



**sen4cap**  
common agricultural policy

# Welcome to the 3<sup>rd</sup> IaaS Sen4CAP webinar



**The webinar will last around 1h**

**The slides will be available on the Sen4CAP website in the coming 48 hrs  
(<http://esa-sen4cap.org/>)**

## **Presenters:**

Cosmin Udroi & Laurentiu Nicola from *CS GROUP - ROMANIA*

Sophie Bontemps, Louis Descarpentries & Dries De Bièvre from *UCLouvain*

Pontus Lurcock & Norman Fomferra from *Brockmann Consult GmbH*



**Members of the consortium available to answer your questions**

ESA UNCLASSIFIED - For Official Use



European Space Agency

- **IaaS Sen4CAP overview**
- Demo
- Conclusions and next steps

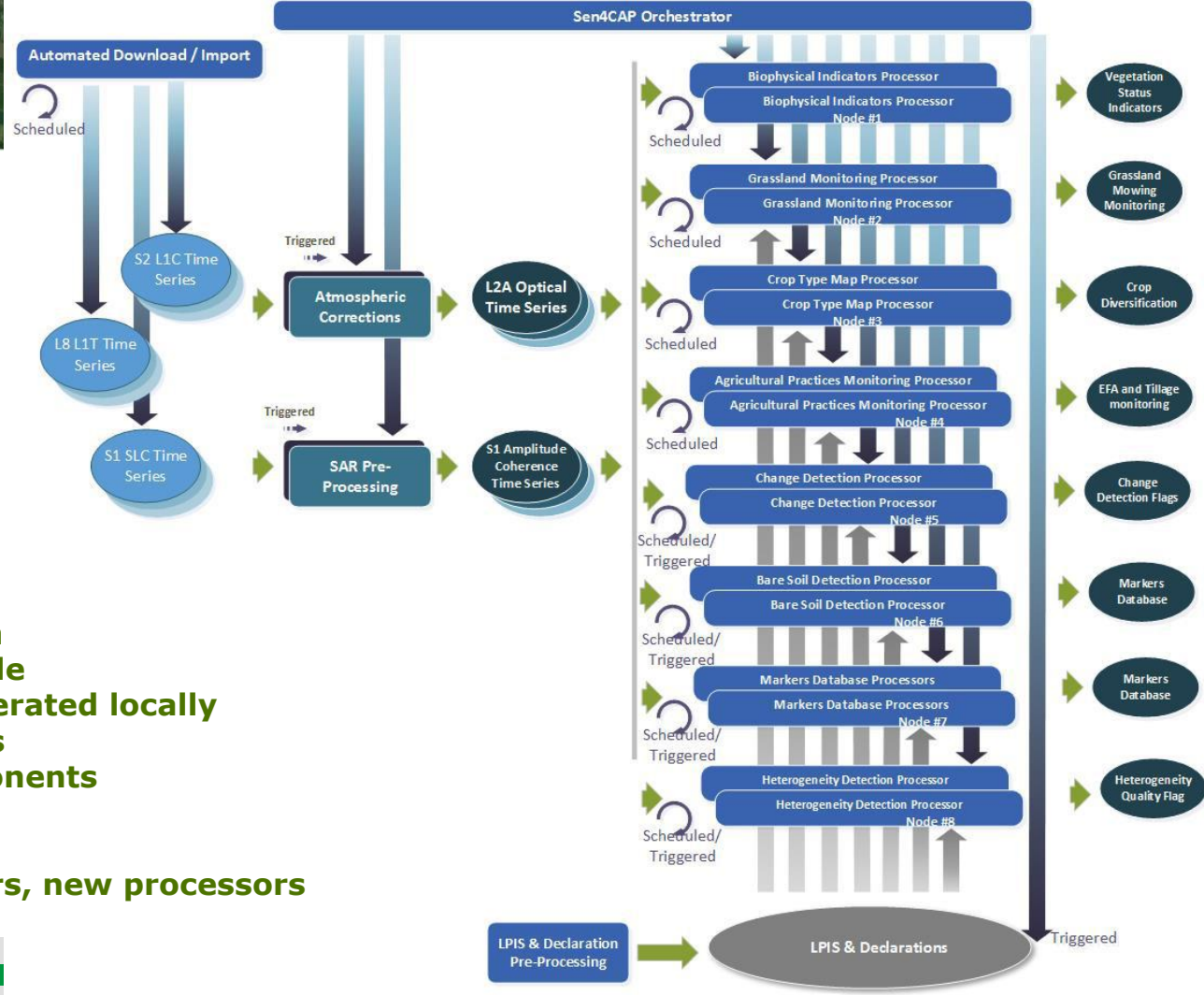
# Sen4CAP

## Open-source system

**Version 5.0 delivered during June/July**

- ❖ Sentinel-1 & -2
- ❖ Automated and modular
- ❖ For NRT or off-line production
- ❖ Demonstrated at national scale
- ❖ Portable on all DIAS-es or operated locally
- ❖ User-friendly & API interfaces
- ❖ Dockerization for main components
  
- ❖ **New Features:**
  - ❖ **S1C, Individual indicators, new processors**

ESA UNCLASSIFIED - For Official Use



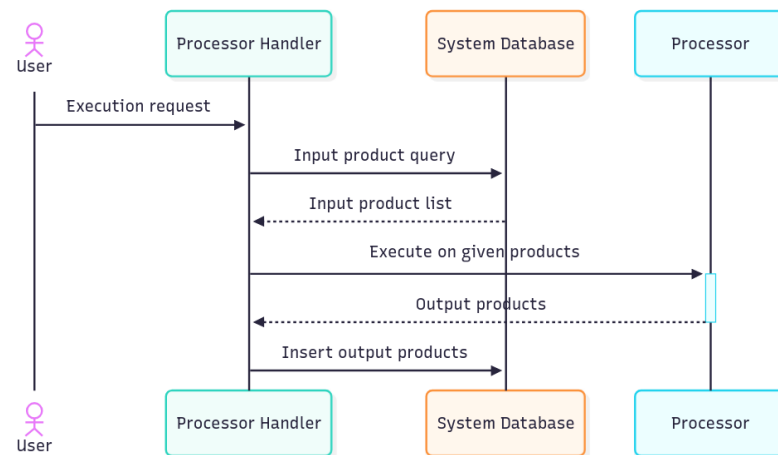
- **Sen4CAP functionality as cloud-based services**
  - Convert the core functionalities of SEN4CAP into cloud-based services
    - refactor into modular cloud-based services exposed via client libraries and API interfaces
  - User access through a Python library and standard RESTData Processing APIs (OGC Processes API standard)
  - Modular design—being able to use individual functionalities independently and decouple them for deployment, scaling and maintenance
  - Containerization and microservices architecture following OGC Application Packaging best practices
  - Allow processing on smaller areas
- **Public cloud deployment and integration**
  - Deploy the services in a public cloud environment (co-located with Sentinel data archives)
  - Onboard the services into the ESA Network of Resources (NOR) service catalogue

- **Open-Source development**

- Release the Sen4CAP code as open-source, but well-structured and fully documented
- Adhere to open-source development best practices
- Support community contributions and long-term maintenance

- **Support for STAC**

- Basically, use STAC: lightweight, extensible, popular, multiple implementations
- Keep PostGIS, ingest L2As, expose a STAC gateway (PgSTAC, stac-fastapi)
- Processor handlers can query STAC instead of PostGIS
- Managing S3 access keys for DIAS buckets (L2A) vs. custom products



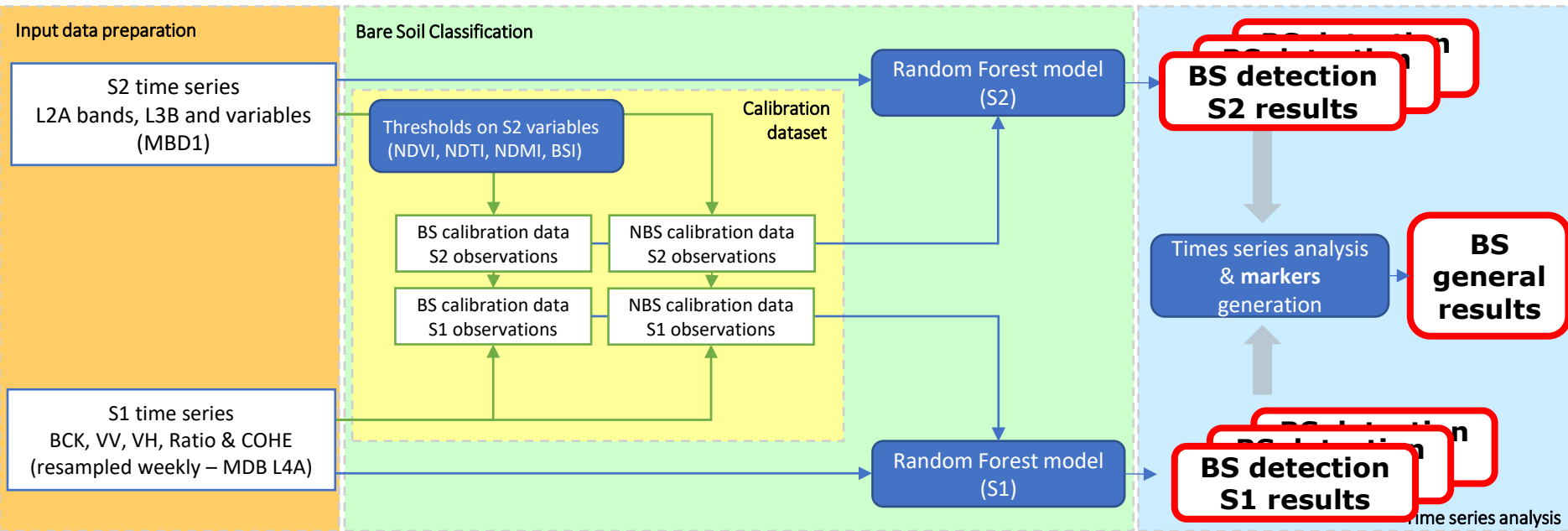
- **Porting to OGC Application Packages**

- a way to describe, package and share geospatial applications
- enables workflow and applications portability between systems / platforms
- enables integration into existing platforms and catalogs (NoR, APEX)

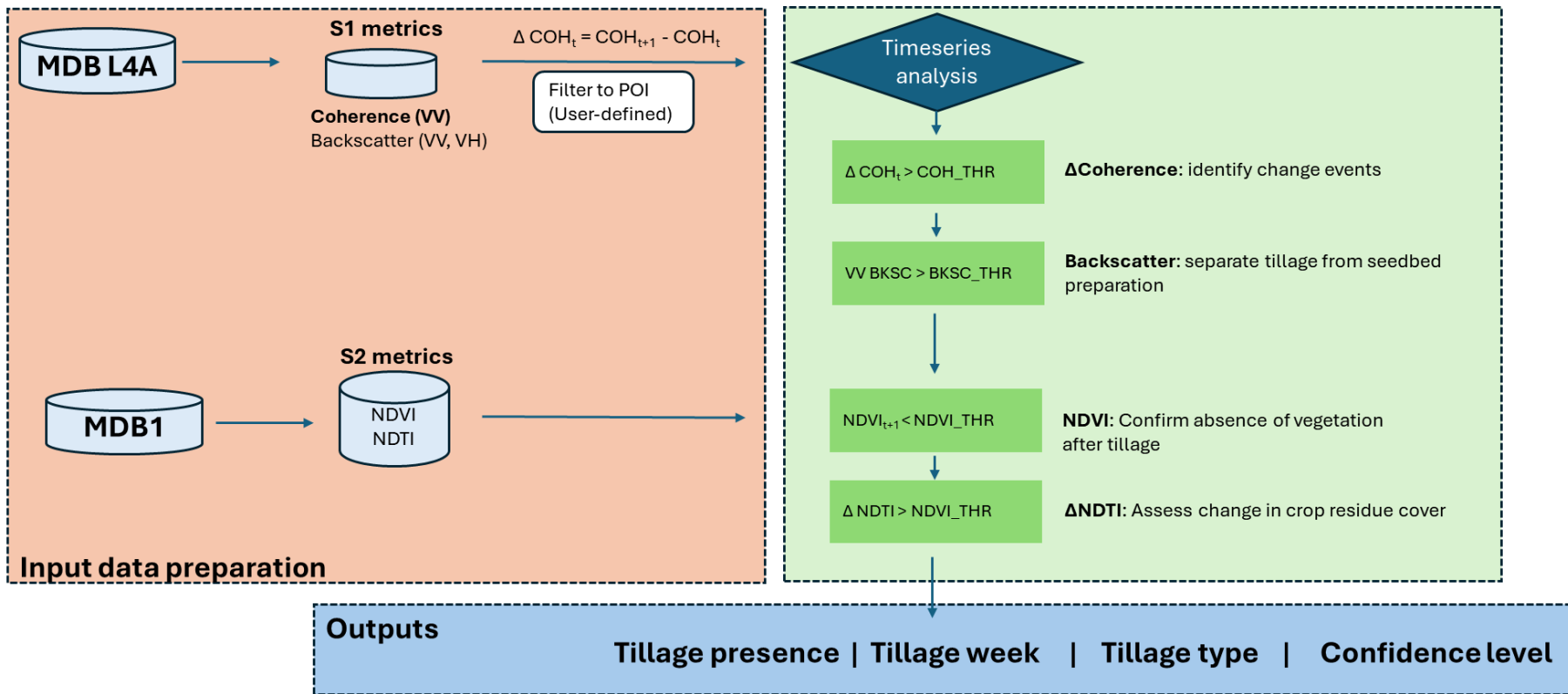
- **Application structure**

- packaged as OCI images
- Common Workflow Language / CommandLineTool descriptors
- command-line tools

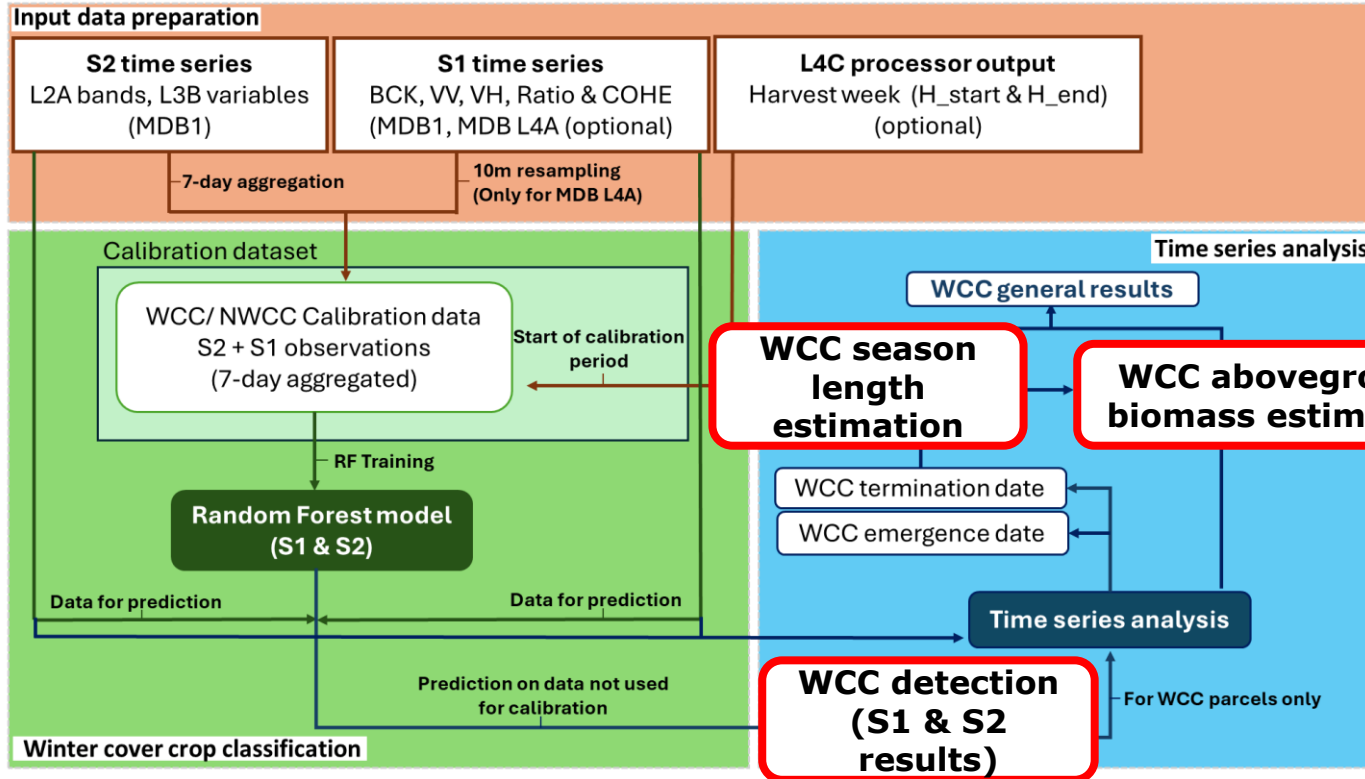
# Scientific Enhancements Bare Soil processor



# Scientific Enhancements Tillage processor



# Scientific Enhancements Winter cover crop (WCC) processor



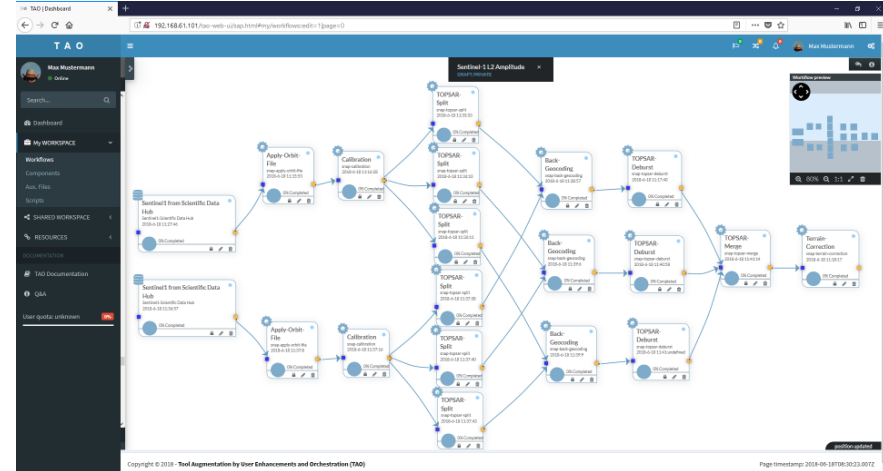
Determined based on user survey responses



# TAO – Tool Augmentation by user enhancements and Orchestration



- **TAO** - integration framework for the Sen4CAP processors
  - ✓ orchestration of heterogeneous processing components and libraries to process data
  - ✓ Preparation of resources (including processing components or tools) and data input
  - ✓ Create and save pipelines as execution workflows
  - ✓ Distributed and scaling workflow executions (up from one to as many nodes as made available)
  - ✓ Retrieval / visualization of the results
  - ✓ User management
  - ✓ Multi-user platform, authentication (local user base, LDAP, Keycloak or even single sign-on), user groups
  - ✓ Support for OpenStack API and Kubernetes for vertical and horizontal execution scaling
  - ✓ Export workflows in different formats



# TAO Strenghts

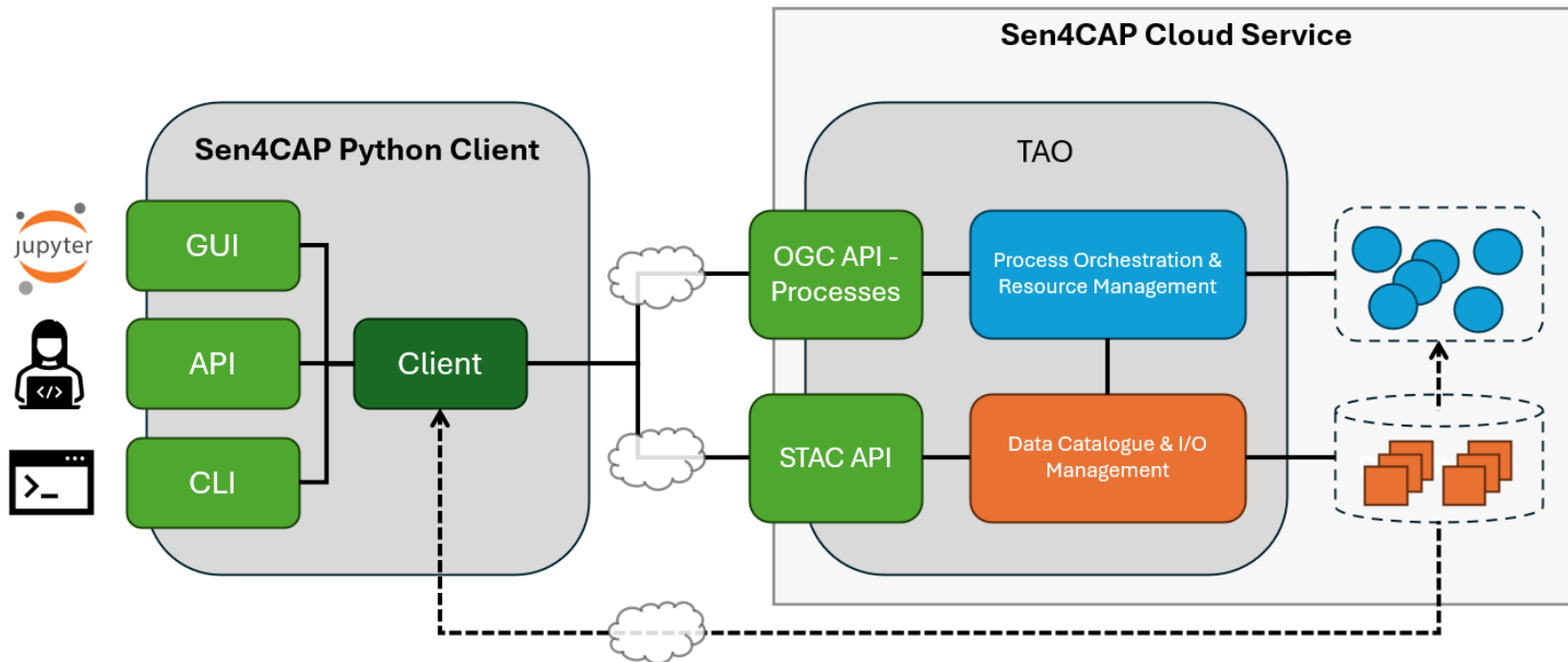


- Integrate hundreds or thousands of tools and data sources
  - ✓ Grouping by category, tagging, filtering, searching tools
  - ✓ Building easily workflows
- Can integrate any tool that can be invoked from the command line
- Services invocation can be performed by adding script wrappers that perform invocation
- Export workflows into other formats (CWL, Argo, shell etc.).
- Exposes a OGC API Processes interface
- Support for S3, STAC and other repository types
- Integration with AI agents

The screenshot displays the TAO user interface. At the top, the user 'Udroiu Cosmin' is logged in as 'ADMIN'. The main content area is titled 'Processing Modules' and lists various tools. A red circle highlights the text 'Processing modules - 9 to 16 of 514'. Two modules are visible: 'BINARYMORPHOLOGICALOPERATION' and 'BLOCKMATCHING'. Below this, a 'Script workflow' dialog is open, showing the 'EMD\_JOT' workflow selected for scripting. The dialog includes fields for 'Execution environment' (DEFAULT), 'Label' (EMD\_JOT\_17-06-2025\_14-47), and 'Execution name' (EMD\_JOT\_17/06/2025 14:47:03). A 'Script type' dropdown menu is open, showing options: 'Script as JSON', 'Script as CWL', 'Script as Bash script', and 'Script as ARGO script'. A 'Filter CSV' button is also visible.



# Extended architecture using Python API



- Client functionality
  - List processes, get process details
  - Submit processing request
  - List jobs, get job details, cancel job
  - Get job results
- Functionality accessible through 3 user interfaces
  - API: for Python scripts and using in own libraries
  - GUI: for convenient use in Jupyter notebooks
  - CLI: for the terminal and in shell scripts
- Implementation assumes a back-end web service compliant to OGC API - Processes - Part 1: Core

# Python API in Jupyter Lab



```
client-api.ipynb
Python 3 (ipykernel)

[5]: client.get_processes()

[5]: ProcessList object:
      links [] 1 item
      processes [] 4 items

[6]: client.get_process(process_id="sleep_a_while")

[6]: ProcessDescription object:
      description "Sleeps for `duration` seconds. Fails on purpose if `fail` is `True`. Returns the effective amount of sleep in seconds."
      id "sleep_a_while"
      inputs
      duration
      fail
      outputs
      title "Sleep Processor"
      version "0.0.0"

[8]: client.execute_process(process_id="sleep_a_while", request=ProcessRequest())

[8]: JobInfo object:

[13]: client.get_jobs()

[13]: JobList object:
      jobs [] 3 items
      0
      created "2025-07-15T14:30:55.953473"
      finished "2025-07-15T14:31:06.177857"
      jobID "job_0"
      processID "sleep_a_while"
      progress 100
      started "2025-07-15T14:30:55.954652"
```



# Processing GUI in Jupyter Lab



client-gui-show.ipynb

version "0.0.0"

inputs

outputs

```
[5]: cClient.show()
```

```
[5]: Process  
simulate_scene
```

Generate scene for testing

Simulate a set scene images slices for testing. Creates an xarray dataset with `periodicity` time slices and writes it as Zarr into a temporary location. Requires installed `dash`, `xarray`, and `zarr` packages.

Variable names

a, b, c

Selected bbox: [5.160938, 50.884045, 15.092578, 55.710155]

Spatial resolution: 0.5

Start date: 2025-01-01

End date: 2025-02-01

Periodicity: 1

Output path:

Execute Open Save Save As... Get Request

client-gui-show.ipynb

```
[5]: Process  
primes_between
```

Prime Processor

Returns the list of prime numbers between a `min_val` and `max_val`.

Min Val: 0

Max Val: 100

Execute Open Save Save As... Get Request

Done

Process ID: primes\_between Created: 2025-07-17 16:46:54.087104

Job ID: job\_3 Started: 2025-07-17 16:46:54.087562

Status: JobStatus.successful Updated: 2025-07-17 16:46:54.087633

Progress: - Finished: 2025-07-17 16:46:54.087646

```
[6]: cClient.show_jobs()
```

Process ID	Job ID	Status	Progress	Message
primes_between	job_0	successful		Done
sleep_a_while	job_1	running		-
sleep_a_while	job_2	running		-
primes_between	job_3	successful		Done

Cancel Delete Restart Get Results

Stored results of job (job\_3) in variable `_results`

```
[7]: _results
```

```
[7]: Results:  
return_value / 25 items  
0 2  
1 3  
2 5  
3 7  
4 11
```



# Processing CLI in terminal emulator



```
Usage: sen4cap-client [OPTIONS] COMMAND [ARGS]...
```

```
Client tool for the Sen4CAP service.
```

```
The tool provides commands for managing processing request templates, processing requests, processing jobs, and gets processing results.
```

```
You can use shorter command name aliases, e.g., use command name "vr" for "validate-request", or "lp" for "list-processes".
```

## Options

<b>--verbose</b>	<b>-v</b>	Verbose output
<b>--install-completion</b>		Install completion for the current shell.
<b>--show-completion</b>		Show completion for the current shell, to copy it or customize the installation.
<b>--help</b>		Show this message and exit.

## Commands

<b>version</b>	Show version and exit.
<b>configure</b>	Configure the client.
<b>list-processes</b>	List available processes.
<b>get-process</b>	Get process details.
<b>validate-request</b>	Validate a processing request.
<b>execute-process</b>	Execute a process.
<b>list-jobs</b>	List all jobs.
<b>get-job</b>	Get job details.
<b>dismiss-job</b>	Cancel a running or delete a finished job.
<b>get-job-results</b>	Get job results.

# Sen4CAP Processors in APEx and NoR



- APEx

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_s1\\_pre\\_processing](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_s1_pre_processing)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l3b\\_biophysical\\_indicators](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l3b_biophysical_indicators)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l3b\\_spectral\\_indices](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l3b_spectral_indices)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l3\\_basic\\_markers](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l3_basic_markers)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l4a\\_crop\\_type](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l4a_crop_type)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l4b\\_grassland\\_mowing](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l4b_grassland_mowing)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l4c\\_agricultural\\_practices](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l4c_agricultural_practices)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l4d\\_bare\\_soil](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l4d_bare_soil)

[https://algorithm-catalogue.apex.esa.int/apps/sen4cap\\_l4e\\_parcel\\_heterogeneity](https://algorithm-catalogue.apex.esa.int/apps/sen4cap_l4e_parcel_heterogeneity)

- NoR

Registered as DPaaS and as User Algorithm Hosting

<https://portfolio.nor-discover.org/>



- IaaS Sen4CAP overview

- **Demo**

**TAO and use of AI in building workflows**

**Python client library**

**APEX and NoR**

- Conclusions and next steps

- **Forum** for your questions and exchanges about the Sen4CAP system
- **If you want access to test and evaluate the new Sen4CAP Services:**  
**URL:**  
**[sen4x.tao.c-s.ro](https://sen4x.tao.c-s.ro)**  
**Contact:**  
**[cosmin.udroi@cs-soprasteria.com](mailto:cosmin.udroi@cs-soprasteria.com)**
- **Your questions ???**

**Thank you for your attention  
and your contribution**



**sen4cap**  
common agricultural policy