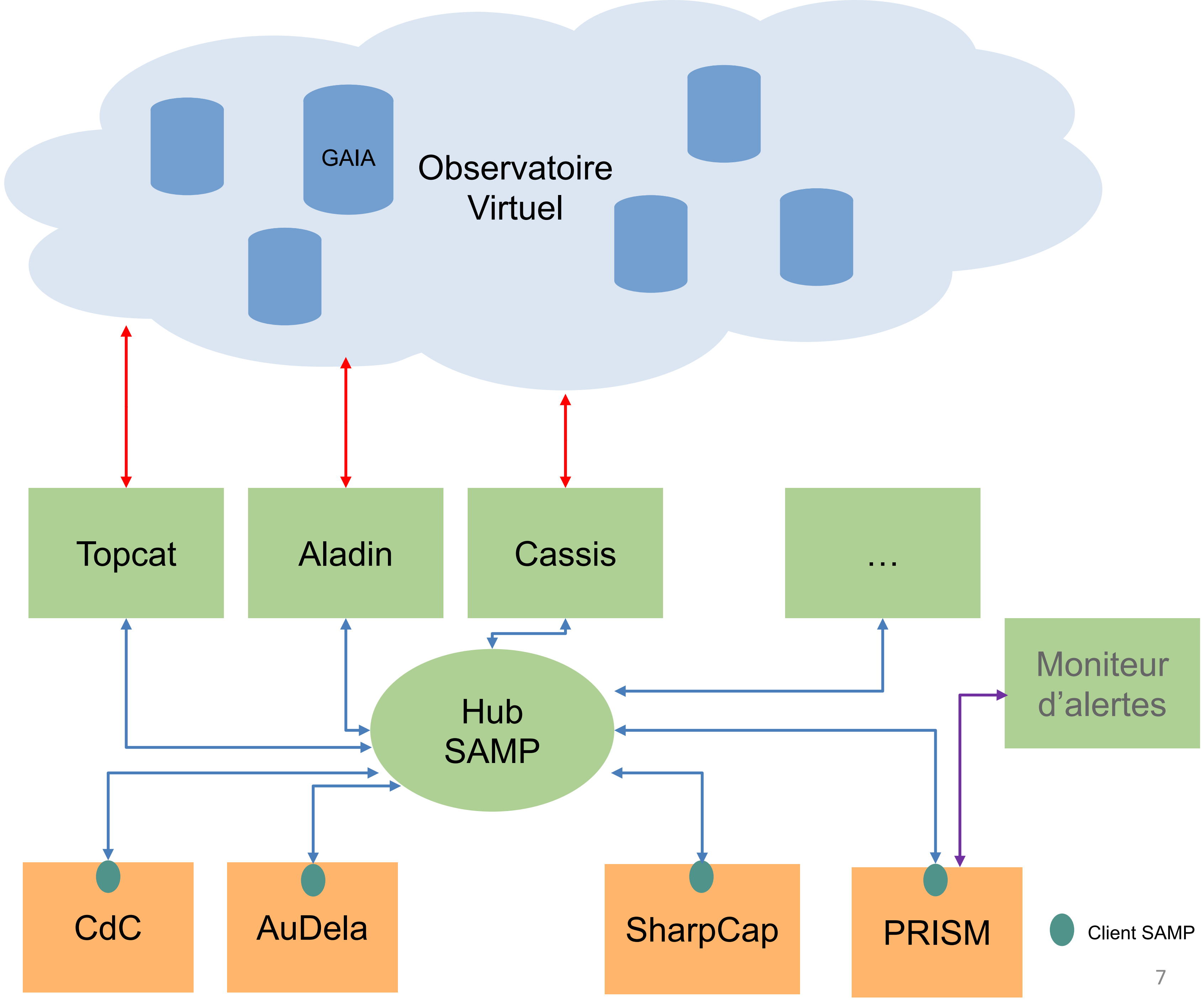
- Contient les **données** astronomiques des archives de la plupart des observatoires professionnels:
    - Catalogues avec caractéristiques des objets (position, phyique) et leur bibliographie
    - Images
    - Spectres,
    - Cubes Spectraux
    - Séries temporelles
    - Alertes (Transients)
- Des **services**: résolution de nom… et des **protocoles** d’accès normalisés aux données (Simple: SCS, SIAP, SSAP… et flexibles: TAP/ADQL)
- Pour les utilisateurs, des **applications** clientes échangeant entre elles via le protocole standard **SAMP: Simple Application Messaging Protocol**

SAMP

- **Simple Application Messenging Protocol**
- Un autre protocole “simple”:



X


GAIA
Observatoire Virtuel
Topcat
Aladin
Cassis
…
Moniteur d'alertes
Hub SAMP
CdC
AuDela
SharpCap
PRISM
Client SAMP

Échanges SAMP Hub/Client

Demandes utilisateur

- Initialisation/connexion
- Déconnexion
- Statut (Paramètres util.)
- Envoi de coordonnées
- Envoi d'Image/tables
- Envoi de VOTables
- ...

Evénements

- Evénements de service
- Arrivée coordonnées
- Arrivée images/tables
- Arrivée VOTables

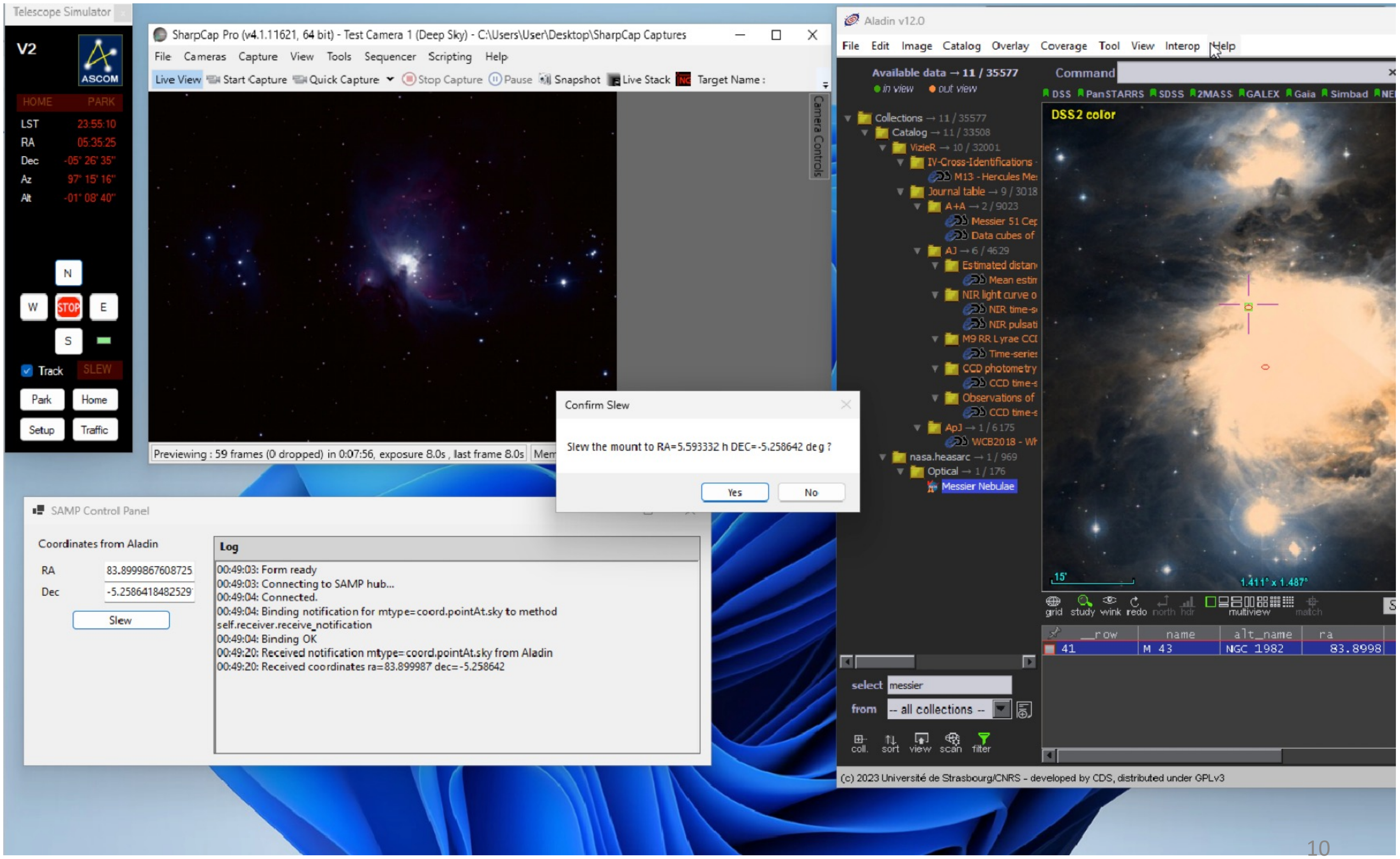
Données

- Délivre Liste Utilisateurs
- Délivre Paramètres utilisateurs

Choix d’un Hub Samp pour PRISM

- Hub SAMP intégré à PRISM :
  - Utile mais pas indispensable
- JSAMP : Hub Java externe
  - Pas d’applicatif associé
  - Fournit des outils de debug du protocole
- Aladin ou TOPCAT: Hub avec applicatif associé
  - Résolution de noms (24 697 catalogues) et arbre des resources
  - GUI sur image extraite du Survey choisie pour le champ (SDSS, 2MASS, GALEX, Planck...)
  - Réception via SAMP des coordonnées pointées par le client
- Hub de astropy.samp et scripts Python
  - Nécessite de maîtriser Python et Delphi4Python

Interlude: SAMP pour SharpCap


Telescope Simulator
V2
ASCOM
HOME PARK
LST 23:55:10
RA 05:35:25
Dec -05° 26' 35"
Az 97° 15' 16"
Alt -01° 08' 40"
N W STOP E S
Track SLEW
Park Home
Setup Traffic
SharpCap Pro (v4.1.11621, 64 bit) - Test Camera 1 (Deep Sky) - C:\Users\User\Desktop\SharpCap Captures
File Cameras Capture View Tools Sequencer Scripting Help
Live View Start Capture Quick Capture Stop Capture Pause Snapshot Live Stack Target Name :
Camera Controls
Aladin v12.0
File Edit Image Catalog Overlay Coverage Tool View Interop Help
Available data → 11 / 35577
in view out view
Command
DSS PanSTARRS SDSS 2MASS GALEX Gaia Simbad NE
DSS2 color
15'
1.411° x 1.487°
grid study wink redo north hdr multiview match
select messier
from -- all collections --
coll. sort view scan filter
(c) 2023 Université de Strasbourg/CNRS - developed by CDS, distributed under GPLv3

```
Previewing : 59 frames (0 dropped) in 0:07:56, exposure 8.0s , last frame 8.0s
```

Confirm Slew

Slew the mount to RA=5.593332 h DEC=-5.258642 deg ?

Yes No

SAMP Control Panel

Coordinates from Aladin

RA	83.8999867608725
Dec	-5.2586418482529

Slew

Log

```
00:49:03: Form ready
00:49:03: Connecting to SAMP hub...
00:49:04: Connected.
00:49:04: Binding notification for mtype=coord.pointAt.sky to method
self.receiver.receive_notification
00:49:04: Binding OK
00:49:20: Received notification mtype=coord.pointAt.sky from Aladin
00:49:20: Received coordinates ra=83.899987 dec=-5.258642
```

Collections → 11 / 35577
- Catalog → 11 / 33508
  - VizieR → 10 / 32001
    - IV-Cross-Identifications
      - M13 - Hercules Me
    - Journal table → 9 / 3018
      - A+A → 2 / 9023
        - Messier 51 Cep
        - Data cubes of
      - AJ → 6 / 4629
        - Estimated distan
          - Mean estim
        - NIR light curve o
          - NIR time-s
          - NIR pulsati
        - M9 RR Lyrae CC
          - Time-series
        - CCD photometry
          - CCD time-s
        - Observations of
          - CCD time-s
      - ApJ → 1 / 6175
        - WCB2018 - Wh
  - nasa.heasarc → 1 / 969
    - Optical → 1 / 176
      - Messier Nebulae

_row	name	alt_name	ra
41	M 43	NGC 1982	83.8998

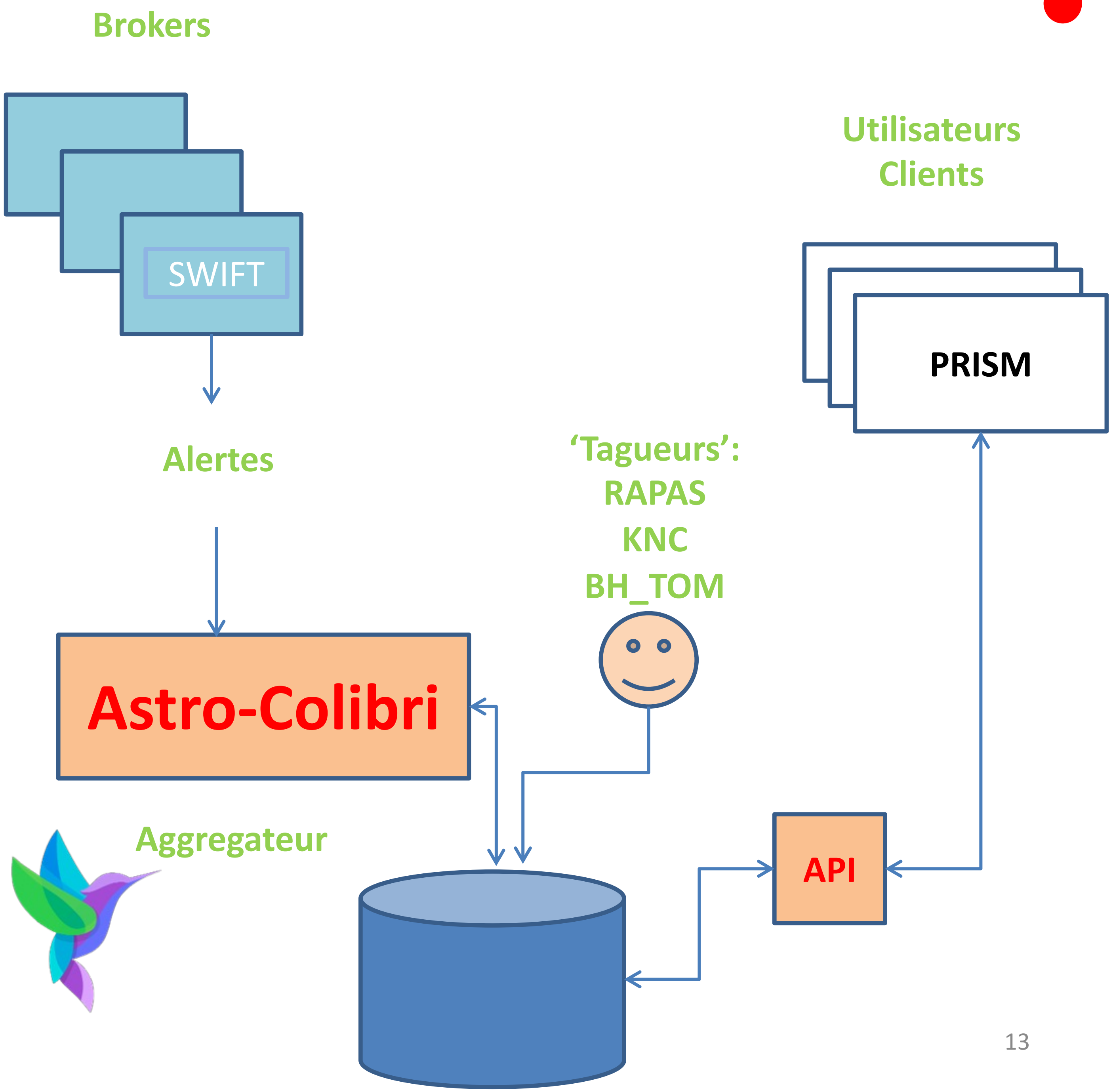
Les Transients ou Phénomènes Transitoires

- Des phénomènes fugaces
- Détectés par des observatoires automatiques
- Multi-messagers
  - Tout le spectre électromagnétique
  - Ondes Gravitationnelles (OG)
  - Neutrinos
- Les observatoires génèrent des alertes, les observateurs recherchent des contre-parties optiques.

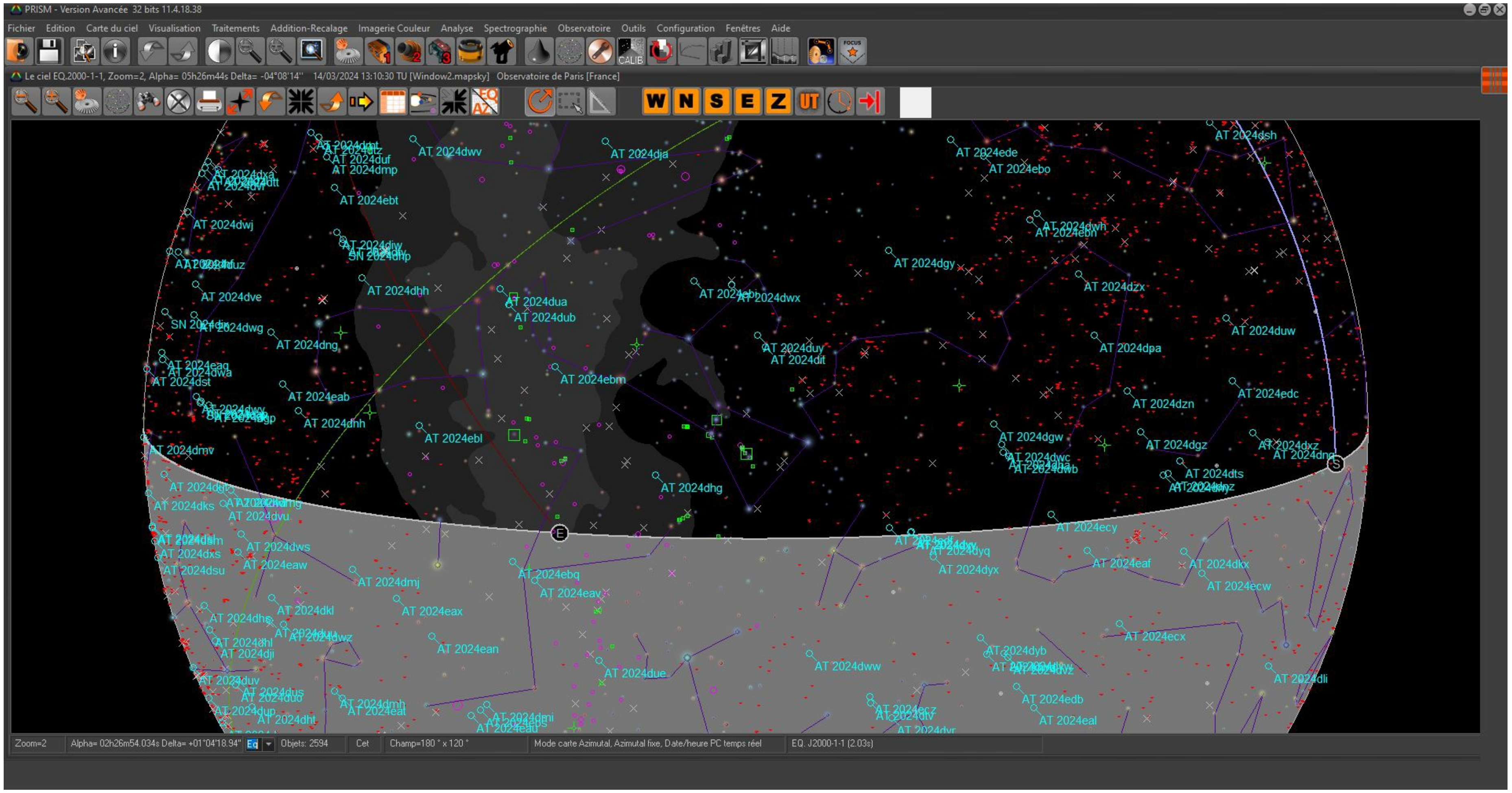
Un Moniteur d’Alertes

- Extérieur à PRISM
- Ecoute les brokers (H24,7/7) et stocke les alertes
- Fournit des filtres (type, mag) pour l’utilisateur
- Restitue des VOEvents complets
- Candidat retenu: **Astro-Colibri**
- Communication avec PRISM via API HTTP/JSON

Observatoires



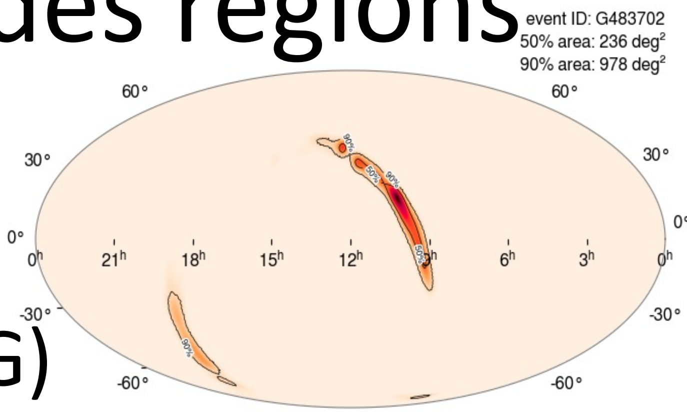
Les Transients dans PRISM



Le ciel EQ.2000-1-1, Zoom=2, Alpha= 05h26m44s Delta= -04°08'14" 14/03/2024 13:10:30 TU [Window2.mapsky] Observatoire de Paris [France]

Zoom=2 | Alpha= 02h26m54.034s Delta= +01°04'18.94" Eq | Objets: 2594 | Ciel | Champ=180 ° x 120 ° | Mode carte Azimutal, Azimutal fixe, Date/heure PC temps réel | EQ. J2000-1-1 (2.03s)

Observation de grandes régions



- Tuiles d'observation (Pour OG)
  - Astro-Colibri offre une assistance à la recherche de contreparties optiques sur de larges portions du ciel.
- Via le EndPoint /tiling de l'API
  - Le service fournit un programme d'observation couvrant la région préssentie avec une planification des observations et les paramètres de pointage.
    Cette facilité est actuellement limitée aux évènements détectés par ondes gravitationnelles (OG).
- Une interface PRISM est envisagée.

Accès a l’OV dans PRISM avec Python

- **Développement d’une console Python pour PRISM**
  - Utilisation de Python4Delphi
- **L’utilisateur peut:**
  - Charger des scripts Python
  - Les éditer, les paramétrer et les exécuter
- **Accès aux protocoles de l'OV en utilisant les modules Python**:
  - astropy
  - pyvo
  - astroquery
- … il sera bientôt possible d’utiliser Python pour **échanger des données avec les formulaires de PRISM et automatiser les observations**

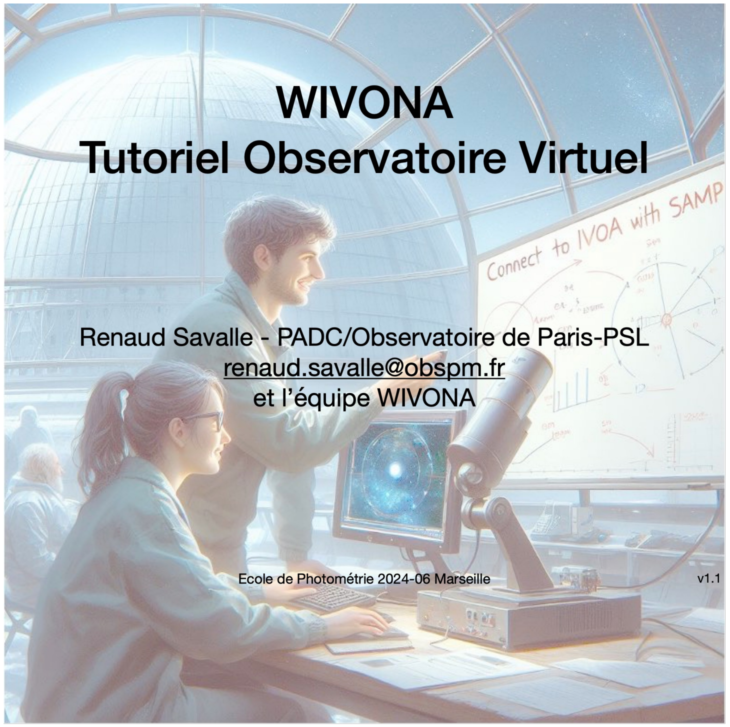
Merci aux Beta-Testeurs

- Cyril Cavadore (Intégration PRISM)
- Stéphane Charbonnel.
- Michel Meunier.
- Marc Serrau.
- …

Etat du Projet

- SAMP, Python, Astro-Colibri seront intégrés à PRISM dès la version de base
- Sources Delphi de SAMP pour PRISM bientôt accessibles via GitHub
- Sources Python (SharpCap) déjà disponibles: https://github.com/rsav/samp4sharpcap
- Démos aux Journées SF2A (cf Tutoriel demain)
- Beta-tests en cours
- Présentation prévue aux RCE2024
- Livraison prévue avec avec PRISM V12

A Suivre…


WIVONA

Tutoriel Observatoire Virtuel

Renaud Savalle - PADC/Observatoire de Paris-PSL

renaud.savalle@obspm.fr

et l'équipe WIVONA

Ecole de Photométrie 2024-06 Marseille

v1.1

Demain 2024-06-08 a 17h30 a l'Observatoire de Marseille

BACKUP SLIDES

P106

Discover your astronomical data from Python – simply!

Renaud Savalle[1], Markus Demleitner[2], Hendrik Heinl[2]

[1] DIO/Paris Astronomical Data Centre (PADC), Paris Observatory, UAR2201-CNRS, PSL Research Univ., 5, place Jules Janssen, 92195 Meudon, France
[2] Universität Heidelberg, Zentrum für Astronomie, Mönchhofstraße 12-14, 69120 Heidelberg, Germany

Contact: renaud.savalle@obspm.fr

The VO Registry is a set of about 30,000 metadata records of astronomical resources. It is queryable using the powerful RegTAP protocol requiring users to write ADQL. A friendlier interface using that protocol has recently been written a part of the PyVO astropy affiliated package. In this poster, we briefly introduce the standards this is implemented against. The new API can be used by astronomers to discover data based on various constraints, ranging from free text to physical concepts and areas in space, time, and spectrum.

The Virtual Observatory Registry

The Virtual Observatory Registry [1] implements a set of IVOA standards to query the metadata for VO data collections and services, thus making them findable. Full Searchable Registries are RegTAP [2] instances which expose that metadata in a set of tables queryable with TAP [3] and ADQL [4].

How to discover astronomical data

The standard programmatic way to discover data in the Registry is to send an ADQL query to a RegTAP instance, using the TAP protocol. This goes far beyond keyword searches; for instance, you can precisely constrain authors, subjects, physics represented in table columns, relations to other resources (such as "Continues" or "IsServedBy"), coverage in space, time, or spectrum, etc. It is also straightforward to restrict the metadata retrieved to what you actually need for the task at hand, which is relevant when a full metadata record can easily be as large as a Megabyte. To keep up with the pace of the growing data in the field, the structure of the Registry schema is evolving. On the one hand, that enables more elaborate and detailed queries, but on the other hand it makes the metadata model more complex, which also impacts the queries.

The downside of this architecture is that common free-text queries (which, of course, this architecture tries to improve upon to some degree) become rather verbose, such as this query for "Gaia lightcurves" (which is not the preferred way to discover this kind of data):

```
SELECT ivoid, res_type, short_name, res_title, res_description, reference_url
FROM rr.resource
NATURAL LEFT OUTER JOIN rr.capability
NATURAL LEFT OUTER JOIN rr.interface
WHERE (ivoid IN (SELECT ivoid FROM rr.resource WHERE 1=ivo_hasword(res_description, 'Gaia
Lightcurves') UNION SELECT ivoid FROM rr.resource WHERE 1=ivo_hasword(res_title, 'Gaia
Lightcurves') UNION SELECT ivoid FROM rr.res_subject WHERE res_subject ILIKE '%Gaia
Lightcurves%'))
GROUP BY ivoid, res_type, short_name, res_title, res_description, reference_url
```

The above query already looks surprisingly complex compared to the idea of "just" searching for a short string. Adding constraints to the query demands even more skills in TAP/ADQL and is far from a reasonable effort the typical VO end-user would want to do, especially compared to how easy all this looks from client applications like Aladin and TOCAT.

The pyvo.registry API

PyVO [5] is a an affiliated package for the astropy package which helps find and retrieve astronomical data available from archives that support standard IVOA Virtual Observatory service protocols. Since PyVO version 1.4, the module `pyvo.registry` provides a new API to Full Searchable Registries. The principle of that new API is to build discovery queries by stacking constraints such as `Freetext(keywords)`. Using PyVO 1.5, the above query looks like this:

```python
from pyvo import registry
resources = registry.search(
    registry.Freetext("Gaia Lightcurves"))
```

Under the hood, an ADQL query is generated from the given set of constraints. That query can be displayed by calling `registry.get_RegTAP_query()` instead of `registry.search()`. This is also a useful way to learn about the RegTAP relational schema.

Search Constraints

The available constraints are listed below [6]. When they accept several arguments, those are combined disjunctively ("OR"). Multiple constraints (possibly of the same type) are combined conjunctively ("AND").

- **`pyvo.registry.Freetext`** (`keywords`): one or more freetext words, mached in the title, description or subject of the resource.
- **`pyvo.registry.Servicetype`** (`servicetype`): constrain to one of tap, ssa, sia, conesearch (or full ivoids for other service types). This is the constraint you want to use for service discovery.
- **`pyvo.registry.UCD`** (`ucd`): constrain by one or more UCD patterns; resources match when they serve columns having a matching UCD (e.g., `phot.mag;em.ir.%` for "any infrared magnitude").
- **`pyvo.registry.Waveband`** (`waveband`): one or more terms from the vocabulary at http://www.ivoa.net/rdf/messenger giving the rough spectral location of the resource.
- **`pyvo.registry.Author`** (`author`): an author ("creator"). This is a single SQL pattern, and given the sloppy practices in the VO for how to write author names, you should probably generously use wildcards.
- **`pyvo.registry.Datamodel`** (`datamodel`): one of obscore, epntap, or regtap: only return TAP services having tables of this kind.
- **`pyvo.registry.Ivoid`** (`ivoid`): exactly match a single IVOA identifier (that is, in effect, the primary key in the VO).
- **`pyvo.registry.Spatial`** (`spatial`): match resources covering a certain geometry (point, circle, polygon, or MOC). *RegTAP 1.2 Extension*
- **`pyvo.registry.Spectral`** (`spectral`): match resources covering a certain part of the spectrum (usually, but not limited to, the electromagnetic spectrum). *RegTAP 1.2 Extension*
- **`pyvo.registry.Temporal`** (`temporal`): match resources covering a some point or interval in time. *RegTAP 1.2 Extension*

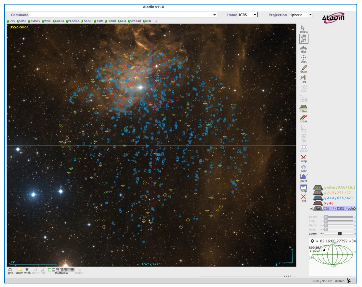
Example

```python
from pyvo import registry
from pyvo import samp
from astropy.coordinates import SkyCoord

resources = registry.search(
    registry.Freetext("Emission line stars"),
    registry.UCD("phot.mag;em.ir%"), # smart use of wildcard to match all IR bands
    registry.Spatial(SkyCoord("79d 34d")))

with samp.connection() as conn:
    for rsc in resources: # loop over the resources found
        if "conesearch" in rsc.access_modes(): # invoke the conesearch services
            objs = rsc.get_service("conesearch").search(pos=(79, 34), radius=0.5)
            if objs: # if sources are found, send them to Aladin
                samp.send_table_to(
                    conn,
                    objs.to_table(),
                    client_name="Aladin",
                    name=rsc.short_name)
```

In this example, we use `registry.search()` to look for VO resources that publish infrared magnitudes (the UCD [7] constraint) near the Flaming Star nebula in Auriga (the Spatial constraint) and mention emission line stars in their human-readable metadata. Then, connecting to Aladin via SAMP [8], we loop over the resources we have discovered. Those that have an SCS [9] endpoint we query and send whatever we get back to Aladin. It would be relatively straightforward to support other data access protocols (e.g., TAP) in this way, but of course the calling sequence would be a bit more complicated in these cases.



References

[1] M. Demleitner et al - The virtual observatory registry, Astronomy and Computing, Volumes 7–8, 2014 https://doi.org/10.1016/j.ascom.2014.07.001
[2] IVOA Registry Relational Schema standard - https://www.ivoa.net/documents/RegTAP
[3] IVOA Table Access Protocol standard - https://www.ivoa.net/documents/TAP/
[4] IVOA Astronomical Data Query Language standard - http://www.ivoa.net/documents/ADQL
[5] PyVO documentation https://pyvo.readthedocs.io/en/latest/
[6] PyVO documentation for pyvo.registry https://pyvo.readthedocs.io/en/latest/registry/index.
[7] IVOA Unified Content Descriptors standard https://ivoa.net/documents/latest/UCD.html
[8] IVOA Simple Application Messaging Protocol standard - https://www.ivoa.net/documents/SAMP
[9] IVOA Simple Cone Search standard - https://ivoa.net/documents/cover/ConeSearch-20080222.html

This research has made use of data obtained from or tools provided by Paris Astronomical Data Centre

V1.2