Sen4CAP - Sentinels for Common Agricultural Policy

System Software User Manual

Milestone	Milestone 2
Authors	Sophie BONTEMPS, Katja BAJEC, Cosmin CARA, Pierre DEFOURNY, Laura DE VENDICTIS, Diane HEYMANS, Lubos KUCERA, Philippe MALCORPS, Grega MILCINSKI, Laurentiu NICOLA, Jana SLACIKOVA, Matthieu TAYMANS, Florin TUTUNARU, Cosmin UDROIU
Distribution	ESA - Benjamin KOETZ

This page is intentionally left blank

Table of contents

List of figures

	Ref	Sen4CAP_SUM_v2.0		
	Issue	Page	Date	
	2.0	8	07/01/2021	

List of tables

1. Introduction

1.1 Purpose and scope

This document is the Software User Manual (SUM) document of the Sentinels for Common Agriculture Policy (Sen4CAP) project funded by the European Space Agency (ESA).

The overall objective for the Sen4CAP project is to provide the European and national stakeholders of the Common Agriculture Policy (CAP) validated algorithms, products, workflows and best practices for agriculture monitoring relevant for the management of the CAP. Special attention shall be given to provide evidence how Sentinel derived information can support the modernization and simplification of the CAP in the post 2020 timeframe.

The SUM is one output of the Task 5 (WP 5000) of the Sen4CAP project, named "System Development". The aim of this document is to detail the operational system implementation, build, installation and maintenance procedure.

1.2 Structure of the document

After this introduction, this document contains **4** main sections and the appendices:

- Section **2** provides a general overview of the system: its main objectives and properties, its processors and its architecture;

- Section **3** describes the deployment procedures of the system, both for the automatic and manual modes;

- Section **4** explains how to use the system, detailing the first steps to allow an automatic use but also the customization of the jobs for the manual operation;

- Section **5** defines the maintenance and troubleshooting procedures;

- Section **6** presents the visualization tool and its deployment;

- 6 appendices are provided at the end of the document.

1.3 Reference & Applicable documents

Table 1-1. Applicable documents

ID	Title	Code	Issue	Date
AD.1	Statement of Work for ESA Sentinels for the common agricultural policy	EOEP-EOPS-SW-17-015	1.0	15/03/2017
AD.2	Sen4CAP Technical Proposal - Chapter 2		1.0	12/05/2017
AD.3	Sen4CAP Implementation Proposal - Chapter 4		1.0	12/05/2017
AD.4	Sen4CAP User Requirement Document	Sen4CAP_URD_1.3	1.3	9/11/2017
AD.5	Sen4CAP Test Data Set document	Sen4CAP_TDS_1.0	1.0	29/03/2018
AD.6	Sen4CAP Technical Specification document	Sen4CAP_TS_1.0	1.0	29/03/2018

AD.7	Sen4CAP Software Development Plan	Sen4CAP_SDP_1.0	1.0	14/02/2018
AD.8	Sen4CAP Design Justification File	Sen4CAP_ DJF_1.0	1.0	29/03/2018
AD.9	Sen4CAP Service Definition File	Sen4CAP_ SDF_1.0	1.0	29/03/2018

Table 1-2. Reference documents

ID	Title
RD.1	Weiss M. and Baret F.. 2016. S2 Toolbox Level 2 products: LAI, FAPAR, FCOVER, V1.1.
RD.2	User, installation and operating manual MAJA. Available at: https://github.com/CNES/Start-MAJA/blob/master/Readme.md

1.4 Acronyms

Table 1-3. Acronyms

Acronym	Definition
API	Application Platform Interface
ATBD	Algorithm Theoretical Basis Document
AWS	Amazon Web Services
CAP	Common Agricultural Policy
CNES	Centre National d'Etudes Spatiales
DDF	Design Definition File
DIAS	Data and Information Access Services
EO	Earth Observation
ESA	European Space Agency
GIS	Geographical Information System
GSAA	GeoSpatial Aid Application
GUI	Graphical User Interface
IACS	Integrated Administration and Control System
ID	Identifier
Inter-Process Communication	Inter-Process Communication
IPT	Innovative Platform Testbed
L1, L1C, L1T	Level 1, Level 1C, Level 1T
L2, L2A	Level 2, Level 2A
L3, L4	Level 3, Level 4
L8	Landsat 8

LPIS	Land Parcel Identification System
LUT	Look-Up-Table
MACCS	Multisensor Atmospheric Correction and Cloud-Screening
MAJA	MACCS-ATCOR Joint Algorithm
OGC	Open Geospatial Consortium
PA	Paying Agency
PEPS	Plateforme d'Exploitation des Produits Sentinel
REST	REpresentational State Transfer
ROI	Region Of Interest
S1, S1A, S1B	Sentinel-1 (A and B)
S2, S2A, S2B	Sentinel-2 (A and B)
SDF	Service Definition File
Sen2-Agri	Sentinel-2 for Agriculture
Sen4CAP	Sentinels for Common Agricultural Policy
SLC	Single Look Complex
SoW	Statement of Work
UCLouvain	Université catholique de Louvain
UI	User Interface
WMS	Web Mapping Service

2. System Overview

2.1 General overview

The Sen4CAP system consist of an Earth Observation (EO) processing system linked with a visualization tool. It is open-source, allowing any user to generate, at his own premises and in an operational way, products tailored to his needs.

The Sen4CAP EO operational system is a standalone processing chain which generates a set of products for agriculture monitoring from Sentinel-2 (S2) L1C, Sentinel 1 (S1) Single Look Complex (SLC) and Landsat 8 (L8) L1T time series. These agriculture monitoring products consist of:

- biophysical vegetation status indicators (Normalized Difference Vegetation Index (NDVI), Leaf Area Index (LAI), Fraction-Absorbed Photosynthetically Active Radiation (fAPAR) and the Fraction of Vegetation Cover (FCover)), describing at the pixel-level the vegetative development of crops;

- metrics and crop markers extracted by parcel, processed along the agricultural season and stored in a database;

- cultivated crop type maps at parcel-level, generated several times during the agricultural season;

- grassland mowing detection products at parcel-level, generated at regular intervals during the agricultural season;

- agricultural practices monitoring product at parcel-level, generated at regular intervals during the agricultural season. Since the version 2.0 of the Sen4CAP system, this product integrates the detection of the tillage practice.

The Sen4CAP EO system is composed of a set of independent processing modules orchestrated by a data-driven approach. These modules are composed by a set of tools which can be re-used into other systems. The logical data flow and the main interfaces of the Sen4CAP operational system is provided in Figure 2-1.

Figure 2-1. Logical data flow of the Sen4CAP EO processing system and its link with the visualization tool

The main and mandatory inputs of the operational system are:

- A time series of S1 SLC and S2 L1C products automatically downloaded from ESA facility or imported directly into the system if they are available locally as in the case of the Copernicus Data and Information Access Services (DIAS-es);

- Mandatory user parameters: Area of Interest defined as a shapefile, definition of the monitoring period, a set of parameters associated with the algorithms (see section 4);

- Subsidy applications layer (generally based on the Land Parcel Identification System (LPIS) or GeoSpatial Aid Application (GSAA)) for each country.

Additional inputs can be provided by the user if available and/or if found relevant. This is the case of the L8 L1T products, but also of other products and parameters that are detailed in section 4.

A Graphical User Interface (GUI) is available to define and handle all the requested parameters. This interface also controls the start of the system, indicates the main directories or files used by the system and allows users visualizing and downloading output products.

The Sen4CAP system was designed to be modular and interactive. It includes two main components:

- A set of Sen4CAP Software Components (SC): each Sen4CAP SC is an independent executable that represents an algorithm or a set of algorithms;

- A Sen4CAP Orchestrator: the Orchestrator is the main component used to manage the above Sen4CAP SC on the system. Its role is to monitor the occurrence of new files in the processing area file system, determine the processing chain to be launched, handle user requests and monitor the execution of the current processing chains.

There are two ways of using the system:

- The *automated mode*, which would be used in operational scenarios with as little as possible operator intervention. It is based on the Orchestrator and it ensures that the system automatically downloads data, processes them until the end of the season and delivers in time the output products. In this functioning mode, the processors can also be executed on user request, by using the public website functionality, which is part of the platform. The website can be reached using a web browser and accessing http://[machine_name_or_ip]/main.php, where [machine_name_or_ip] is the server name or pc.

- The *manual mode*, which offers the possibility of a manual processing, independently for any processor, without installing and configuring the whole system. In this case, only the processors (which are mainly OrfeoToolbox (OTB) applications), and optionally S2-TB / Sentinel Application Platform (SNAP) application (for visual execution) are necessary.

The modular design and standardization of the interfaces make the Sen4CAP architecture targeted at various user profiles:

- **High-level analysts**, who want to integrate the products into their agricultural monitoring analyses and are interested in a direct delivery of the products generated by the Sen4CAP system;

- **Operator level users**, who aim at running the system and generating the Sen4CAP products in their own facilities to control and validate the results, then disseminate them to their own high-level users;

- **Research users**, who would rather access and tune the components available, to generate new or adjusted products and possibly operate them as "classical" added-value producers.

The system can either be built from sources, available on a git repository, or installed from precompiled binaries packaged into Redhat Package Manager (RPM) binaries. "Appendix D - Building from source" presents the steps to be followed in order to build the RPM binaries. Section 0 describes the installation and configuration of the obtained RPM binaries (either from sources or from a distribution package). If any of the components' binaries are available or installed, the corresponding step will be skipped.

The Sen4CAP visualization tool (i.e. Geopedia) is a Software as a Service based WebGIS application, which has been configured and upgraded to demonstrate how Sen4CAP products can be integrated into existing standard Geographical Information System (GIS) environment of the Paying Agency (PA) using standard protocols. It provides an option to analyze the Sentinel-derived markers and products generated within the project, together with the input datasets provided by the PAs. In general, we expect PAs to use their own GIS environments as there are not many special functions needed. Still, the package for user interface is provided for the demonstration purposes.

2.2 Processors

The processors are split into several modules dedicated to the Sen4CAP products, in order to allow different parallelization strategies. They are (Figure 2-1):

- Sentinel 1 L2A pre-processor, based on the SNAP chains, that performs extraction of the amplitude and coherences products;

- Sentinel 2 L2A processor, based on the MACCS-ATCOR Joint Algorithm (MAJA) chain, that performs atmospheric corrections and produce bottom-of-atmosphere reflectance values with snow, water, cloud and cloud shadow masks;

- L3A processor that delivers biophysical indicators on vegetation status;

- Markers Database (MDB) processor that delivers the metrics and crop markers by parcel along the season.

- L4A processor that delivers crop type maps along the season;

- L4B processor that delivers grassland mowing information along the season;

- L4C processor that delivers agricultural practices monitoring along the season;

2.2.1 Biophysical Vegetation Indicators (L3A) processor

The biophysical vegetation indicators processor provides four types of products informing about the evolution of the green vegetation:

- The *NDVI*, the most popular indicator operationally used for vegetation monitoring, provided to ensure continuity with existing long-term time series and thus, allowing anomalies detection;

- The *LAI*, an intrinsic canopy primary variable that should not depend on observation conditions, which determines the size of leaf interface for exchange of energy and mass between the canopy and the atmosphere;

- The fraction of Vegetation Cover (*FCOVER*), corresponding to the fraction of ground covered by green vegetation. It quantifies the spatial extent of the vegetation;

- The fraction of Absorbed Photosynthetically Active Radiation (*FAPAR*) by the green and alive elements of the canopy. The FAPAR depends on the canopy structure, vegetation element optical properties, atmospheric conditions and angular configuration.

The product will use as input all available optical S2 (and possibly L8) data.

Complete specifications of all products generated by the biophysical vegetation processor are provided in Table 2-1.

Table 2-1. Vegetation status indicators specifications

Properties	Value
Available vegetation indicators	NDVI, LAI, FAPAR, FCOVER
Spatial extent	Global (not only crop area)
Spatial resolution	10 m
Temporal resolution	NDVI, LAI, FAPAR and FCOVER: for each S2 (L8) acquisition
Geometric accuracy	Same than L1C input data
Format	GeoTIFF raster images
Projection	UTM / WGS84
Metadata	XML file

The products are delivered with several masks that will help appraising its quality:

- the status of the input L2A pixels (no data / cloud / snow / water / land);
- the definition domain of the input spectral bands and of the output variables.

The main and mandatory input of this L3A processor is a S2 time series, optionally completed by an additional L8 time series, turned into L2A products through the L2A processor.

The NDVI is computed using a standard formulation applied to the S2 red (B4) and narrow Near InfraRed (NIR) (B8a) bands.

The LAI retrieval is performed from the bands 3, 4, 5, 6, 7, 8, 9, 12, 13 using machine learning to build a non-linear regression model. For the LAI, FCOVER and FAPAR, the implementation is derived from the one already proposed in the frame of the ESA Sentinel-2 toolbox [RD.1]. The LAI, FCOVER and FAPAR Biophysical Indicators (BI) retrieval is performed by applying a global Artificial Neural Network (ANN) on each pixel considering the reflectance values of all the available bands pre-processed at the Level 2A (L2A) and some geometric configuration as input. The training of the ANN, which consists in generating the training database, defining the neural network architecture and calibrating the network, is not performed within the Sen4CAP system. Instead, the Sen4CAP system benefits from an already trained ANN, made openly and freely available by the Institut National de la Recherche Agronomique (INRA) which developed the algorithm. From the system implementation point of view, this trained ANN is given as auxiliary data to the processor.

Figure 2-2 presents the general workflow of the L3A Biophysical Indicators product algorithm.

Figure 2-2. General workflow of the L3A Biophysical Indicators product algorithm

2.2.2 Markers Database (MDB) processor

This processor aims at providing metrics and crop markers by parcel along the season. These metrics and crop markers were identified and chosen during the Sen4CAP project, for the development of the products and use cases inside the project. They are used by the advanced processors L4A, L4B and L4C inside the system, but they are also accessible - through an API service - to be used in any crop monitoring application outside the system.

2.2.2.1 List of markers

Four types of metrics and crop markers are calculated:

- The **Marker Database MDB1** contains the basic single date markers:
  - o NDVI: mean and standard deviation values by parcel; calculated for each L3B NDVI product;
  - o LAI: mean and standard deviation values by parcel; calculated for each L3B LAI product;
  - o FAPAR: mean and standard deviation values by parcel; calculated for each L3B FAPAR product;
  - o FCOVER: mean and standard deviation values by parcel; calculated for each L3B FCOVER product;
  - o Amplitude: mean and standard deviation values by parcel; calculated for each amplitude product;
  - o Coherence: mean and standard deviation values by parcel; calculated for each coherence product;

- The **Marker Database MDB2** contains the ratio single date markers:
  - o Amplitude VV/VH ratio: mean and standard deviation values by parcel; for each amplitude product;

- The **Marker Database MDB3** contains the markers specifically used for the harvest detection inside the L4C processor; they are calculated each week and are the results of a time series analysis:
  - o M1: Presence of vegetation based on NDVI time series (binary);
  - o M2: Loss of vegetation based on NDVI time series (binary);
  - o M3: Loss of vegetation based on amplitude VV/VH ratio time series (binary);
  - o M4: Low/no vegetation after based on amplitude VV/VH ratio time series (binary);
  - o M5: Loss of vegetation based on coherence time series (binary);

- The **Marker Database MDB_L4A** contains the markers specifically used by the L4A crop type processor. They are the results of a temporal resampling in the case of Sentinel-2 data and of a statistics calculation over different periods in the case of Sentinel-1 data:
  - o Optical, every 10 days, based on 10-meters resolution data:
    - NDVI: mean and standard deviation values by parcel;
    - NDWI: mean and standard deviation values by parcel;
    - Brightness: mean and standard deviation values by parcel;
    - Green B3: mean and standard deviation values by parcel;
    - Red B4: mean and standard deviation values by parcel;
    - NIR B8: mean and standard deviation values by parcel;
    - SWIR1 B11: mean and standard deviation values by parcel;

- o Optical, every 10 days, based on 20-meters resolution data:
  - Red Edge B5: mean and standard deviation values by parcel;
  - Red Edge B6: mean and standard deviation values by parcel;
  - Red Edge B7: mean and standard deviation values by parcel;
  - SWIR2 B12: mean and standard deviation values by parcel;

- o SAR, every week (7 days), based on 20-meters resolution data:
  - Mean Amplitude Ascending VV: mean and standard deviation values by parcel;
  - Mean Amplitude Ascending VH: mean and standard deviation values by parcel;
  - Mean Amplitude Descending VV: mean and standard deviation values by parcel;
  - Mean Amplitude Descending VH: mean and standard deviation values by parcel;
  - Mean Amplitude VV/VH ratio: mean and standard deviation values by parcel;
  - Mean Coherence Ascending VV: mean and standard deviation values by parcel;
  - Mean Coherence Ascending VH: mean and standard deviation values by parcel;
  - Mean Coherence Descending VV: mean and standard deviation values by parcel;
  - Mean Coherence Descending VH: mean and standard deviation values by parcel;

- o SAR, every month, based on 20-meters resolution data:
  - Mean Coherence VV: mean and standard deviation values by parcel;
  - Mean Coherence VH: mean and standard deviation values by parcel;
  - Quantile 10 Coherence VV: mean and standard deviation values by parcel;
  - Quantile 10 Coherence VH: mean and standard deviation values by parcel;

- o SAR, every 2 months, based on 20-meters resolution data:
  - Mean and Coefficient of Variation of Amplitude Ascending VV: mean and standard deviation values by parcel;
  - Mean and Coefficient of Variation of Amplitude Ascending VH: mean and standard deviation values by parcel;
  - Mean and Coefficient of Variation of Amplitude Descending VV: mean and standard deviation values by parcel;
  - Mean and Coefficient of Variation of Amplitude Descending VH: mean and standard deviation values by parcel;
  - Mean and Coefficient of Variation of Amplitude Ascending VV/VH ratio: mean and standard deviation values by parcel;
  - Mean and Coefficient of Variation of Amplitude Descending VV/VH ratio: mean and standard deviation values by parcel;

- o SAR, over the whole monitoring period, based on 20-meters resolution data:
  - Standard Deviation of Coherence VV: mean and standard deviation values by parcel;
  - Standard Deviation of Coherence VH: mean and standard deviation values by parcel.

IMPORTANT NOTE: in version 2.0, **only the markers from the Marker Database MDB1, MDB2 and MDB3 are processed by the MDB processor**. The markers from the Marker Database MDB_L4A are processed by the L4A crop type processor. Still, these MDB4 markers are available in the same way as the other ones, but they will only be available after having launched a L4A crop type job.

2.2.2.2 Accessing the markers

Once they are processed, the markers are stored in .csv and .ipc tables inside the system and they are accessible through an API service outside the system. The location of the tables inside the system, as well as all the information concerning the API service are given in section 4.2.9.

2.2.3 Crop type map (L4A) processor

The crop type map processor generates a crop type map based on the dense time series of S1, S2 (and possibly L8) images and on the subsidy application dataset, using a Random Forest classification algorithm. A subset of the parcels from the subsidy application dataset is used to train the Random Forest model which is then applied to the whole subsidy application dataset. The crop type predicted by remote sensing at the parcel-level can then be compared with the subsidy application dataset to check the conformity of the declaration.

The main output of the crop type map (L4A) processor is a layer (.gpkg or .shp) with the crop type detection based on the EO data, for each parcel in a given region or country. The attributes associated with each parcel include:

- **The original attribute fields** from the original subsidy application dataset characterizing the polygons;
- **The declared crop type for the classification**, using a Look-Up-Table (LUT) provided by the user which makes the correspondence between the original crop type code (numeric or text and a numerical code specific to the classification;
- **The first and second crop types predicted by the classification**, associated with the two highest confidence levels (using the internal numeric code specific to the classification);
- **The confidence levels** corresponding to the first and second crop types predicted by the classification (ranging from 0 to 1);
- **A set of flags** characterizing the polygons in terms of:
  - General high-level land cover class:
    - 0: other natural areas;
    - 1: arable land;
    - 2: annual crop;
    - 3: grassland;
    - 4: fallow land;
    - 5: greenhouse and nursery;
  - Area;
  - Number of S1 and S2 pixels covered by the parcel;
  - Shape;
  - Geometry: validity, duplication and overlap.

An automatic validation of the classification is also performed by the processor. It provides different statistics as the Overall Accuracy and the Kappa values of the classification, as well as the individual F-Scores of the different classified crop types.

It also processes automatically the crop markers that are used by the classification. These markers are defined in the ATBD document. As the markers processed by the MDB processor, these markers are available through tables inside the system and through an API service outside the system.

Figure 2-3 presents the general workflow of the crop type map (L4A) processor. More information on the algorithms underlying this processor can be found in the corresponding ATBD document.
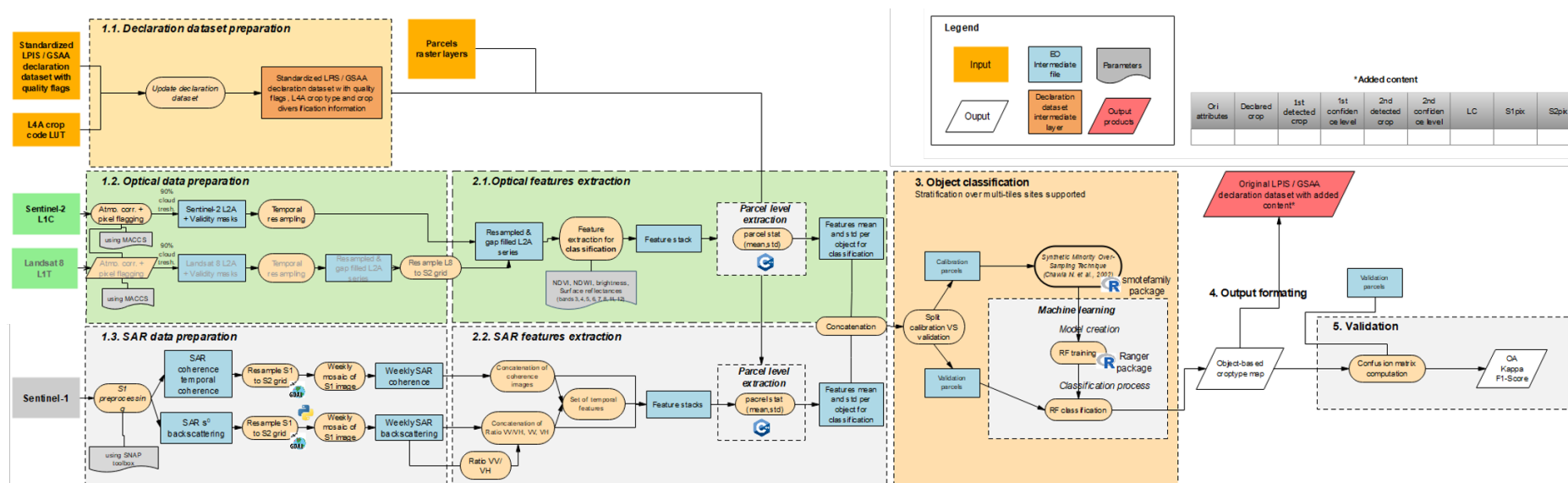
Figure 2-3. General workflow of the L4A crop type map production

	Ref	Sen4CAP_SUM_v2.0	
.esa	Issue	Page	Date	**sen4cap**
	2.0	23	07/01/2021	common agricultural policy

A typical use of this crop type map is the compliancy check at the holding-level with regard to the crop diversification regulation. It is performed in a second step by executing a dedicated script. The crop diversification rules that apply to each holding are first identified and then assessed. It enables to add the following information to the crop type map:

- **Crop code specific for the crop diversification assessment**, resulting from the possible merging of individual crop types belonging to a same group of crops in the specific context of the crop diversification rules;
- **Conformity assessment at the parcel-level**, comparing the crop type declared by the farmer with the ones predicted by remote sensing (conform, not conform, not classified);
- **Crop diversification category to which the holding belongs**, that will drive the subsequent crop diversification assessment by defining the rules that the holding needs to respect to be compliant;
- **Results of the compliancy assessment at the holding-level** with respect to the crop diversification rules **(**not required, compliant, not compliant or missing info).

2.2.4 Grassland Mowing (L4B) processor

The grassland mowing detection product (L4B) is based on the processing of S1 and S2 derived products, which are respectively coherences, calibrated amplitude backscatter and three BIs which are the NDVI, the FAPAR and the LAI. The processing is performed independently on the S1 and on the multispectral optical data and the results are merged to provide a single output by fusing the detections through the exploitation of their reliability indicators. Following this approach, the system architecture is mainly based on two parallel processing chains linked at the final stage through the "mowing detections fusing" module.

The output of this processor is an ESRI shapefile which includes, for each parcel, the following attributes:

- Data representing the input for the analysis:
    - Unique identifier of the parcel (parcel_id);
    - Declared agriculture practice and its type (crop_code);
- Results of the analysis:
    - number of mowing events detected on the whole period analyzed (mow_n)

      For n that can vary from 0 to a maximum of 4 mowing events:

        - temporal interval in which the mowing event occurred: start date of each possible n mowing period (mn_dstart) and end date of each possible mowing period (mn_end);
        - reliability flag given in terms of probability of right mowing (mn_conf);
        - satellite mission data used for detection of mowing (mn_mis);
    - flag (compl), providing the results about compliancy or not with country regulations and reasons (e.g.: no mowing occurred in the mandatory temporal window).

The general workflow of the module is depicted in Figure 2-4.

Figure 2-4. General workflow of the L4B Grassland Mowing product algorithm

2.2.5 Agricultural practices monitoring (L4C) processor

The agricultural practices monitoring product aims at:

1. the identification of the agricultural practice of crop harvesting/clearance,
2. the comparison of the temporal behavior of vegetation of fallow lands, catch-crops or nitrogen-fixing crops with the pre-defined (country specific) EFA practices rules,
3. the tillage practice detection.

These three objectives are achieved through the analysis of dense time series over the agricultural parcels.

The first objective is achieved for the parcels with the land cover class arable land, the second for the parcels for which EFA practices have been declared and on which their compliance with national regulations are required to be verified, and the third can be achieved for the two sets of parcels.

The developed methodology relies on the analysis of dense temporal profiles.

The generation of temporal profiles is based on optical (S2 and L8) and Synthetic Aperture Radar (SAR - S1) imagery. NDVI is used as the optical-based signal, at a spatial resolution of 10 m. The SAR-based signals include backscatter temporal profiles (ascending and descending orbits for dual VV and VH polarization) and coherence temporal profiles (for VV polarization) at 20 m spatial resolution.

A set of markers are derived from these time series, which inform about the following conditions:

- the presence of vegetation / bare soil;
- the no loss (growth) of vegetation (for EFA practices);
- the loss of vegetation (in the form of harvest, clearance);
- the changing (dynamic) or stable status of the parcel conditions within 6-day periods.

These markers are then interpreted to provide an identification of the harvest/clearance week, an initial assessment of the degree of compliancy of an agricultural parcel with a particular agricultural practice and the tillage practice week.

The logical dataflow of the L4C agricultural practices monitoring processor is provided in Figure 2-5.

Figure 2-5. Logical data flow of the L4C agricultural practices monitoring product

2.3 System implementation

The Sen4CAP system is composed of the following layers:

- User interface layer

- Service layer

- Persistence layer

The architecture of the system as well as the relationship between the components identified is given in Figure 2-6.
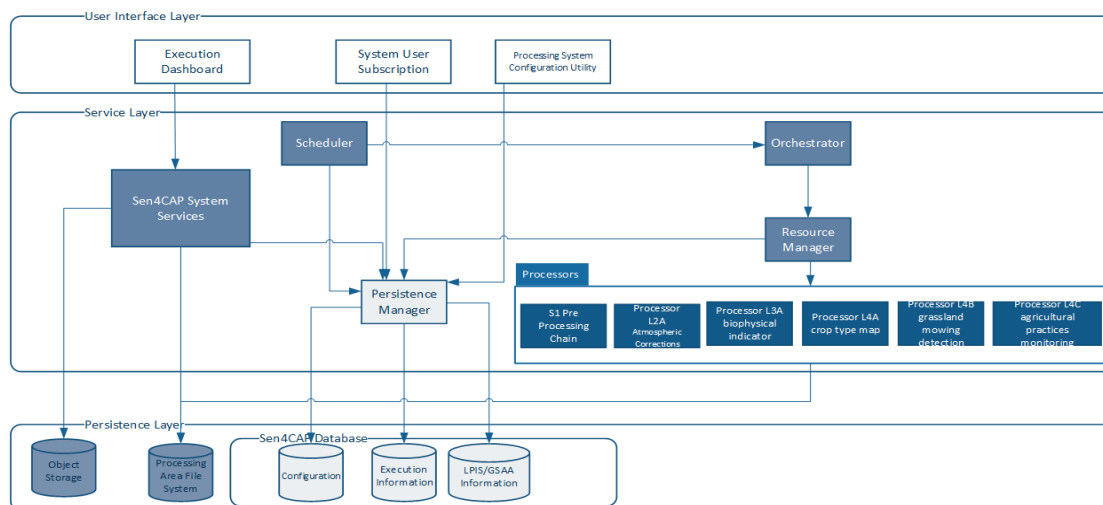
Figure 2-6. Software System Architecture

2.4 The Sen4CAP Orchestrator

Automated and *manual* (created from the Sen4CAP system website) processing jobs are executed via the Sen4CAP Orchestrator. This is a system component that translates a job request into actual command invocations. The Sen4CAP Orchestrator handles:

- parametrization of jobs: system-default and job-specific parameters are merged by the Orchestrator, allowing for any parameter to be customized by the user;

- job decomposition: the Orchestrator splits jobs into multiple steps, some of which can be executed in parallel;

- job invocation: the steps of a job are sent to SLURM[1], allowing the system to take advantage of a processing cluster, if configured as such;

- job monitoring: the Orchestrator is notified of failed steps and cancels the remaining steps that depend on those;

- automated job creation: when processing jobs are scheduled via the website, the Orchestrator decides whether a job can be executed and chooses the input data.

---

[1] SLURM is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters (https://Slurm.schedmd.com/)

As such, if running in the automated mode is desired, the Orchestrator is an essential part of the Sen4CAP system.

2.5 The Sen4CAP Services

Sen4CAP Services is an evolution of the former download component. It is decoupled from the Sen4CAP system and it offers the following functionalities:

- Pluggable architecture for querying and fetching products from remote repositories: out-of-the box, the module comes with plugins for Copernicus SciHub (for S1 SLC products and S2 L1C products), for USGS (for L8) and for DIAS-ses (for S1, S2 and L8);

- More configurable support for L1 products querying and retrieval:
  - Exposal of query parameters so that each data source can be customized;
  - Separation of query and download sources; for example, SchiHub can be configured for querying the list of available products for a site while a local storage can be configured as download site for the products returned by the SciHub query;
  - Independent parallelism thresholds for data sources; for example, SciHub can be configured to allow maximum 2 connections (which is its actual limit), while Amazon Web Service (AWS) can be configured to allow 4+ connections;

- A RESTful API for accessing/controlling different functionalities of the Sen4CAP system from an external application. The operations that can be performed using the services are:
  - Querying the list of sites, delete a specific site or get the seasons of a site;
  - Enable or disable downloading for a certain site and/or satellite and extract the status of downloading for a site and/or satellite;
  - Stop or start the downloading of a certain site and/or satellite;
  - Get the data sources configuration;
  - Reprocessing a S1 acquisition for the selected polarisation(s);

- Execution of the S1 pre-processing chain;

- Receive download progress notifications from the Sen4CAP system;

- Allow moving the L2 products to Object Storage.

2.6 The Sen4CAP Visualisation Tool

The Sen4CAP visualization tool, built on top of Geopedia back-end, supports:

- analyzing the Sentinel-derived markers and products generated within the project: after importing the Cultivated crop type map, Grassland mowing product and Agricultural practices monitoring product to Geopedia as vector layers, user can add the metadata of the Sen4CAP products, make queries over multiple fields within the same layer (e.g. by Farm Id and Parcel Id) and view results at a farm level for compliance assessments;

- viewing of the input datasets provided by the PAs: the subsidy applications data can be imported to Geopedia as a vector layer;

- integration of the satellite imagery used in/by the Sen4CAP EO system: after adding the "Sentinel" layer (which is special type of layer in Geopedia that allow accessing the Sentinel, Landsat and biophysical vegetation status indicators data), user can:

    o check the time intervals for which the data is available ("Available dates" tool);

    o display the images for a specific date;

    o get an overview of available satellite images within a specified period by using the "Time animation" tool;

    o get the statistical information on the biophysical vegetation status indicators values within a selected polygon ("Data feature info" tool).

2.7 The Sen4CAP WMS / Statistical Info Services

The Sen4CAP Web Mapping Service (WMS) / Statistical Info Services provide core distribution engine for Sen4CAP results. These services, Open Geospatial Consortium (OGC) compliant and proprietary Application Platform Interfaces (APIs), make it possible to easily integrate the data processed in the cloud (e.g. the DIAS) within PA's environment.

Any standard GIS application can use the Sen4CAP's WMS service to visualize the S2 and L8 pre-processed bands, S2 and L8 true color/false color imagery and biophysical vegetation status indicators. The Sen4CAP distribution services also include the statistical info service which provides the biophysical vegetation status indicators data for the given time, computes statistics on these data at the parcel-level, and returns the result in the form of a JSON object.

The Sen4CAP WMS/Statistical Info Services package allows user to set up a distribution environment (similar to Web Map Server but optimized for Sen4CAP data) at his/her own premises.

3. EO processing system deployment

3.1 Prerequisites for automatic usage

Before starting the installation of the Sen4CAP system for an automatic use, a set of preconditions is expected to be met in terms of system hardware, operating system or packages installed.

The minimum hardware requirements are:

- disk space for system installation: 80 GB;

- disk space for the resulted products (/mnt/archive - see section 3.3.1): this depends on the number of sites to be defined, the seasons and the number of tiles to be used for each site.

- disk space for the internal directory where the shapefiles of the created sites will be uploaded in order to be used by the system (/mnt/upload - see section 3.3.1): 20 GB is needed;

- RAM: between 64 and 128 GB, depending on the size of the country;

- number of CPUs: between 8 processors and 16 processors (or more), depending on the size of the country.

CentOS 7 (minimum version 7.5) 64-bit is required by the Sen4CAP system. The graphic interface (X Server) should be also installed if it is intended to use the Sen4CAP Configurator (see Appendix E.1 Processing System Configuration Utility).

The Linux user who will perform the installation should have the root rights (i.e. be part of the sudoers list), the installation process will perform a lot of `yum` installation

Also, in case of proxy server, the "/etc/yum.conf" file should be properly configured. Complete instructions to perform this configuration of the proxy server in yum.conf see are given in https://www.centos.org/docs/5/html/yum/sn-yum-proxy-server.html. For example, some lines with the ones below should be added in yum.conf file:

```
# The proxy server - proxy server:port number

proxy=http://mycache.mydomain.com:3128

# The account details for yum connections

proxy_username=yum-user

proxy_password=qwerty
```

Due to licensing issues, the Sen4CAP installation package does not contain the installation package for MAJA. MAJA must be downloaded separately from the CNES website, performing the following steps:

- *Go to https://logiciels.cnes.fr/en/node/58?type=desc*

- Click on the "Download" tab;

- Click on the "Download" button of the 3.2.2 TM or 4.0 version;

- Read the license carefully;

- Fill the form with your information;

- Don't forget to accept the license;

- The downloading is starting.

IMPORTANT NOTE: The Sen4CAP v2.0 system supports **both the versions 3.2.2 TM and 4.0 of MAJA**. Both versions can be used. Nevertheless, it is important to note that the biophysical indicators and the different L4x products obtained using MAJA 4.0 have not been validated so far.

The MAJA application is needed for the generation of L2A products from S2 L1C and L8 L1T products and it must be installed prior to the Sen4CAP system. Instructions for installation are provided by CNES inside of the MAJA installation package.

In any case, the Sen4CAP installer will check if MAJA is already installed and will prompt the user to provide the installation path if it is not found in the default one (which is /opt/ maja). For any information about MAJA, its own Software User Manual can be checked [RD.2].

IMPORTANT NOTE: If MAJA is installed before the system installation and not by the Sen4CAP installer, by default, MAJA is installed in the /opt/maja directory but with read/write rights only for the user root. In order to be accessible by the Sen4CAP default system user (sen2agri-service), the following command needs to be executed after MAJA installation

```
sudo chmod -R a+rx /opt/maja
```

For downloading S1, S2 and L8 acquisitions from the distribution server, the system will need 2 accounts to be provided after the installation:

- An account and a password for the ESA Sentinels Scientific Data Hub (SciHub), allowing downloading S1 and S2 products. This account can be obtained accessing https://scihub.copernicus.eu/dhus/#/home and then accessing the "Sign up" section;

- An account and a password for the United States Geological Survey (USGS) portal, allowing downloading L8 products. The account can be created accessing the link https://ers.cr.usgs.gov/login/ and following the "Create new account" option.

  Nevertheless, in order to activate the downloads for your account the following additional operations need to be performed :

  o Login to your account on USGS

  o Go to https://ers.cr.usgs.gov/profile/access

  o Push the "Request Access" button and fill the information in the form that will be displayed:

    ▪ "Datasets Required" – Landsat Collection 1 Level-1

    ▪ "Data Products Required" - Landsat 8 OLI/TIRS C1 Level -1

    ▪ "Scenes required" – In your case you can fill 2 monthly or 24 yearly (depends on your site dimension). You can also put more if you consider you will have additional sites to monitor

    ▪ "Scripting Capabilities": python, java

    ▪ For the other fields, you will have to fill the requested information

  After that, you should receive after several hours an email from USGS confirming the activation. Once this is done, the L8 should start also being downloaded.

IMPORTANT NOTE: Due to recent changes in the USGS API, the **automatic ingestion of Landsat-8 data from the USGS portal is not functional** in the version 2.0 of the system. Corrections will be made in the next version to take into account these recent changes.

Sub-section "4.2.1 Configure the data sources" describes how to configure these accounts in the system.

Later on, after the system is completely installed, a .zip archive containing the shapefile delineating your site of interest is required. This archive should include a ".shp", a ".prj", a ".dbf" and a ".shx" file. More information about the way this archive is used is given in the sub-section 4.2.2).

3.2 Installation package content

The installation package of the Sen4CAP system consists in the following main folders:

- **install_script** - contains the installation scripts that are used to create the distribution and to install the system;

- **rpm_binaries** - the RPM files for all other system components (SLURM, orchestrator, downloader, processors);

- **sen2agri-services** - the folder containing the Sen4CAP services to be installed.

Additionally, the following archives are provided beside the system installation package:

- **gipp maja**- the gipp files used by the L2A processor (MAJA) during the processing of the L1C products. These gipp files correspond to the MAJA 3.1 and 3.2.2 version. Please note that if another more recent MAJA version will be installed, these gipp files should be updated accordingly;

- **srtm.zip and swbd.zip -** these archives contain the Digital Elevation Model (DEM) files needed by the L2A processor (MAJA). Although they are part of the installation package, they are provided separately due to their size and because they will normally not change from a version to another. Before starting the installation of the system, these two files should be copied inside the installation package, in the "Sen4CAPDistribution" directory

IMPORTANT NOTE: The L2A processor, named MAJA, is not provided with the system installer and shall be downloaded separately from CNES site (see below). If MAJA is installed before running the Sen4CAP installer, as described in the next paragraph, the Sen4CAP install script will look for the executable and eventually prompt for its path.

Nevertheless, for an easier installation, MAJA version 3.2.2. / 4.0 installer can be downloaded and copied in a subfolder called **"maja"** in the root of the installer (on the same level with **install_script** and **rpm_binaries** folders) and the system will automatically install it from here. This is the recommended method as the Sen4CAP installation script will take care of some steps like setting some rights to the MAJA installation path (by default MAJA is installing in /opt/maja but its installer does not give read rights for this folder to other users than root and because of this the sen2agri-service user fails to access this folder, as already described in the paragraph 3.1).

IMPORTANT NOTE: **Alternative DEM** can be used by the system for regions not covered by the SRTM (for example, AsterDEM can be used for S1 pre-processing or/and EUDem for MAJA pre-processing). These cannot be provided at installation but they can be provided later. See Appendix A.6 Using AsterDEM for S1 pre-processing and Appendix A.7 Using other DEM for S2 L2A pre-processing for examples of configuring system with a different DEM.

3.3 Installation and configuration for automatic usage

3.3.1 Installation procedure

The following procedure allows installing the system for automatic usage. It is worth noting that the installation for automatic usage implies that all functionalities of the system will be installed and made available, including those for running the manual mode.

Before of the installation, some default directories have to be created in the system either physically or mounted. The default directories are:

- /mnt/archive – this is the default working directory for the system;

/mnt/upload – the folder where the files from the user are uploaded (for example shapefiles with in situ data).

These directories can be changed after installation using the *"Sen4cap Configurator"* application (see Appendix E.1 Processing System Configuration Utility).

The user sen2agri-service (which will be added by the installer, see below) should have full access rights (read/write/execution) over these default directories. In order to do this, launch the following command from a terminal, as a user which is in sudoers list:

```
sudo mkdir /mnt/archive

sudo mkdir /mnt/upload

sudo chmod -R a+wrx /mnt/archive /mnt/upload
```

The directories "/mnt/archive" and "/mnt/upload" can also be mount points to another external storage.

The users on a system and the root user are not recommended to be used as there cannot be known a priori. During the installation, the Sen4CAP installer will create the user *"sen2agri-service"*. This *"sen2agri-service"* user will be the user under which all the Sen4CAP services are executed. The *"sen2agri-service"* user will be created by the installer; he should have write access to these mount points. Since the *"sen2agri-service"* user might not be already created before the installation, these external directories should therefore have initially access rights for all users.

As explained in section 3.1, the SRTM and SWBD files are provided separately, because of their size and also because they will normally not change from one version to another and don't need to be re-downloaded at each update. The following options exist to have these dataset usable by the Sen4CAP system:

- Copy the srtm.zip and swbd.zip in the root folder of the installation package (in the Sen4CAPDistribution directory). In this case, the installer will automatically unzip them and copy them into the right location (/mnt/archive);
- Or, the user unzip the files from the two archives in folders /mnt/archive/srtm and /mnt/archive/swbd respectively. In this case, the installer will detect their presence in the right location and will use them directly.

An example of a structure of the package, including the SRTMs and MAJA is presented in Figure 3-1.

Name	Date modified	Type	Size
docs	7/16/2019 11:15 AM	File folder	
gipp_maja	5/28/2019 4:06 PM	File folder	
install_script	11/4/2019 10:00 AM	File folder	
licenses	7/16/2019 11:15 AM	File folder	
maja	10/28/2019 4:39 PM	File folder	
reference_data	7/16/2019 11:16 AM	File folder	
rpm_binaries	10/22/2019 9:57 AM	File folder	
sen2agri-services	11/1/2019 3:48 PM	File folder	
tools	7/25/2019 10:28 AM	File folder	
AUTHORS.txt	11/28/2018 11:11 AM	TXT File	2 KB
COPYING.txt	11/28/2018 11:11 AM	TXT File	35 KB
LICENSE.txt	11/28/2018 11:11 AM	TXT File	104 KB
SOURCES.txt	11/28/2018 11:10 AM	TXT File	2 KB
srtm.zip	9/22/2017 2:58 PM	ZIP File	16,975,729 ...
swbd.zip	9/22/2017 2:59 PM	ZIP File	833,194 KB
VERSION	11/28/2018 11:11 AM	File	1 KB

Figure 3-1. Structure of the installation package (including the SRTMs and MAJA)

To install and configure the Sen4CAP system, with all its dependencies, it is enough to run the following script that can be found in the distribution package:

```
## open a terminal -- go into /install_script folder:

cd /path/to/Sen4CAPDistribution/install_script

## Run the install script

sudo ./sen4capPlatformInstallAndConfig.sh
```

This script will automatically install the system (SLURM, orchestrator, processors, downloader, website, the database and all other dependencies). The installation is completely automatic, requiring minimum interaction from the user.

A PostgreSQL[2] database is used to keep track of the system information. The database is installed and configured by the installation script. The reference tables are already filled in with all the necessary details when the database is installed. For a list of reference tables and access for advanced users, the reader is referred to Appendix B - Sen4CAP database tables for manual configuration.

---

[2] PostgreSQL is a powerful, open source object-relational database system – see
https://en.wikipedia.org/wiki/PostgreSQL

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
				common agricultural policy
	2.0	35	07/01/2021	

IMPORTANT NOTE: The installation scripts should have execution rights. If by some operations, these rights are lost, the following command could be optionally executed before starting the installation:

```
chmod -R a+x /path/to/Sen4CAPDistribution/
```

3.3.2 Update procedure

The Sen4CAP system contains also an update processor that will allow easily upgrading to newer versions, once these become available. To update your system, follow these steps:

- Download only the Sen4CAP distribution, from the Sen4CAP website. You don't need to download again the other datasets (SRTM, SWBD, etc.) that are still available on your machine;
- Copy the Sen4CAP distribution on the machine where the system is installed;
- Go in the install_script foler and launch the update script:

```
cd /path/to/Sen4CAPDistribution/install_script

sudo ./update.sh
```

IMPORTANT NOTE: you can update your system even if you have already processed or if you are still processing data for a site and season:

- **S1/S2/L8 data download and pre-processing** will be stopped during the update but triggered again when it is finished; everything is automatic.
- **L3B, MDB and L4x processors** will be stopped during the update but not triggered again when it is finished; **you will have to relaunch them** manually after the update.

3.3.3 Additional configurations

Some additional steps are needed to complete the installation of the system. Some of them are mandatory, while other ones are optional.

3.3.3.1 Configure your proxy for the Sen4CAP-Services downloaders [optional]

This step is optional.

If a proxy server is used for the internet connection, the files have to be manually edited by adding as root user the following configuration keys in the */usr/share/sen2agri/sen2agri-services/config/services.properties* file:

- *proxy.host*=<host of the proxy server>

- *proxy.port*=<port of the proxy server>

- *proxy.user*=<user for the proxy>

- *proxy.password*=<password of the proxy user>

- *proxy.type*=<Type of the proxy> that can be one of the following values:

    o *DIRECT* - Represents a direct connection, or the absence of a proxy;

    o *HTTP* - Represents proxy for high level protocols such as HTTP or FTP;

      o   *SOCKS* - Represents a SOCKS (V4 or V5) proxy.

> **IMPORTANT NOTE:** After modifying the file services.properties, the sen4cap-services should be restarted using the command:
>
> ```
> # sudo systemctl restart sen2agri-services
> ```

These values will be used for the first execution of the system, after a fresh installation, when nothing is configured in the system. The next configurations of the system need to be performed only via the IHM of the system that has a dedicated page for this, accessible from the "data sources" item of the main menu (see 4.2.1).

3.3.3.2 Configure your proxy for the web browser [optional]

This step is optional.

In order to allow the web browser having the map world interface functional, it can be needed to add your HTTP/HTTPS proxy information in the Sen4CAP terminal or directly into the Sen4CAP web browser.

From the terminal, you can use the following command lines with the right information:

```
export http_proxy=http://${PROXY_USER}:${PROXY_PASSWORD}@${PROXY_HOSTNAME}:    ${PROXY_PORT}
export https_proxy=https://${PROXY_USER}:${PROXY_PASSWORD}@${PROXY_HOSTNAME}: ${PROXY_PORT}
```

From the Sen4CAP web browser, you should set the proxy information in our settings. For exemple for Firefox, you should set in the Advanced section / Network / Settings the correct information.

3.3.3.3 Sen4CAP website users management [optional]

This step is optional.

By default, the website interface is provided with the following user having admin rights:

Username: *sen4cap*

Password: *sen4cap*

The website uses a user hierarchy to change the user parameters. There are two types of users: admin users and non-admin users. One might take advantage of this "non-admin users", when several sites are run in parallel on the same Sen4CAP system. In this case, a "non-admin user" can be assigned to each site. In this case, in the website interface, the non-admin user can see only information related to the assigned site.

To add users in the database, a special tab is available in the web interface (see section 3.3.4 to see how to connect). Using this tab, you can check the current information corresponding to a user and add a new one (Figure 3-2). You can also define the role of a specific user and, in the case of a regular user (not admin), limit the access only to certain sites.
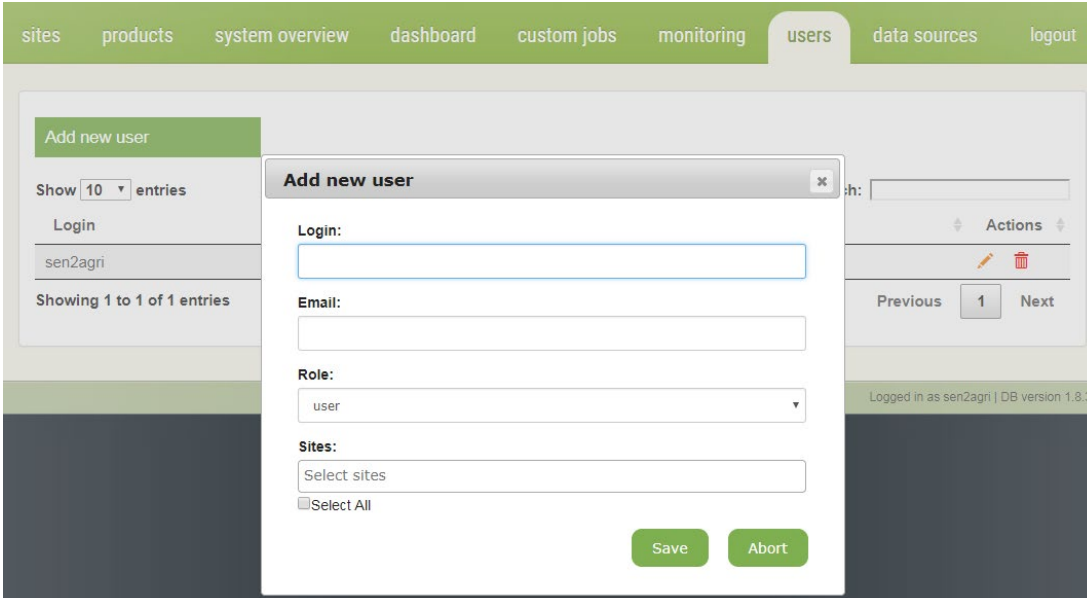
Figure 3-2. In the tab user of the web interface, "Add new user"

After a user is added, the corresponding password need to be set during the first login, by selecting "Set a password" from the login page (Figure 3-3).
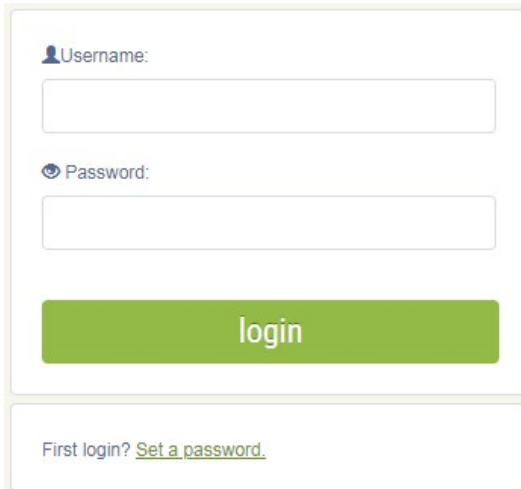
Figure 3-3 In the case of a first login: "Set a password" in the login page

3.3.4 First interaction with the Sen4CAP system

After the installation is finished and all the additional configurations are performed, the web interface of the system can be accessed from a web browser if the user is connected to the same machine as the Sen4CAP system. The following address needs to be inserted: http://localhost/login.php or http://127.0.0.1/login.php

The following interface (Figure 3-4) will be displayed.
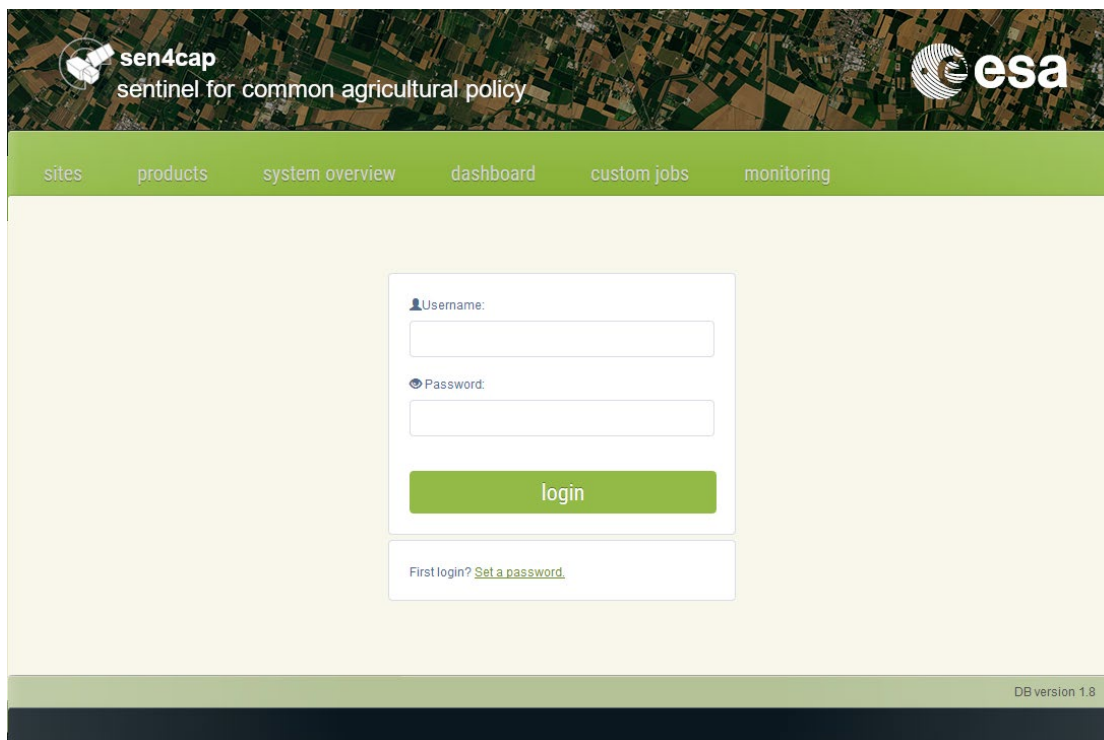
Figure 3-4. Main page of the Sen4CAP web interface

The interface can be also accessed in the same manner from another PC that has access to the port 80 of the Sen4CAP system, by providing in the browser the IP address or the computer name of the Sen4CAP system. For example, the local instance of the Sen4CAP system website can be accessed via http://127.0.0.1/login.php or http://localhost/login.php.

After the user provided a correct user and password (by default, "sen4cap" and "sen4cap"), the user is automatically redirected to the "Sites" page where he will be invited to configure his site to monitor (see section 4.2.2).

This web interface should allow performing all operations of the Sen4CAP system, allowing users that are not-Linux expert to configure and run the system without problem. Yet, in this version 1.0 release candidate, this is not totally the case. The web interface can be used to perform all operations except editing the configuration parameters in the database. In order to edit these configuration parameters in the database, the sen4cap configurator (or manual editing in the database) must be used. More details about these operations are provided in the next section.

4. How to use the Sen4CAP EO processing system

4.1 General

To put it simple, the Sen4CAP EO processing system transforms S1 SLC, S2 L1C and L8 L1T products into higher-level products by means of its L2, L3A (biophysical indicators), L4A (crop type), L4B (grassland mowing) and L4C (agricultural practices monitoring) processors.

The normal functioning state of the system is the *automated mode*, i.e. the automatic download / import and processing of the S1 SLC, S2 L1C and L8 L1T products and production of higher-level products.

The automatic execution of the system is based on the following items:

- At least one site needs to be defined inside the system. Additional sites can be created and edited from the "Sites" page of the Sen4CAP website;

- For each site, the extent and the season to monitor need to be defined. At least one season should be defined, specifying the start and end dates of the season. These parameters are associated to by-default values, which can be modified by the user. The season dates are defined by a month, a day and a year;

- Based on these two information (extent and season definition), the system automatically launches the downloader and the processors to generate the higher-level products.

IMPORTANT NOTE: the subsidy application dataset will also be needed to run the higher-level processors. How to format and import this dataset is detailed in section 4.4.3 Subsidy application dataset preparation and in the Appendix F11. LUT file structure.

As for the download / import, some precisions need to be done:

- The system starts to download / import optical products 2 months before the start of the season date. For example, considering a start of the season on 2018-05-01, products download will thus start on 2018-02-01. These 3 first months of data are needed to initialize the L2A processor (see section 4.4.2) but will not be used to produce the higher-level products;

- Similarly, the system also tries to download products during 2 months after the end season (so considering an end of season of 2018-09-31, it means until 2018-10-31), to avoid losing any products inside the season which may be added after the end of season day.

Each processor is associated with main and advanced parameters and each parameter has by-default values. Each processor is automatically run using these by-default values except if the user changes them. More information regarding the processors parameters can be found in section 4.4. This by-default parameterization results from consistent studies carried out during the project. It is therefore recommended to be cautious when modifying them.

As an alternative to the *automated* mode, each processor can also be run in the *manual mode*. In this case, processors can be invoked in various ways:

- from a terminal window, by executing a specific Python script;

- from the Sen4CAP web interface, for L3A, L4A, L4B and L4C processors. In this case, immediate execution is not performed. Instead, a scheduled job is created and the processor will be executed when the system decides it has enough resources to do it. This is done with the purpose of not interfering with the system production jobs already planned.

The following sections present how running the system in the automated and manual modes. They also give information on the way to deal with the data and files that must be imported and that are required for the L4x processors.

4.2 First steps for the automated usage of the system

4.2.1 Configure the data sources

Before starting the configuration of the sites to be monitored, it is needed to configure first the data sources to be used for fetching products into the system. The Sen4CAP system offers the possibility to adapt the data sources configuration directly from the web interface (Figure 4-1). For example, you can define which downloading platform to use for the "query" and "download" tasks of the downloading, the user and password corresponding to your downloading platform, define the destination folder for the downloaded products, and many more (Figure 4-1).
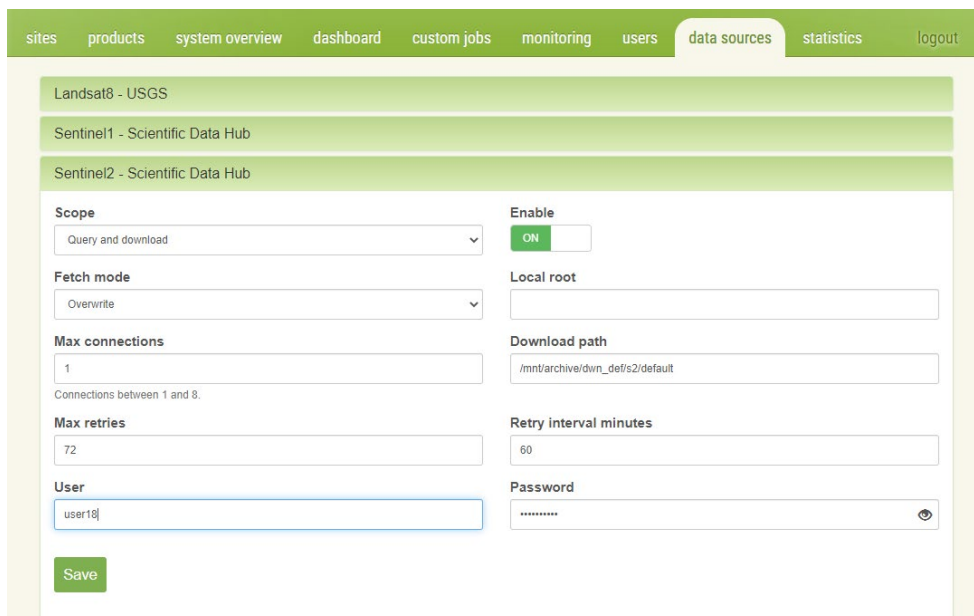
Figure 4-1. The "data sources" tab of the web interface - Configure the data sources parameters

A data source is a module that has the capability of searching and downloading products from remote sources. Every such module is intended for a combination of the remote source and a satellite. Out-of-the-box, the Sen4CAP system comes with three pre-configured data sources: two for Copernicus Scientific Data Hub (SciHub – for Sentinel-1 and Sentinel-2 products) and one for US Geological Survey (USGS – for Landsat-8 products).

The default configuration uses both SciHub and USGS for search and download. This is enough for an on-premises installation of the system. But, depending on particularity of the installation, several configuration options may be deemed necessary.

IMPORTANT NOTE: The sen4cap-services are periodically checking for the changes in the configuration with a timeout of 1 hour. If it is desired to take into account the changes immediately, the application can be restarted from a command line using:

```
sudo systemctl restart sen2agri-services
```

4.2.1.1 On-Premises Installation without any Local Product Repository

This is the default scenario and the only necessary configuration of the Sen4CAP data sources consists in providing the credentials (user and password) for both SciHub and USGS.

4.2.1.2 On-Premises Installation with a Local Product Repository

Additionally to 4.2.1.1, in cases when the system is installed on an environment which has a product repository available (i.e. products already downloaded in a local folder or on a share on the local network), the following should be applied (the steps are to be applied for all the data sources):

- Change the **[Scope]** of the data source to "*Query*". This way, the remote source will only be used for searching products over a site for the given season, but not to download them.

- Change the **[Fetch mode]** to "*Symbolic link*" or *"Direct link to product"*. When the services will execute the product lookup, they will just create symbolic links to the locally available products or the path to the local directory will be added directly into the database respectively.

- Set the **[Local root]** to the root path of the local product repository. Each data source can have its own local root.

  The default expected organisation of the local products is a folder hierarchy into which products are grouped by year, month and day.

  For example:

  ```
  /mnt/store -> this is the local root
    /2019
      /01
        /20
          /<product_1>
          /<product_2>
        /21
          /<product_3> etc.
  ```

  Nevertheless, the structure of the local repository can be described via a couple of settings which are available in the **/usr/share/sen2agri/sen2agri-services/config/services.properties** configuration file. These are:

  o **<datasource>.local.archive.path.format**

    A relative path, supporting some placeholders, should be used. The supported placeholders (for all satellites) are:

    - yyyy = the year

    - MM = the month, left-padded with '0'

    - dd = the day, left-padded with '0'

    For example: "S1-SLC/yyyy/MM/dd", where S1-SLC is a real folder

    This would allow the grouping of products by acquisition date and for easier existence check.

- o **`<datasource>.path.suffix`**

  This specifies if the products, if uncompressed, have their product folder ending in .SAFE, or, if compressed, the extension of the archive file. Allowed values are:

  - *none*

  - *.SAFE*

  - *.zip*

  - *.tar_gz*

- o **`<datasource>.product.format`**

  This specifies the format of the products and can have one of three values: '*folder*', '*zip*' or '*tar_gz*'

  where <datasource> can be one of:

  - *SciHubDataSource.Sentinel1*

  - *SciHubDataSource.Sentinel2*

  - *USGSDataSource.Landsat8*

- o **`<datasource>.Sentinel1.auto.uncompress`**

- o **`<datasource>.Sentinel2.auto.uncompress`**

  These two keys specify if when downloading products from the specified source to uncompress the products or not (possible values "true" or "false" with the default value set to "false").

  IMPORTANT NOTE: if value for key "downloader.use.esa.l2a" is set to true in the config table, this value should be set on "true" for the Sentinel2 as the higher-level processors will not be able to handle archives.

4.2.1.3   On-Premises Installation within a DIAS Platform

The configuration of the data sources, in the case of an installation on a DIAS platform (supported DIAS platforms are CREODIAS, Mundi, Sobloo and Onda) is similar to that of 4.2.1.2.

However, to cope with the specific paths of the products for each of the DIASes, additional configuration needs to be done.

In the configuration file **/usr/share/sen2agri/sen2agri-services/config/services.properties**, there are 3 sections, commented, for CREODIAS, Mundi, Sobloo and Onda (CREODIAS is using the default configuration).

Therefore, depending on the DIAS platform where the system is installed, the respective section **has to be uncommented** (and leave the others commented).

4.2.2 Configure a new site

By-default, the Sen4CAP platform does not contain any geographical site. This is to the user to create at least one site. This site is then used by the downloader/import processor (see Appendix E.7 Sen4CAP Downloader) to download S1 SLC, S2 L1C and optionally, L8 L1T products. The first step for the automated usage is therefore to create and configure a new site.

In order to create a new site, the Sen4CAP system interface should be used. Select the "**Sites**" tab and press the "**Create new site**" button. At that moment, the dialog box for configuring a new site is displayed (Figure 4-2).
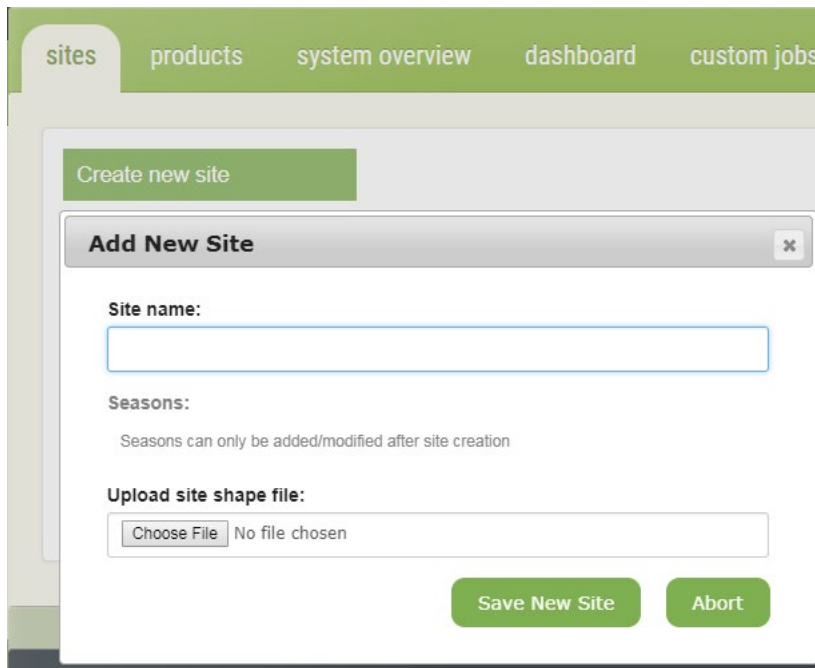
Figure 4-2. Dialog box for site creation through the Sen4CAP system interface

In the "**Add New Site**" dialog box, the following information should be provided:

- Site name: this should be a unique site name defined in the system. The name of the site should follow the following rules: first letter must be uppercase and letters, digits, spaces and underscores are allowed;

- Upload a shapefile with the site extent. The uploaded file will have to be in a ".zip" archive that contains at least ".shp", ".shx" and ".prj" files (".dbf" files can be also present but they are not mandatory). The site can be defined by more than a polygon, i.e. the shapefile can contain a multi-part polygon.

IMPORTANT NOTE: Make sure that your polygon (or each polygon in the case of a multi-part polygon) is **not made of more than 150 points.**

After providing all information in the dialog box, the user should choose "**Save New Site**" and the site is created.

An error will be given if the information provided is not correct (for example in the case of invalid shapefile archive).

Once at least one season is added and the site is "Enabled", the download and processing will be automatically started. If the user needs to check and/or change some of the processors parameters in the system, it is highly indicated to disable the site and re-enable it when all the parameters are set.

4.2.3 Editing the sites to define season and uploading the mandatory auxiliary data

4.2.3.1 Editing a site

As specified in the previous paragraph, by default, a new created site is not associated with season dates, which are to be defined in a second step. To this end, on the Sen4CAP system interface, go to the "**Sites**" tab and press the "**Edit**" button corresponding to your site (Figure 4-3).
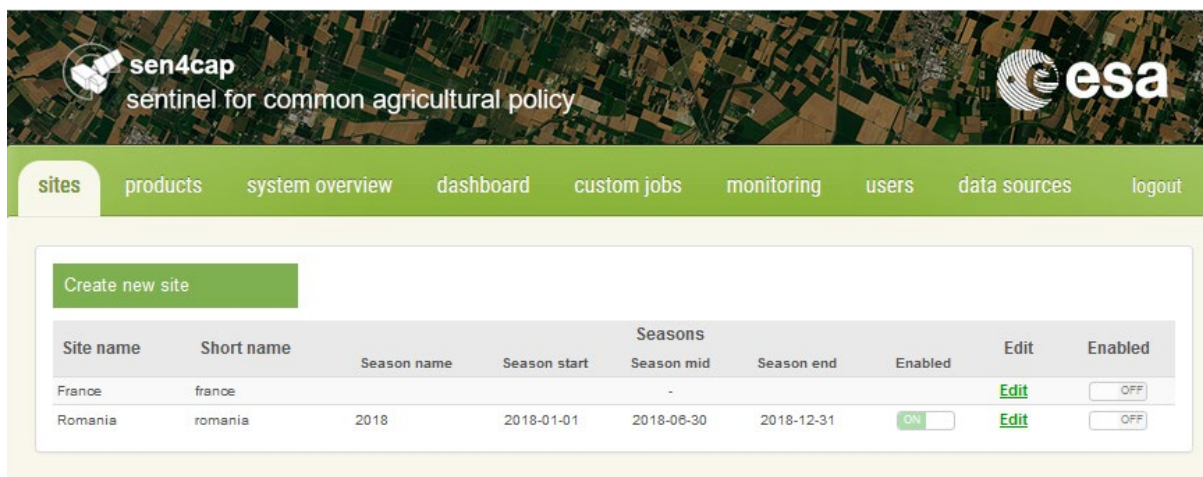
Figure 4-3. Site page of the Sen4CAP interface, to edit site or seasons

The dialog shown in Figure 4-4 will then open.

Figure 4-4. Dialog box for sites edition (including season dates) through the Sen4CAP system interface

In order to add a season, press the ✚ icon in the "**List of Seasons**" section. It will allow you specifying the following parameters: "Season Name", "Season start", "Season mid" and "Season End". It will also give you the possibility to decide if the season should be enabled or not.

At this moment, the user has the possibility to specify which processor will be run in the automated mode for the site and season just defined. This is done simply by ticking the boxes associated with the processors you want to activate, as illustrated in Figure 4-5. Note that the pre-processing processors (for S1, S2 and subsidy application dataset) are always run automatically.

The user can also select what sensor to enable for downloads: by default, L8, S1 and S2 sensors are "Enabled" but the L8 sensor can be disabled. The dialog presents also the list of S2 and L8 tiles intersecting the site. The user has the possibility to remove some of them, if he is not interested in all of them. The editors for the tiles become enabled once the site is enabled.

Figure 4-5. Dialog box when a season is defined and the possibility to activate processors is visible

By selecting the processors to be active for the season, the corresponding scheduled jobs will be added to the season for automatic processing. Please refer to section 4.3 for additional information about the scheduled jobs. The processors that can be active are:

- L2A and L2-S1: these processors correspond to the pre-processing of both S1 and S2. As already mentioned, they are enabled by default (cannot be modified) as they are the ones that provide input products for all other processors;

- LPIS: this processor performs the formatting and quality control of the subsidy application datasets. It is also enabled by default (cannot be modified) and it is triggered only when a new dataset is uploaded into the system;

- L3B - Biophysical products (NDVI, LAI, FCOVER and FAPAR): this processor is associated with a "Cycle" type schedule; it is executed every day in order to generate one set of L3A products for each L2A product that was acquired and pre-processed during the last day;

- MDB – Markers Database: this processor is associated with a "Cycle" type schedule; it is executed every day and starts the 2nd day after the start of the season;

- L4A - Crop type map: this processor is associated with a "Repeat" type schedule; it is executed at the end of each month and starts at the mid-season date;

- L4B - Grassland mowing detection: this processor is associated with a "Repeat" type schedule; it is executed every month and starts 1 month after the start of the season;

- L4C - Agricultural practices monitoring: this processor is associated with a "Cycle" type schedule; it is executed every week (7 days cycle) and starts 7 days after the start of the season.

IMPORTANT NOTE: This possibility to specify the active processors is only available when the season is defined. It cannot be edited later on.

Once the season is defined, it can be saved using the 💾 icon. The changes can be cancelled using the icon ❌.

After the successful creation of a season, a set of scheduled jobs corresponding to the selected processors are automatically created for that season. These jobs are available in the "Dashboard" section of the Sen4CAP system interface and will start to execute once the season and the site are Enabled.

At this moment, after the creation and the activation of the season (Enabled), the user has the possibility to enable the site.

4.2.3.2 Uploading and importing subsidy application dataset

The subsidy application dataset is mandatory to run the higher-level products (L4x). The system can start working without this dataset, performing already the download / import, the S1 and S2 pre-processing and also creating the L3A Biophysical Products. As soon as it is available (and in any case, before the creation of the L4x products), the subsidy application dataset needs to be uploaded.

In order to upload the subsidy application data (but also other processors files), the "Upload files" option can be used. A new window then opens, like in Figure 4-6.
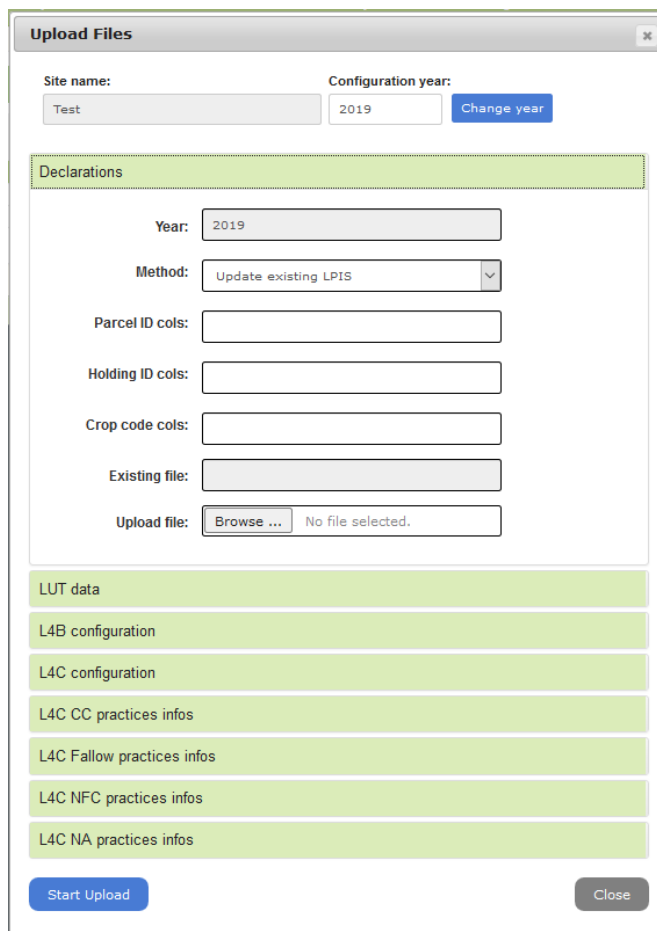
Figure 4-6 - Uploading subsidy application data

In order to upload the subsidy application data, the "Declarations" section can be used where the following information are requested:

- The import method ("Replace the existing", "Update the existing" or "Reuse existing (incremental)");
- One or several columns in the uploaded shapefile that is/are used to define the original parcel ID. If only one column is provided, this will be used as the original parcel ID. If several columns are provided, they should be specified as comma (",") separated and a unique ID will be created by concatenating the column names via "-". For example, giving the columns A,B,C the generated unique ID will be "A-B-C";
- The holding ID column name(s) in the uploaded shapefile. Same rule as for the parcel ID applies also here;
- The crop code column name in the uploaded shapefile;
- The uploaded file needs to be a zip file containing the shapefile for the subsidy applications.

IMPORTANT NOTE: The input shapefile shall **NOT** contain one of the following column names: "ori_id", "ori_hold" or "ori_crop." Additionally, if the values in the column specifying the crop code are numeric, then the type in the input shapefile should also be numeric or integer.

4.2.3.3 Uploading and importing the crop code Look-Up-Table

This Look-Up-Table (LUT) aims at associating each crop declared by the farmer with a unique crop code (integer) and crop name (string) that will be used for the internal Sen4CAP processing. This LUT also associates each declared crop type with a broad land cover category made of 6 classes:

- 0: other natural areas;
- 1: annual crop;
- 2: permanent crop;
- 3: grassland;
- 4: fallow land;
- 5: greenhouse and nursery.

This broad classification will serve to determine if a given parcel is of interest for a given processor. For instance, the L4A Crop Type processor will only process by-default the parcels belonging to the 1 to 4 categories.

The LUT must contain all the possible crop types declared by the farmer and must link to the crop code column of the subsidy application data uploaded before with a new field named "Ori_crop". The structure of this LUT is provided in Appendix F11. LUT file structure.

The LUT table can be uploaded through the section "LUT data" of the "Upload file" tab (Figure 4-7).
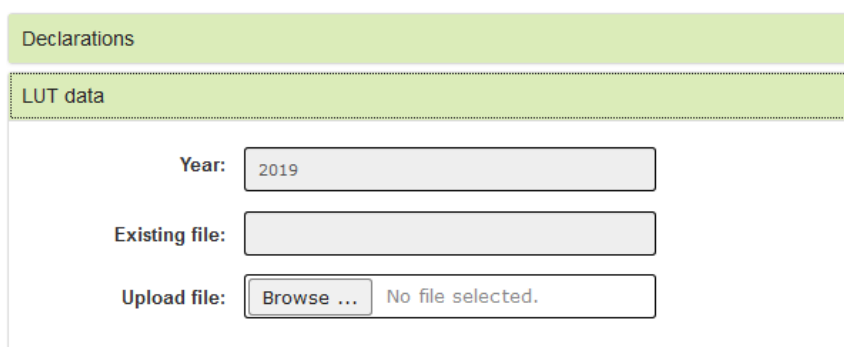
Figure 4-7. LUT data upload

4.2.3.4 Uploading Grassland Mowing configuration file

An example and default configuration file is provided in Appendix E.12 Grassland mowing configuration file.

The Grassland mowing configuration can be uploaded through the section "L4B configuration" of the "Upload file" tab (Figure 4-8).
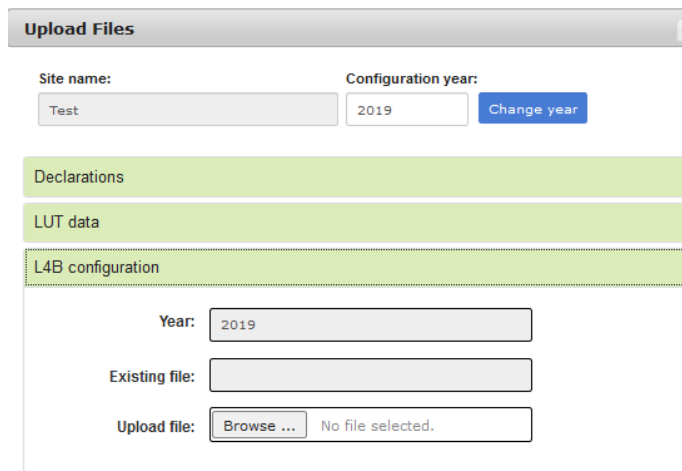
Figure 4-8. Uploading L4B configuration

4.2.3.5 Uploading Agricultural Practices configuration file

An example and default configuration file is provided in Appendix E.13 Agricultural practices configuration file. The Agricultural Practices configuration file can be uploaded through the section "L4C configuration" of the "Upload file" tab (Figure 4-9).
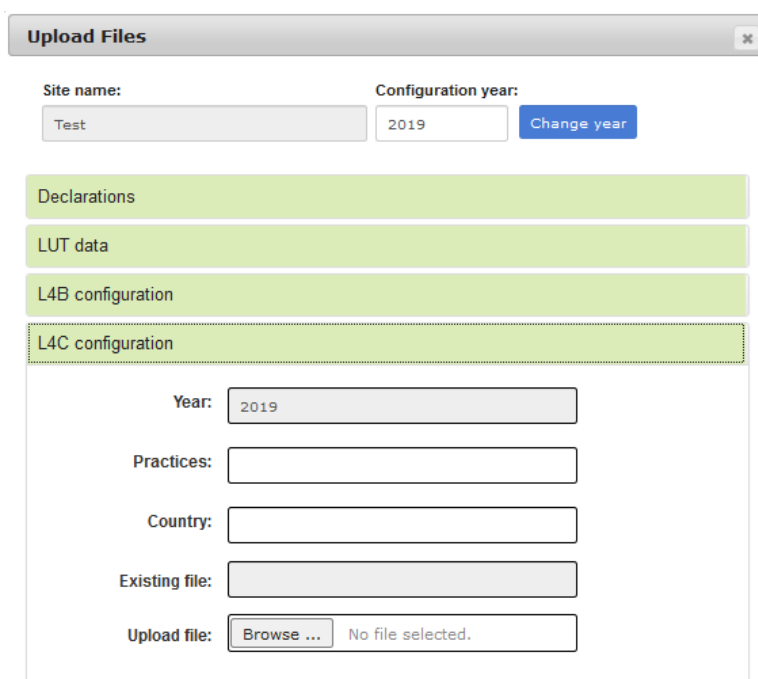
Figure 4-9. Uploading L4C configuration file

During the upload of the L4C configuration file, the following information needs to be provided:

- **Practices**: the list of practices to be followed, comma separated. Example: "CC, NA" or "CC, FL, NFC, NA" where:
    - o CC stands for Catch Crop EFA practices;
    - o FL stands for Fallow Land EFA practices;
    - o NFC stands from Nitrogen Fixing Crops EFA practices;

o NA stands for harvest only (parcels with no EFA practice declared);

- **Country**: any country or region code corresponding to the monitoring area:
  - o It will be used in the products names to facilitate the identification;
  - o In the case of Fallow Land and Nitrogen Fixing Crops monitoring, the Time Series Analysis (TSA) algorithms associated to these two practices were adapted during the project phase to the specific rules of the following countries. If one of these country codes are defined in this section, the specific version of the algorithm corresponding to the country is used:
    - ▪ LTU, CZE or ITA for Fallow Land monitoring;
    - ▪ CZE, ITA, ESP or ROU for Nitrogen Fixing Crops monitoring;

- **Upload file**: the L4C configuration file.

IMPORTANT NOTE: in the version 2.0, it is not possible to configure the parameters related to the tillage detection. The tillage detection has only 2 parameters which are currently set to their default values. In the next version, they will be externalized in the configuration file and will appear as distinct parameters in the configuration file to upload.

4.2.3.6 Uploading Agricultural Practices input tables files

Once the configuration file is uploaded, the practices files for the configured practices can be uploaded using the "L4C XX practices infos" sections of the "Upload file" tab. Figure 4-10 gives an example for the Catch Crops, with the "L4C CC practices infos". For each monitored practice, a practice table must be imported. An example of such a table is given in Appendix E.14 Agricultural practices table file. Note that there is no specific information to be provided for tillage detection. Information is the same as for the harvest detection.
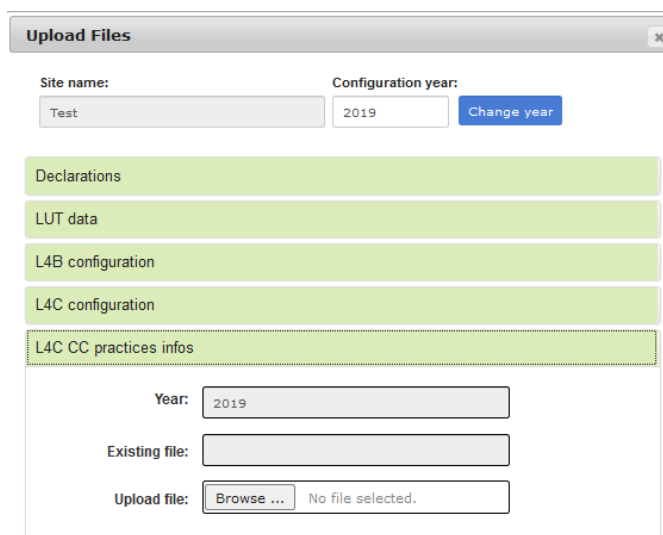
Figure 4-10. Uploading L4C practices file

4.2.4 Optionally adapt the main processor parameters

As introduced in sections 2.1 and 2.2, the system is made of 4 main processors, allowing producing vegetation status indicators, crop type maps, grassland mowing and agricultural practices products. Each

of these processors is parameterized by-default when the system is installed. The automated mode will be run using this **by-default parameterization** except if the user wants to change them. The change of the by-default parameters can be performed by using the "sen2agri-config" application or by manually changing the parameters values in the database (see Appendix B - Sen4CAP database tables for manual configuration and Appendix E – Advanced system configuration items). It should be noted that **this by-default parameterization were found to be the best generic values** after a consistent benchmarking carried out during the project.

The parameters of each processor are presented in detail in the section 4.4 about manual operations.

4.2.5 Activate or change the site parameters

If the site was not activated during the season definition, it can be "Enabled" at any time by navigating in the "**Sites**" tab and accessing to the Site Edition dialog by pressing the "**Edit**" button (Figure 4-11).
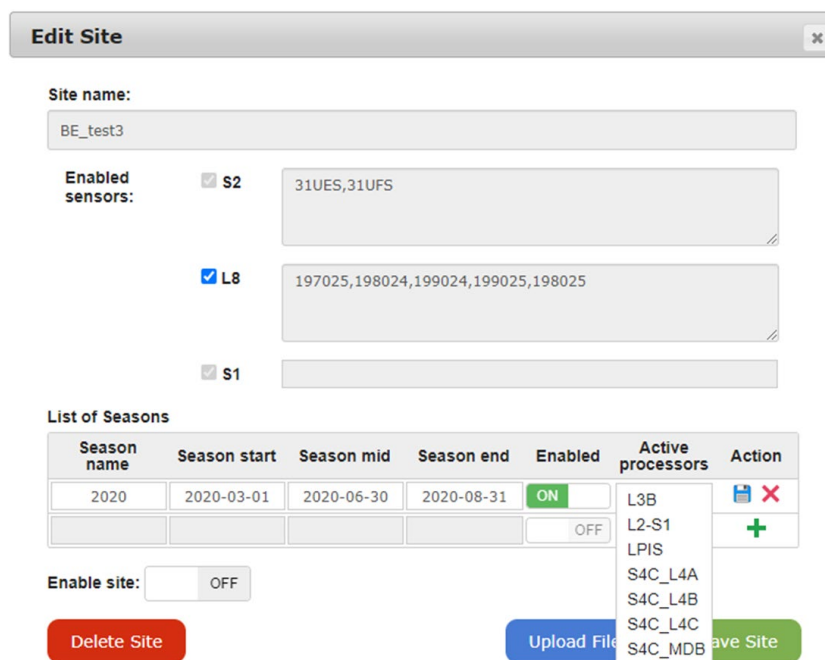
Figure 4-11. Edit site window allowing users modifying site parameters and/or activating it

This dialog box allows activating the site, modifying its season and uploading or modifying the subsidy application or the other processor configuration files.

Seasons can be edited using the button ✏ or deleted using 🗑. The list of activated processors for a certain season can be viewed by moving the mouse over the text *"hover to reveal"* but not modified.

The user has the possibility to enable or disable the L8 downloads by unchecking the "L8" button from the "Enable sensor" area.

Please note that there exist two levels of enabling options:

- At season level: the user can enable or disable different seasons of a site;

- At site level: the site, with all seasons (if more than one season has been defined) can be enabled or disabled at once.

4.2.6 Delete a site

The user has the possibility to delete a site at any time by navigating in the "**Sites**" tab and accessing to the Site Edition dialog by pressing the "**Edit**". From this dialog box the user should press the button "**Delete Site**" (Figure 4-11).

By default, this dialog box allows the user to delete the site with all associated products.

But, the user can keep some products of this site by unchecking the boxes associated with the products. Then the user will select the button "**Confirm Delete Site**" (Figure 4-12).
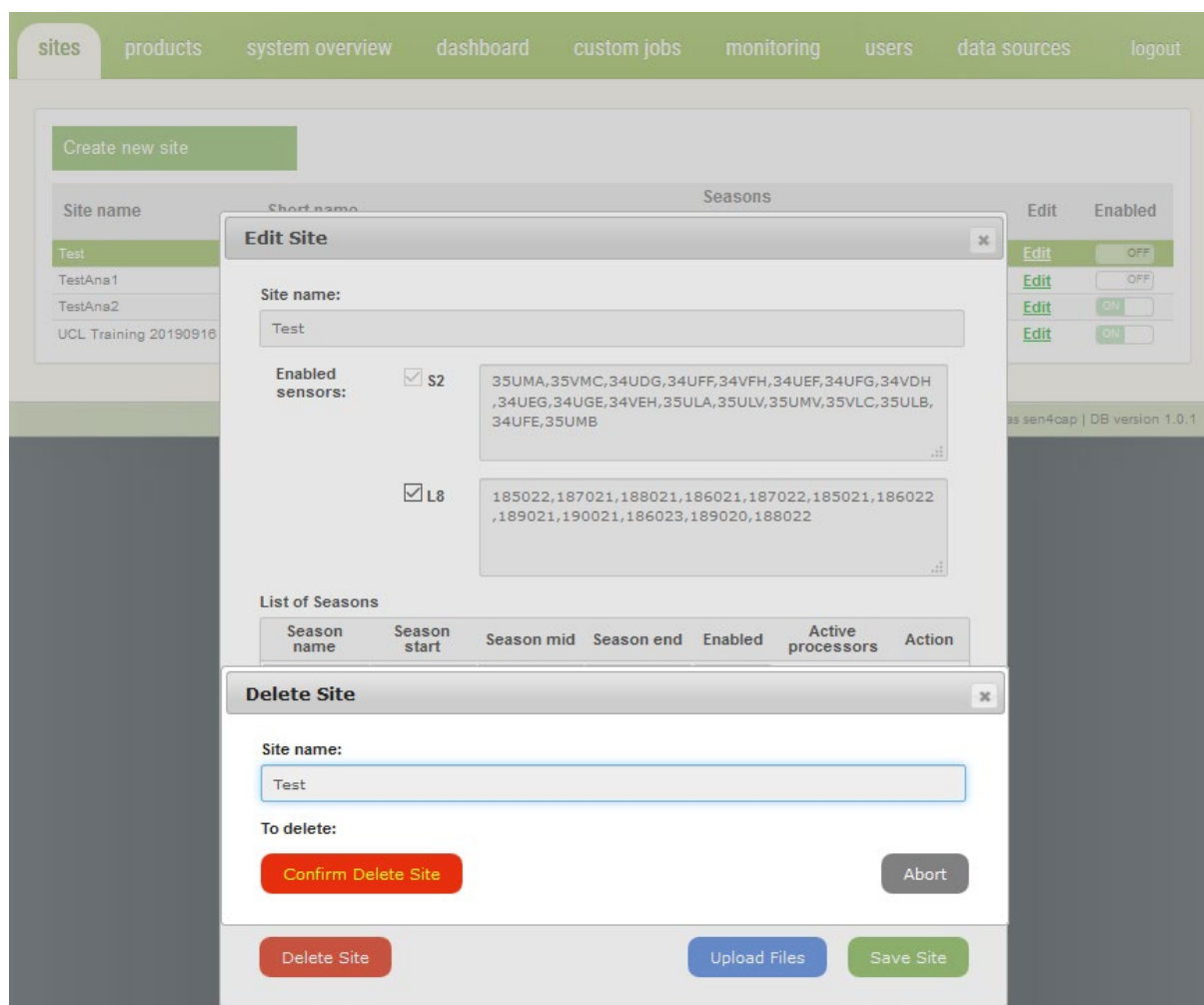
Figure 4-12. Site page of the Sen4CAP interface, to delete a site

4.2.7 Monitor the downloading

The Monitoring tab of the web interface gives various information concerning the downloading of the data (Figure 4-13). In the "Download statistics" bar, you can find:

- The number of images that have been downloaded (and %);
- The number of images that are currently being downloaded (and %);
- The number of images from which the download has failed (and %);
- The number of images from which the download has failed but will be retried (and %).

It also gives you the estimated number of products still to be downloaded.
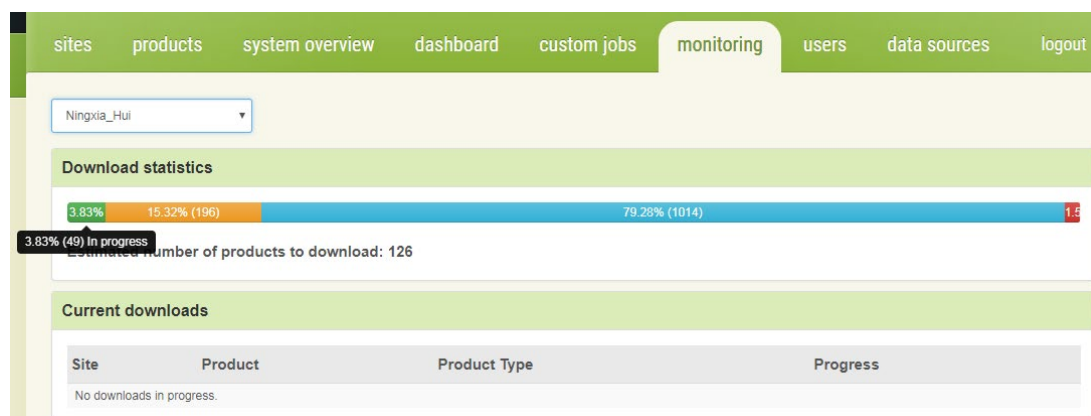
Figure 4-13. The "monitoring" tab of the web interface - A view to the downloading

4.2.8 Monitor the processing

The Execution Dashboard enables the user to monitor the Sen4CAP system. It consists of a web page that dynamically loads data from the Sen4CAP HTTP listener service.

The System Overview tab of this Dashboard (Figure 4-14) presents the jobs that are currently being executed by the system as well as the performance of the server(s).
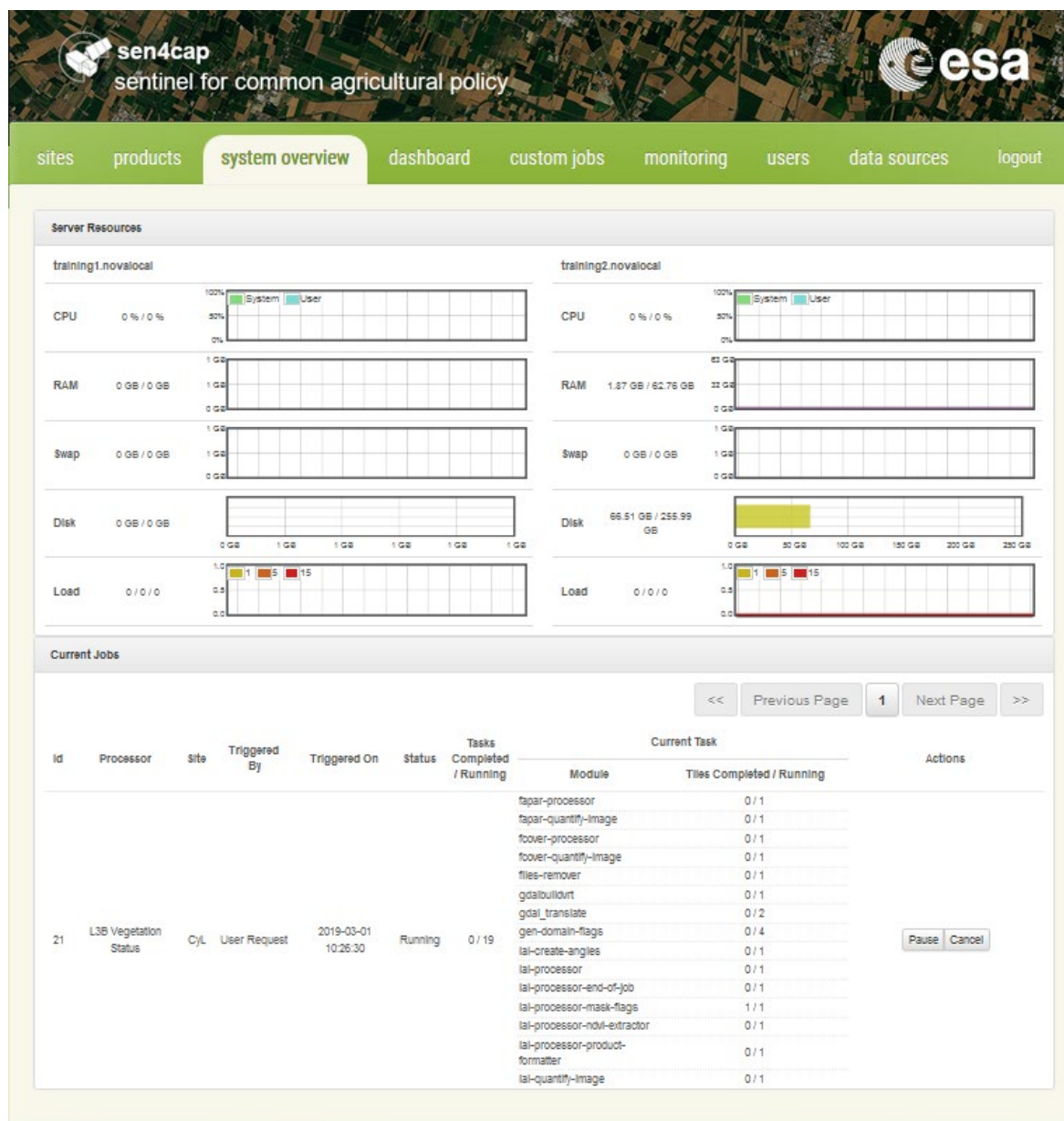
Figure 4-14. Sen4CAP Execution Dashboard – System Overview

For each server (top of the system overview), the following resources are presented:

- CPU (current value and a graph showing the evolution over the last 15 minutes);
- RAM (current value and a graph showing the evolution over the last 15 minutes);
- Swap (current value and a graph showing the evolution over the last 15 minutes);
- Disk – the used disk space from the available disk space;
- Load (current value and a graph showing the evolution over the last 15 minutes) – the load on the server is provided by the operating system in the form of 3 numbers that represent the 1/5/15 minutes averages of the *run-queue length*: the sum of the number of processes currently running plus the number of processes waiting (queued) to be run.

For every job, the following information is presented:

- Id;
- Processor;
- Site;
- Triggered By - the possible values are:
  o "Scheduler request" – when the job was created by the automated scheduling mechanism;
  o "User request" when the job was created with a custom job request;
  o "Available product" when the job was created due to the occurrence of a product that the corresponding processor is monitoring;
- Triggered On;
- Status (of the job) - the possible values are:
  o Running;
  o Canceled;
  o Paused;
- Tasks Completed/Runing;
- For the current task:
  o Module;
  o Tiles Completed/Running;
- Actions (depending on the status):
  o As possible actions for a job, we have "Pause", "Resume" and "Cancel";
  o "View config" – will show the parameters used when the job was submitted.

Each job has associated two buttons:

- Pause or Resume:

  o The button "Pause" is displayed when the current job is running and allows the user to interrupt the job execution

  o The button "Resume" is displayed when the current job is interrupted

    and allows the user to resume the job execution

- Cancel : This button allows the user to cancel the job execution

4.2.9 Where to find the markers from the markers databases

4.2.9.1 List of Markers Databases

As explained in section 2.2.2, four types of markers are processed by parcel, along the season. Each of these types correspond to a specific name: MDB1, MDB2, MDB3 and MDB_L4B. Moreover, the MDB_L4B type is divided itself in four different types. The reference name (inside the system) of these different markers types is given below. It is important because you will have to refer to these names to retrieve the markers values through the API service. The complete list of markers inside the database is given in section 2.2.2 and can also be consulted through an API service.

- **s4c_mdb1**: basic single date markers, corresponding to MDB1 from section 2.2.2;
- **s4c_mdb2**: ratio single date markers, corresponding to MDB2 from section 2.2.2;
- **s4c_mdb3**: harvest detection markers, corresponding to MDB3 from section 2.2.2;
- **s4c_mdb_l4a_opt_main**: L4A optical main markers, corresponding to MDB4, optical, every 10 days markers, based on 10-meters resolution data from section 2.2.2;

- **s4c_mdb_l4a_opt_re**: L4A optical red-edge markers, corresponding to MDB4, optical, every 10 days markers, based on 20-meters resolution data from section 2.2.2;
- **s4c_mdb_l4a_sar_main**: L4A SAR main markers, corresponding to MDB4, SAR, every week (7 days) markers, based on 20-meters resolution data from section 2.2.2;
- **s4c_mdb_l4a_sar_temp**: L4A SAR temporal markers, corresponding to MDB4, SAR, every month, every 2 months and over the whole monitoring period markers, based on 20-meters resolution data from section 2.2.2.

IMPORTANT NOTE: in version 2.0, **only the markers from the Marker Database MDB1, MDB2 and MDB3 are processed by the MDB processor**. The markers from the Marker Database MDB_L4A are processed by the L4A crop type processor. Still, these MDB4 markers are available in the same way as the others, but they will only be available after having launched a L4A crop type job.

4.2.9.2   Access the markers inside the system (.csv and .ipc tables)

Once they are processed, the markers are stored in csv and ipc tables inside the system. The location of these tables is given below:

- s4c_mdb1:
  /mnt/archive/<site_short_name>/s4c_mdb1/SEN4CAP_MDB1_SXX_V<start_date>_<end_date>_<creation_time>
- s4c_mdb2:
  /mnt/archive/<site_short_name>/s4c_mdb1/SEN4CAP_MDB2_SXX_V<start_date>_<end_date>_<creation_time>
- s4c_mdb3:
  /mnt/archive/<site_short_name>/s4c_l4c/SEN4CAP_MDB3_SXX_V<start_date>_<end_date>_<creation_time>
- s4c_mdb_l4a_opt_main:
  /mnt/archive/<site_short_name>/s4c_l4a/SEN4CAP_MDB_L4A_OPT_MAIN_V<start_date>_<end_date>_<creation_time>
- s4c_mdb_l4a_opt_re:
  /mnt/archive/<site_short_name>/s4c_l4a/SEN4CAP_MDB_L4A_OPT_RE_V<start_date>_<end_date>_<creation_time>
- s4c_mdb_l4a_sar_main:
  /mnt/archive/<site_short_name>/s4c_l4a/SEN4CAP_MDB_L4A_SAR_MAIN_V<start_date>_<end_date>_<creation_time>
- s4c_mdb_l4a_sar_temp:
  /mnt/archive/<site_short_name>/s4c_l4a/SEN4CAP_MDB_L4A_SAR_TEMP_V<start_date>_<end_date>_<creation_time>

4.2.9.3   Access the markers through the API service

Once they are processed, the markers are accessible through an API service. Two methods enable to get the names of the markers and their values. These two methods are described below.

Requesting the names of the markers

This method allows retrieving the markers names for a given markers type, available for a site and a year.

- Model:

http://{ip_address}:{port}/markers/names?site={site_short_name}&productType={markers_type}&year={year}

- Parameters (Table 4-1)

Table 4-1. Parameters of the function to be used to retrieve markers names through the API service

Parameter	Description	Mandatory
site: *String*	The site short name as it appears in the sites tab	Yes
productType: *ProductType*	The markers product type identifier. Available marker product types: s4c_mdb1 – basic single date markers s4c_mdb2 – ratio single date markers s4c_mdb3 – harvest detection markers s4c_mdb_l4a_opt_main - L4A optical main markers s4c_mdb_l4a_opt_re - L4A optical red-edge markers s4c_mdb_l4a_sar_main - L4A SAR main markers s4c_mdb_l4a_sar_temp - L4A SAR temporal markers	Yes
year: *int*	The year of the product	Yes

- Example:

http://10.36.100.177:8080/markers/names?site=be_test2&productType=s4c_mdb1&year=2020

```
{"data":["mean_AMP_VH_110","mean_AMP_VV_110","mean_COHE_VH_110","mean_COHE_
VV_110","mean_FAPAR","mean_FCOVER","mean_LAI","mean_NDVI","stdev_AMP_VH_110
","stdev_AMP_VV_110","stdev_COHE_VH_110","stdev_COHE_VV_110","stdev_FAPAR",
"stdev_FCOVER","stdev_LAI","stdev_NDVI","mean_AMP_VH_037","mean_AMP_VV_037"
,"mean_COHE_VH_037","mean_COHE_VV_037","stdev_AMP_VH_037","stdev_AMP_VV_037
","stdev_COHE_VH_037","stdev_COHE_VV_037","mean_AMP_VH_161","mean_AMP_VV_16
1","mean_COHE_VH_161","mean_COHE_VV_161","stdev_AMP_VH_161","stdev_AMP_VV_1
61","stdev_COHE_VH_161","stdev_COHE_VV_161","mean_AMP_VH_088","mean_AMP_VV_
088","mean_COHE_VH_088","mean_COHE_VV_088","stdev_AMP_VH_088","stdev_AMP_VV
_088","stdev_COHE_VH_088","stdev_COHE_VV_088"],"message":null,"status":"SUC
CEEDED"}
```

Requesting the markers values

This method allows retrieving the values of the selected markers, available for a product type, a site and a year.

- Model:

http://{ip_address}:{port}/markers?site={site_short_name}&productType={markers_type}&year={year}&markers={marker_name}&parcels={parcel_NewID}

- Parameters (Table 4-2)

Table 4-2. Parameters of the function to be used to retrieve markers values through the API service

Parameter	Description	Mandatory
site: *String*	The site short name as it appears in the sites visualization tab	Yes
productType: *ProductType*	The markers product type identifier	Yes
year: *int*	The year of the product	Yes
markers: *String*	Comma separated list of markers names obtained with a /marker/names query	Yes
from: *String*	Filtering start date for markers values retrieval	No
to: *String*	Filtering end date for markers values retrieval	No
roi: *String*	ROI as WKT for filtering parcels	No
parcels: *String*	Comma separated string for filtering parcels (NewID)	No

- Example:

http://10.36.100.177:8080/markers?site=be_test2&productType=s4c_mdb1&year=2020&markers=mean_NDVI&parcels=7500

```
{"data":{"dates":["2020-05-02","2020-05-05","2020-05-07","2020-05-
15","2020-05-17","2020-05-20","2020-05-25","2020-05-27","2020-05-30","2020-
06-01","2020-06-06","2020-06-21","2020-06-24","2020-06-
26"],"parcels":[{"id":7500,"markers":{"mean_NDVI":[0.0,777.9825,0.0,764.736
8,0.0,772.9649,695.8246,0.0,640.0351,0.0,0.0,0.0,629.4211,0.0]}}]},"message
":null,"status":"SUCCEEDED"}
```

s4c_mdb3: harvest detection markers

There is a special case concerning the harvest detection markers which have in the L4C processor the values TRUE, FALSE, NA, NA1 or NR (see the Agricultural Practices ATBD for the meaning of these values). These values are translated in the marker returned values using the following mapping to integer values (Table 4-3).

Table 4-3. Values of L4C harvest markers in the markers database

L4C Value	Marker Value
TRUE	1
FALSE	0
NA	-1
NA1	-2
NR	-3

4.2.10   Where to find output products

The "Products" tab of the Execution Dashboard (Figure 4-15) shows a tree view with the products that are available. On clicking the site node, all the products available for this site become visible. If the icon of the product is clicked, the overview is visible on the map. Conversely, if the download symbol in the right of the product is clicked, the product is downloaded locally so that it can be visualized with another application like QGis or SNAP.
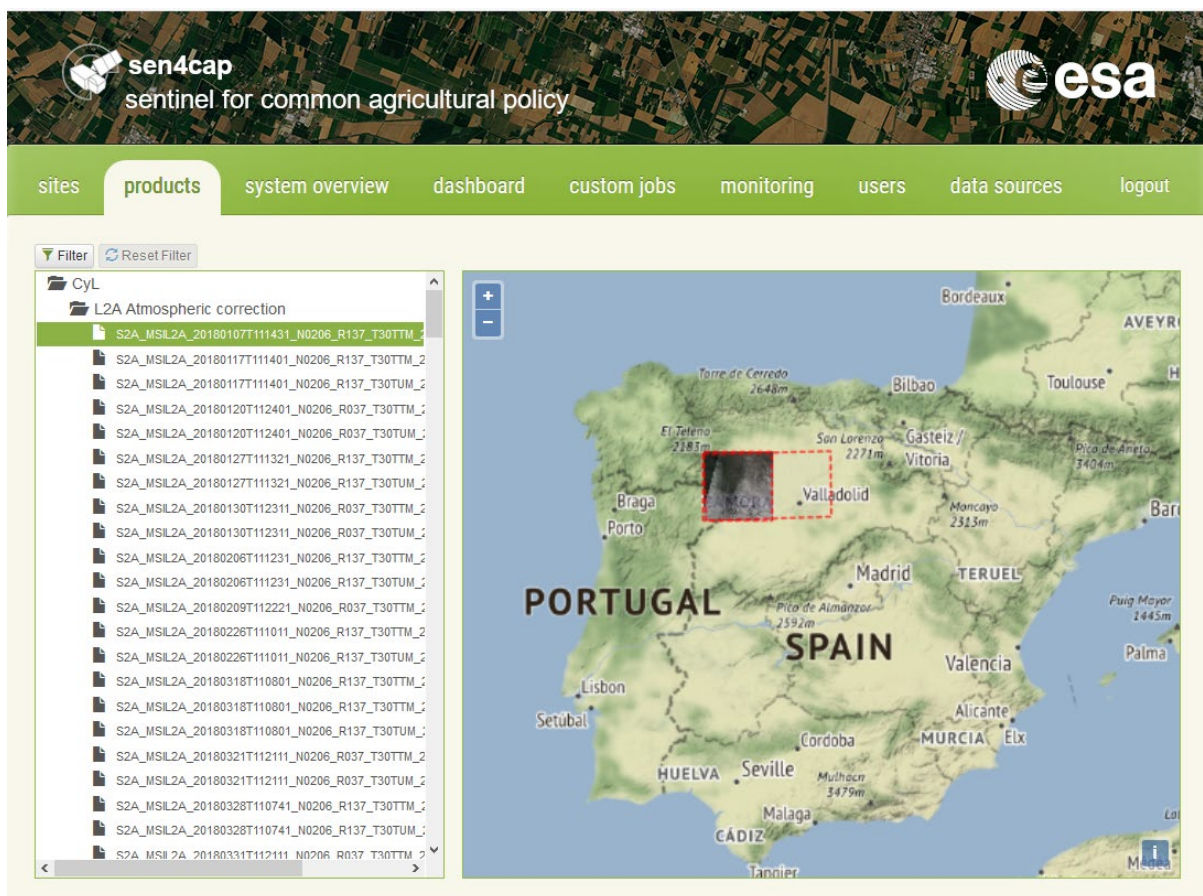
Figure 4-15. Sen4CAP Execution Dashboard – Product Availability

The paths where the products can be found are the following ones:

- Normally, when working on DIAS, the L1C/L1T products are used directly from the EODATA repository. Nevertheless, if they are not available here after a certain timeout, they will be downloaded locally. In this case, the L1C / L1T products found in */mnt/archive/dwn_def/* (see configuration parameters "Write directory for L8" and "Write directory for S2" from Appendix E.7 Sen4CAP Downloader)

- Optical L2A products (Sentinel 2 and Landsat 8 L2A products) found in */mnt/archive/maccs_def* (see configuration parameter *"demmaccs.output-path"* from Appendix E.8 Sen4CAP Demmaccs_launcher)

- SAR L2A products (Sentinel 1 L2A products) found in */mnt/archive/{site}/l2a-s1,* where *{site}* represents the short name of the site.

- L3A products stored in a folder having the short name of the site[*], which can be found in *"/mnt/archive"*. For example, *"/mnt/archive/romania/l3a"* or *"/mnt/archive/romania/l3a"*.

- L4A products stored in a folder having the short name of the site[*], which can be found in *"/mnt/archive"*. For example, *"/mnt/archive/lithuania/l4a"*.

- L4B products stored in a folder having the short name of the site[*], which can be found in *"/mnt/archive"*. For example, *"/mnt/archive/italy/l4b"*.

- L4C products stored in a folder having the short name of the site[*], which can be found in *"/mnt/archive"*. For example, *"/mnt/archive/czechia/l4c"*.

4.2.11 Statistics

The "Statistics" tab of the Execution Dashboard shows a list of reports that the user can access in order to analyze processing statistics and daily acquisition numbers (Figure 4-16).
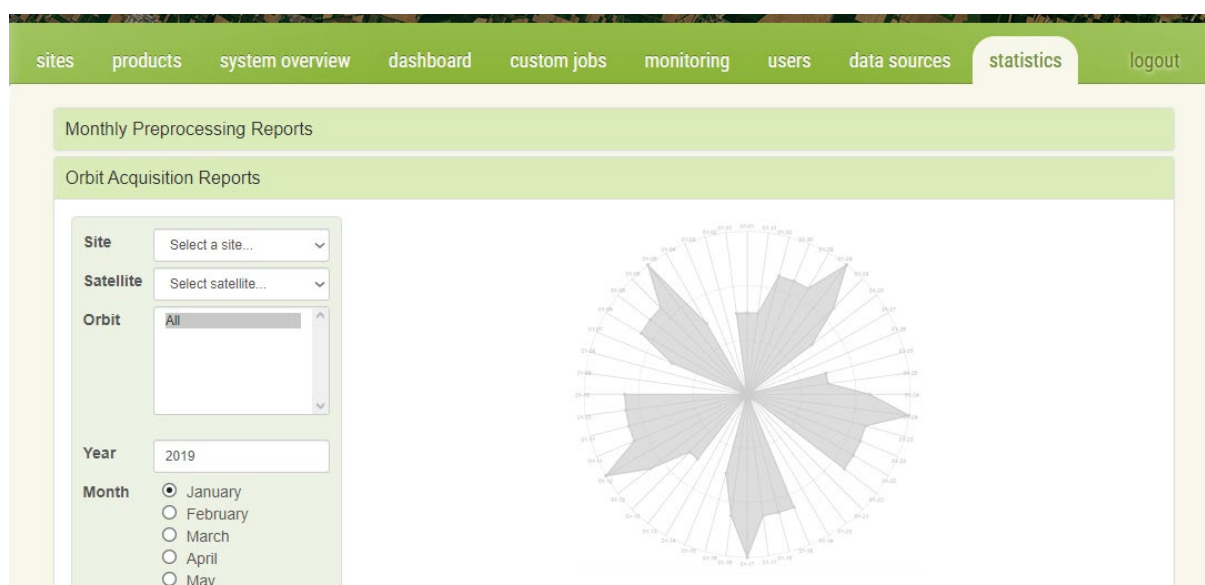
Figure 4-16. Statistics" tab of the Execution Dashboard

4.2.11.1 Monthly Processing Reports

As the name suggests, each report shows operations for each day in a month. These reports can be customized using the left side filters panel where the user can choose a set of months in a given year to produce that many charts, all for a selected site, satellite and orbit/s.

The reports will depict Acquisitions as a continuous line with values shown on the right vertical axis, while all other operations (Processed, Errors, Cloudy, …) will be depicted as discrete vertical bars of different colors with values shown on the left vertical axis (Figure 4-17).
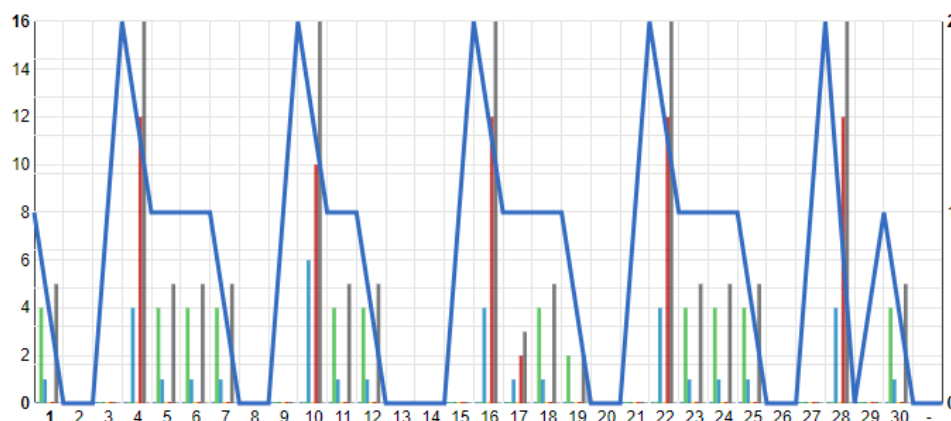
Figure 4-17. Monthly reports of pre-processing operations

For better readability and to facilitate comparison between different months, all reports will be generated as a stack of charts, each with 31 values on the horizontal axis. For months with less than 31 days, empty sequences will be added at the end (Figure 4-18).
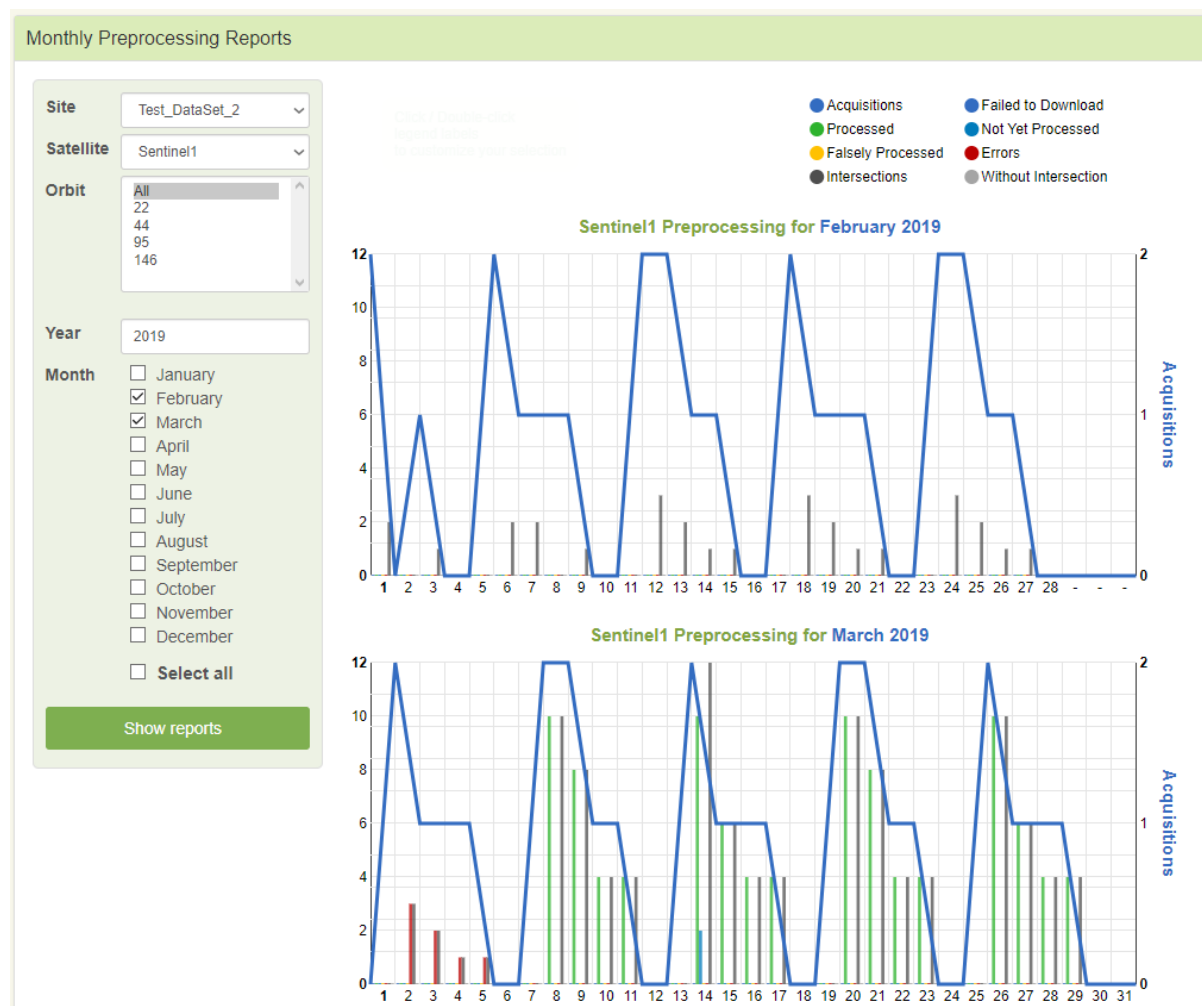
Figure 4-18. Monthly reports for months with less than 31 days (with empty sequences at the end)

For all reports, a single legend is generated at the top of the stack. The legend is interactive, meaning that the user can customize the selection of values to be displayed, by clicking or double-clicking on its labels. Figure 4-18 shows an example where the user only selected 3 operations: Acquisitions (blue line), Processed (green bar) and Errors (red bar). Selected operations are depicted in the legend as filled colored circles, while all other as empty circles. A message at the left side of the legend reminds the user that it's possible to customize the report.
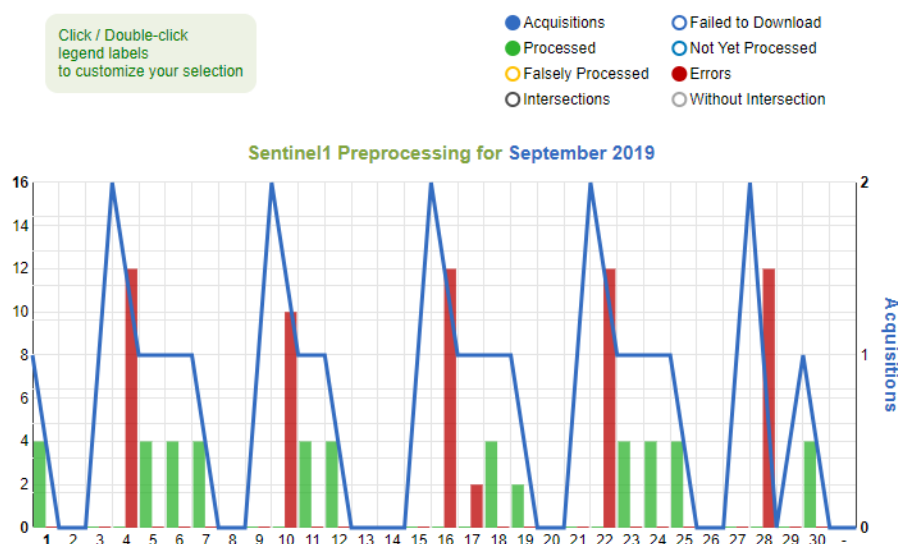
Figure 4-19. Interactive legend of the processing reports

While hovering a report chart, a vertical selection bar is displayed which spans over an entire day to include all elements belonging to that day. This allows the user to easily identify the values corresponding to a single day. Alongside this selection bar, a textbox type tooltip is displayed with all the necessary information for the user to read its selection (Figure 4-20).
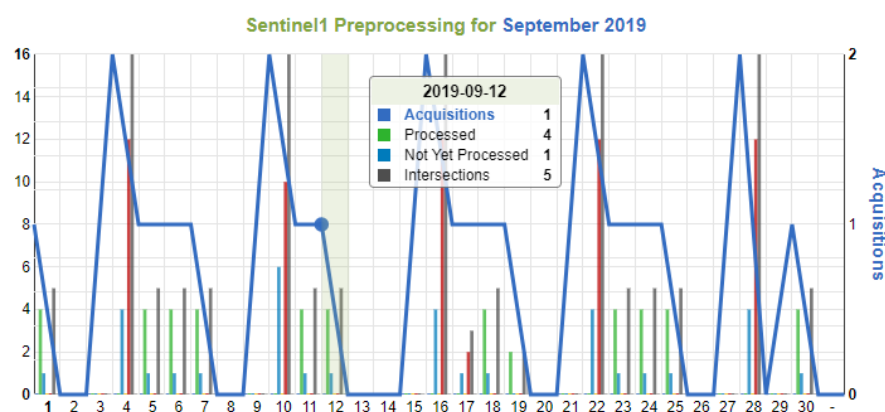
Figure 4-20. Text box displayed on the processing reports to summarize key information

4.2.11.2 Orbit Acquisition Reports

This report is generated as a "spider web" type chart (Figure 4-21), where as opposed to the previous type of reports, only one operation is shown, which is the "acquisitions per day". This report can be customized using the left side filters panel where the user can choose either a single month in a given year, or a custom time window, to produce a single "spider web" chart for a selected site, satellite and orbit/s.
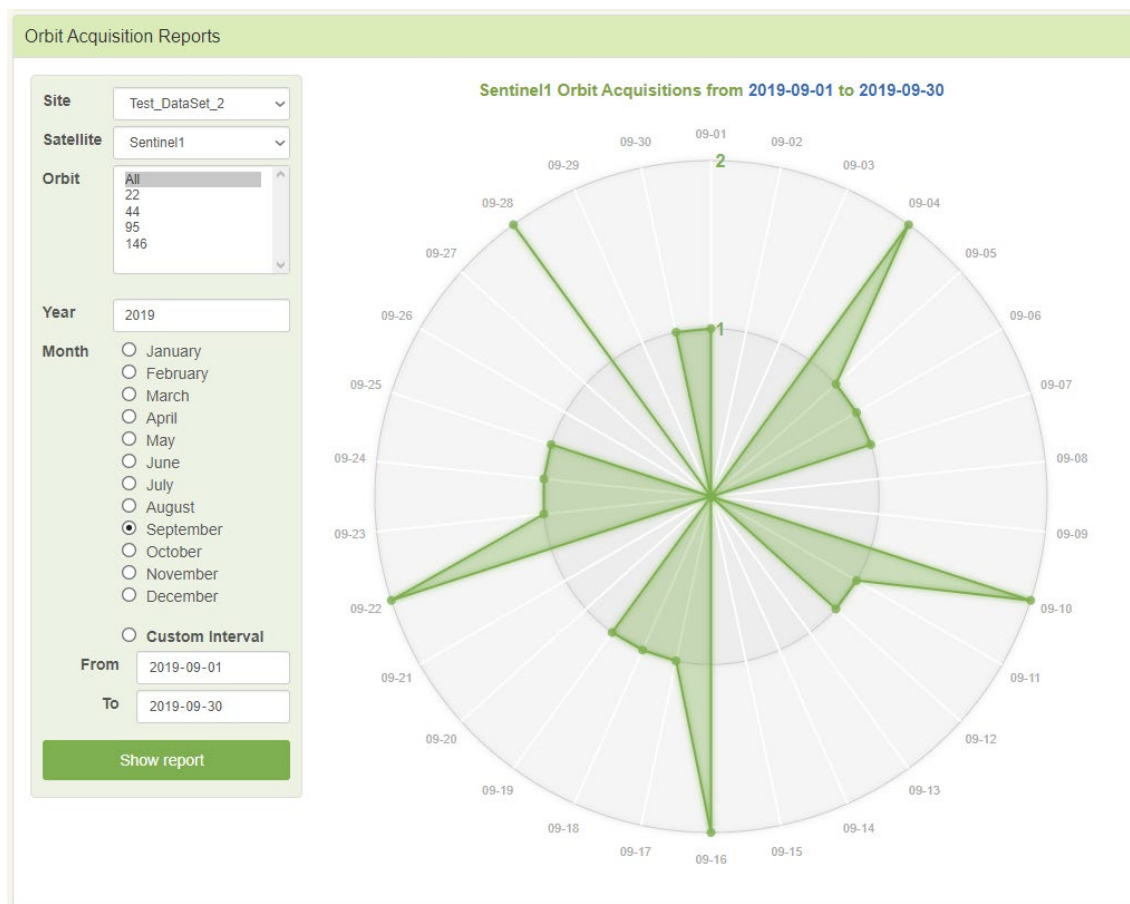
Figure 4-21. Orbit Acquisition Report presented as a "spider web" type chart

The Acquisitions values are shown as concentric circles, as many as the maximum value for the selected time frame (Figure 4-22). The days are shown as labels on the circumference of the top-level circle. If the number of selected days exceeds 60, the number of labels shown on the circumference will be reduced so they don't overlap. The Acquisition values however will all be shown.

To identify which day corresponds to an Acquisition point, the user can hover that point, then a selection will be displayed with all necessary information. All labels on the circle will be hidden to avoid text cluttering.
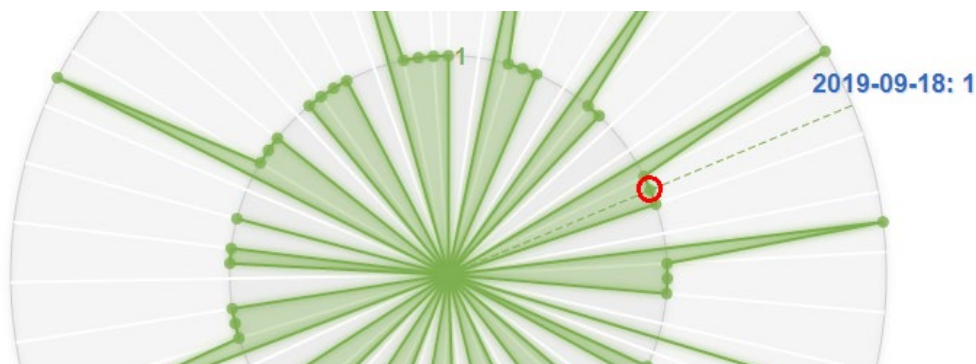
Figure 4-22. Acquisition values display in the Orbit Acquisition Report, as well as the acquisition days

It is recommended to avoid selecting a large time frame, or the chart will become hard to read. The following example shows one-year worth of Acquisition data. It's still usable, but hard to read (Figure 4-23).
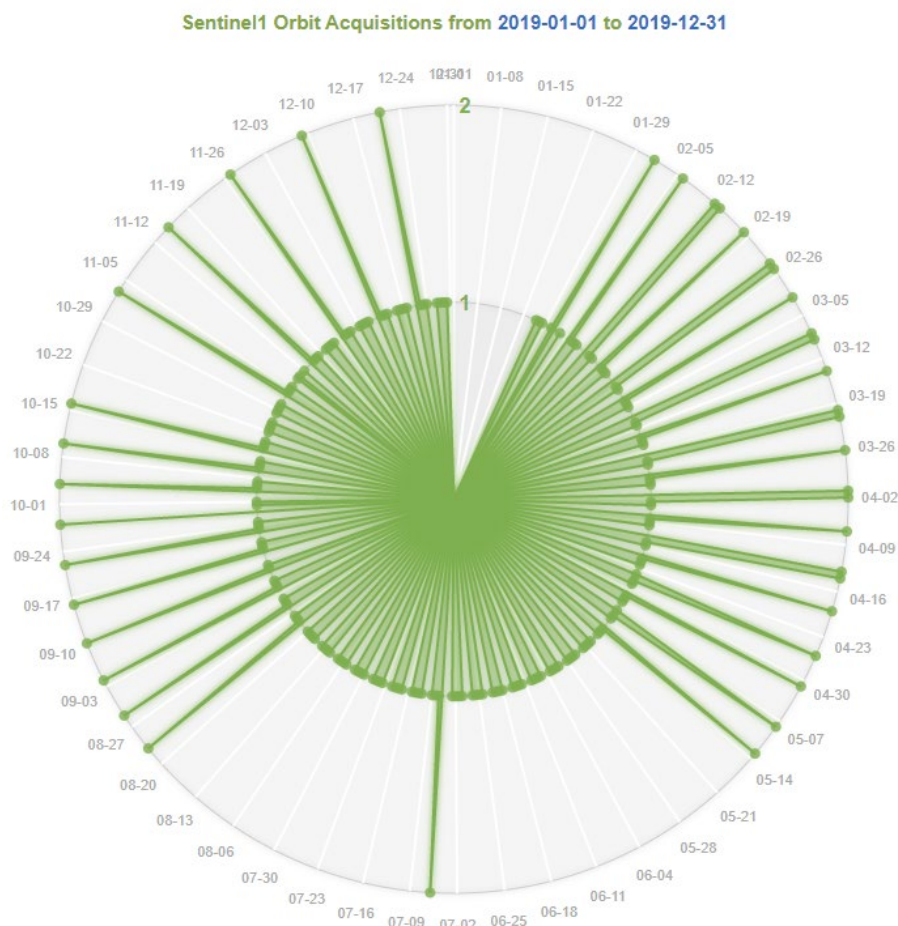
Figure 4-23. Example of Orbit Acquisition Report over a one-year period
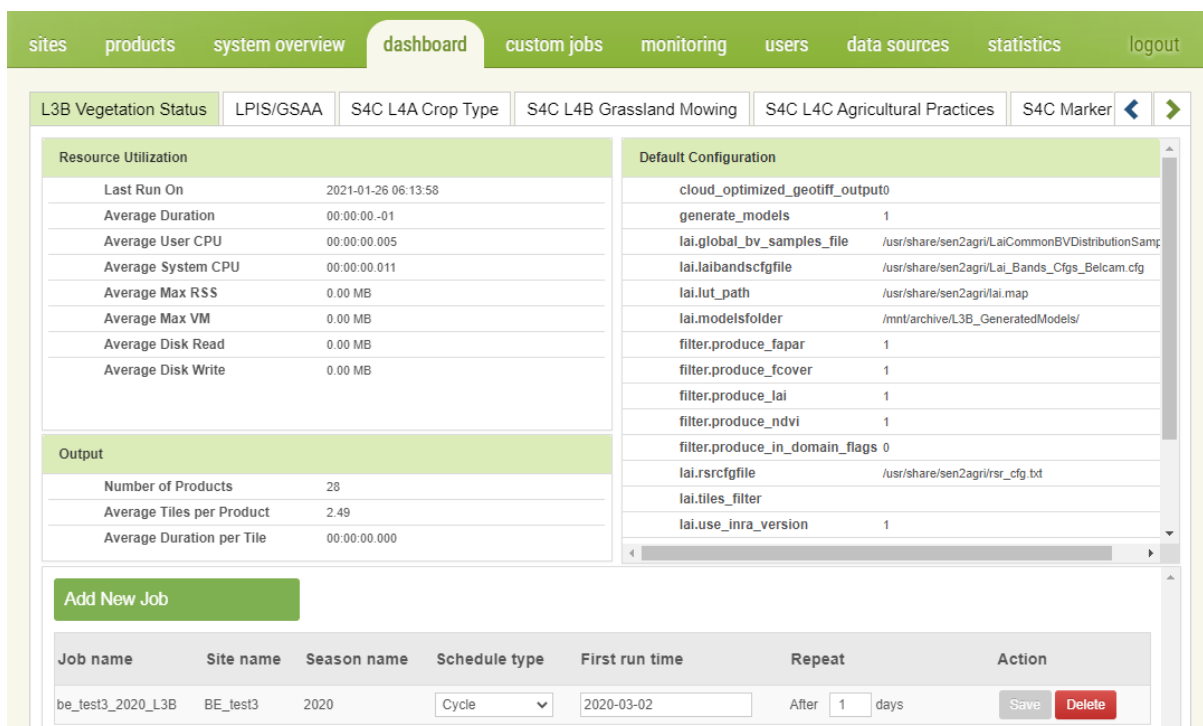
4.3 How to add scheduled jobs

The "dashboard" tab provides the possibility to define scheduled jobs, that is executions that are performed at certain moments in time (one-time executions or periodic executions) and that use the products available at that moment in time, according also to the configuration defined in the database.

The scheduled tasks can be added for all the processors excepting the L2A processor that is a built-in execution module. The scheduled tasks can be added from the "Dashboard" page of the Sen4CAP website.

The scheduled tasks will use the default parameters configured in the database for that processor. The default parameters used can be seen also in each tab of the processor from the "Dashboard" page (Figure 4-24).

Figure 4-24. Processor parameters

Adding a scheduled task is similar for all processors.

In order to add a scheduled task the following steps should be followed:

- Press the "Add Job" button from the tab corresponding to the processor.

  A new entry will be added in the list of scheduled tasks (Figure 4-25):
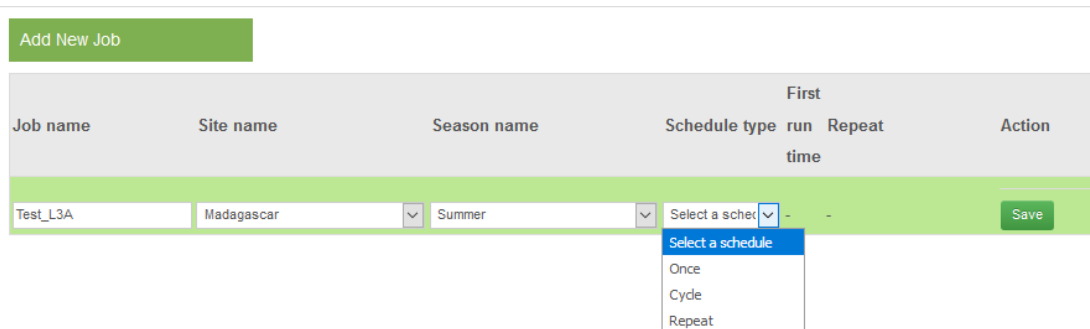
Figure 4-25. Adding a new scheduled job

- In the "Job Name" field add an identifier for the added task (ex. "myjob"). The job name should not contain space and should start with a capital letter.

- In the "Site" field should be selected one of the existing sites.

- From the "Schedule" field, the following options are available:

  o "Once" – this options specifies to the scheduled task to run only once at the specified date. Selecting this option, a new field "Date" will be added to the entry where the user can select the date when the scheduled task should run (Figure 4-26):
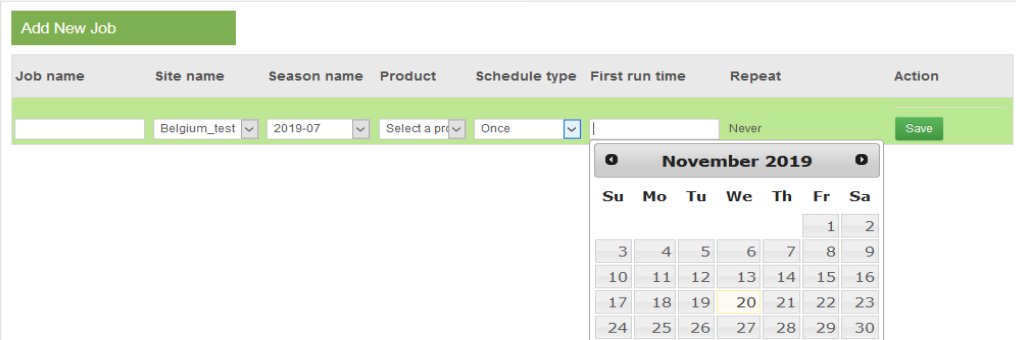
Figure 4-26. Adding "Once" scheduled type

o "Cycle" – this option allows to schedule a task at a certain date and then to reschedule the task after a number of specified days (Figure 4-27). For example, a configuration with the start date of 1st of March and "Repeat after" set to 4 days will execute the scheduler on 1st of March then on 5th of March, then on 10th of March and so on.
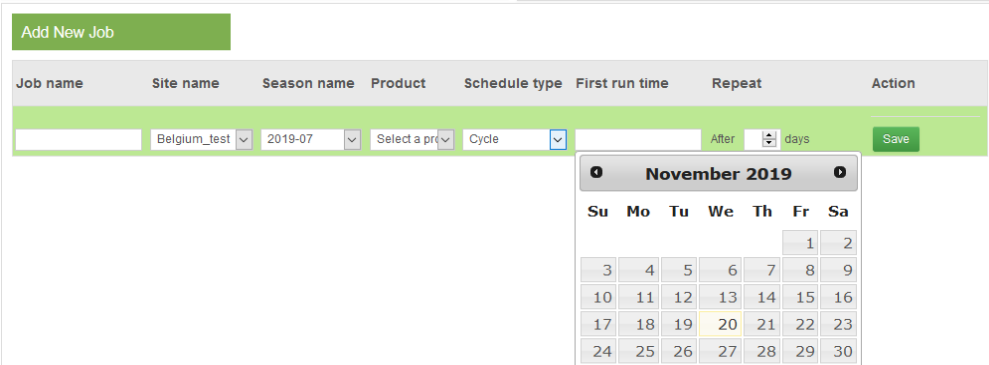
Figure 4-27. Adding "Cycle" scheduled type

o "Repeat" – this option allows to schedule a task at a certain date and then to reschedule the task at a certain day of the month (Figure 4-28). For example, a configuration with the start date of 1st of March and "On every" set to 4 will execute first on 1st of March, then on 4th of March, then on 4th of April, then on 4th of May and so on.
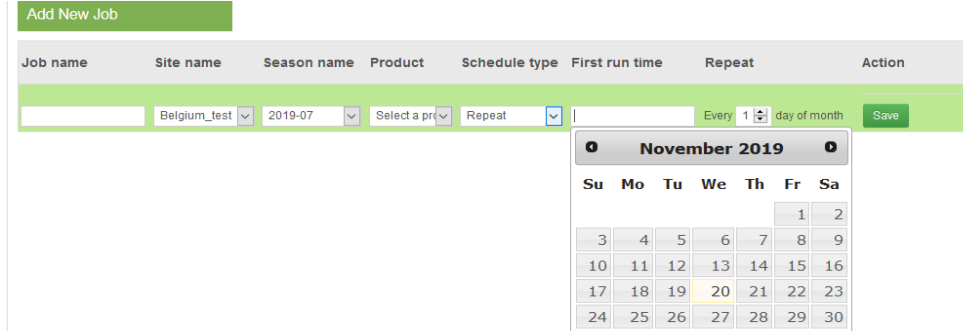
Figure 4-28. Adding "Repeat" scheduled type

In order to delete an added scheduled job, the "Delete" button can be used (Figure 4-29).
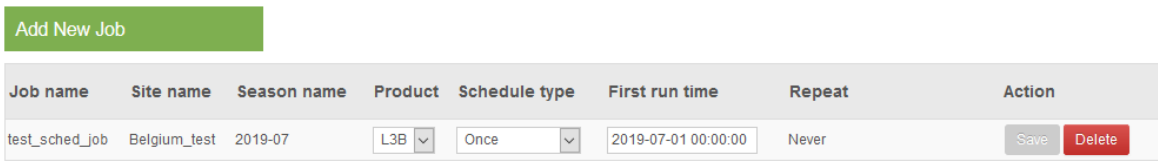
Figure 4-29. Delete scheduled job

4.4 Perform manual operations

The manual operations of the Sen4CAP system can be done by:

- Executing the processors from the command line using a specific Python scripts for each processor. This execution is available both in automated and manual mode installation of the system. Manually invoking the processors from the command-line does not use the SLURM scheduler. These manual executions from command line are intended for special scenarios like SNAP integration in which SLURM might not be available. When running in manual mode, some of the potential parallelism present in a job is not exploited. If needed, multiple manual jobs can be started at the same time;

- Executing the processors from the web interface. This execution is similar as with the command line invocation except that the execution will be performed using the orchestration mechanism that allows executions of parallel tasks and optionally on different machines and priorities (according the SLURM configuration). Though, this option is available only if the system was installed for an automated usage.

IMPORTANT NOTE: When running the system components in the manual mode, make sure that the output location is writable by the current user;

The following sections explain how to manually run the system for a set of operations.

4.4.1 Sentinel 1 pre-processing (S1 L2 processor)

The Sentinel 1 pre-processing of the SLC products is done automatically by the sen4cap-services module, by inspecting periodically the database and determining the products to be processed along with the previous products needed for computing the coherences.

The S1 pre-processing module cannot be run manually.

4.4.2 Atmospheric Correction (S2 L2A processor)

An atmospheric correction on a L1C and L1T products is performed using the L2A processor (also called *demmacs*) which is used in the automatic mode by the demmaccs_launcher processor. For each tile to process, the MAJA executable needs the output of DEM processor. The demmaccs processor automatically runs both DEM and MAJA processors for each tile. It has the parameters listed in Table 4-4.

Table 4-4. L2A processor (demmaccs) launching parameters

input		input L1C directory
-w	--working-dir	working directory
	--srtm	SRTM dataset path
	--swbd	SWBD dataset path
	--gipp-dir	directory where gipp files are to be found
	--maccs-launcher	the path where MAJA binary file is installed
	--processes-number-dem	maximum number of parallel processes which run DEM processor. The DEM processor handles one tile only, thus multiple DEM processors may be run in the same time to minimize the processing time. NOTICE: the value for this key should be handled with caution and should not exceed 60% from the server number of logical CPUs (default 20)
	--processes-number-maccs	maximum number of parallel processes which run MAJA processor. The MAJA processor handles one tile only, thus multiple MAJA processors may be run in the same time to minimize the processing time. NOTICE: the value for this key should be handled with caution and should not exceed 60% from the server number of logical CPUs (default 8)
	--maccs-address	MAJA has to be run from a remote host. This should be the ip address of the pc where MAJA is to be found
	--prev-l2a-tiles	the MAJA processor may be run in 2 modes: L2INIT and L2NOMINAL. The former will be used if there aren't any previously processed tiles by MAJA. The latter will be used if there are already previously processed tiles and can be used by MAJA to enhance the final result. The tiles should be inserted as a list separated by spaces
	--prev-l2a-products-paths	the path for each previously processed tile found in the –prev-l2a-tiles list. For each present tile there should be a corresponding path. The paths are separated by spaces
output		output location

4.4.2.1 Executing L2A processor from a terminal window

The L2A processor can be launched from terminal by using the script *demmacs.py*:

```
./demmaccs.py --working-dir /mnt/archive/temp_maccs/workingDir/ --srtm /mnt/archive/srtm/ -
-swbd /mnt/archive/swbd/ --processes-number-dem 20 --gipp-dir /mnt/archive/gipp/ --maccs-
launcher                                                      /opt/maja/3.2.2/bin/maja
/mnt/archive/Sen2Agri_DataSets/2015/Ukraine/EO/L8/LC81810252015240LGN00/
/mnt/archive/temp_maccs/
```

4.4.3 Subsidy application dataset preparation

The subsidy application dataset preparation step standardizes the original dataset in a way that it can be used by the different Sen4CAP processors. It also qualifies each parcel with a set of indicators or flags related to their geometry and area, and provides intermediate products which are used by the subsequent L4x processors.

4.4.3.1 Input data

The subsidy application dataset should be a ZIP archive with an ESRI Shapefile dataset containing the following information:

- o the parcel geometry (`POLYGON`, `MULTIPOLYGON`)
- o a column or set of columns that uniquely identify a parcel
- o a column or set of columns that uniquely identify a holding
- o a column that identifies the declared crop type

The column names don't matter, but they have to be entered in the website when uploading the dataset. The column types are important and they should match the type of the values stored within, e.g. if the crop codes are numeric, the column should have an integral type and there should be no padding spaces or zero digits. Text and even diacritics are supported too, provided they are encoded as UTF-8 with no BOM.

In addition to the subsidy application dataset, a dedicated crop code Look-Up-Table (LUT) must also be imported to perform the dataset preparation. The content and format of the crop code LUT is detailed in the Appendix F11. Crop Code LUT file structure.

4.4.3.2 Import and preparation step

The following steps must be carried out for the import of the subsidy application dataset in the system and its preparation for the subsequent processors.

1. Copy the two input datasets described in the last point (subsidy application dataset and the crop code LUT) to the server disk drive, e.g. in /mnt/archive/lpis-import

2. Make sure the dataset is readable by the sen2agri-service user by running:

```
sudo chown -R sen2agri-service: /mnt/archive/lpis-import
```

IMPORTANT NOTE: The parent directories also need to be accessible.

3. Find out the site id:

```
sudo -u sen2agri-service psql sen4cap -c "select id, name from site"
```

4. Run the preparation script:

```
sudo su -l sen2agri-service

/usr/bin/data-preparation.py -s SITE_ID --year 2019 --parcel-id-cols ori_id --holding-id-
cols  ori_hold  --crop-code-col  ori_crop  --lpis  /mnt/archive/lpis-import/*.shp  --lut
/mnt/archive/lpis-import/*.csv --mode update
```

Table 4-5. Subsidy application import and preparation script

Parameter	Description	Type
config-file (-c)	Configuration file location	file
site-id (-s)	Site ID to filter by	int
year	Year (default current year)	int
lpis	Subsidy application dataset	file
lut	LUT dataset	file
export	Export dataset (optional)	file
parcel-id-cols	Name(s) of column(s) containing the unique parcel ID	string
holding-id-cols	Name(s) of column(s) containing the unique holding ID	string
crop-code-col	Name of column containing the crop type codes	string
mode	Execution mode (optional – default "update" – choices : "update", "replace", "incremental"	string
debug (-d)	Enter debug mode	
working-path	Working path (optional)	file
parcel-id-offset	Offset for parcel renumbering (default 0)	int
holding-id-offset	Offset for parcel renumbering (default 0)	Int

4.4.3.3 Output data

The preparation step generates a series of output files, which are used by the subsequent L4x processors. These outputs are described below.

- **Standardized subsidy application dataset with quality flags**

The first output is a postGIS layer. It contains the original subsidy application dataset information as well as the quality flags generated during the preparation step. It has the same number of lines (parcels) than the original subsidy application dataset. It contains the following attribute fields (fields in orange are already present in the original subsidy application dataset) (Table 4-6).

This information is also copied in a csv file, accessible in the folder /mnt/archive/lpis/site-name/.

Table 4-6 Attribute fields in the standardized subsidy application dataset with quality flags

Field name	Role	Default value [format]
Ori attributes	All the original attributes of the original subsidy application dataset	[integer, float or string]
Ori_id	The initial id from the subsidy application dataset	[integer or string]
Ori_hold	The initial holding id from the subsidy application dataset	[integer or string]
Ori_crop	The initial crop code name from the subsidy application dataset	[integer or string]
NewID	New sequential ID of the parcel	[integer]
HoldID	New sequential ID of the holdings	[integer]
CTnum	The new croptype code	[integer]
CT	The crop type name	[string]
LC	Landcover category [optional]	[integer]
GeomValid	Identify parcels for which no polygon exists in the subsidy application dataset or with a not valid geometry	[integer, binary]
Duplic	Identify parcels that have the exact same geometry as another	[integer, binary]
Area_meters	Parcel area in the UTM projection (m²)	[integer]
Overlap	Identify parcels which overlaps with neighbouring parcels	[integer, binary]
ShapeInd	The crop type name	[float]
S1pix	Indicates the number of used S1 pixels in the parcel	[integer]
S2pix	Indicates the number of used S2 pixels in the parcel	[integer]

This output is stored in the database. It is possible to export the layer (for example in shapefile format) from the database by using the following command in the terminal:

```
ogr2ogr -f "ESRI Shapefile" -lco ENCODING=UTF-8 {output_folder}/parcels_dataset.shp
PG:"host=localhost user=admin dbname=sen4cap password=sen2agri port=5432" -sql "SELECT * from
decl_{site_short_name}_{season_name}"
```

- **Parcels buffer layers**

The second outputs are the parcels reprojected buffer layers (5m and 10m inner buffers). These buffers are reprojected in the WGS 84 / UTM zone {x} projections that correspond to all the UTM zones that cross the country, and in ETRS89-LAEA. It only contain one field, the NewID of the parcel.

The parcels buffer layers:

- are shapefiles;
- are projected in WGS 84 / UTM zone {x} (as many as the number of UTM zones that cross the country) and in ETRS89-LAEA;
- do not have necessarily the same number of lines (parcels) than the original subsidy application dataset;
- are generated in the folder /mnt/archive/lpis/site-name/.

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
	2.0	72	07/01/2021	common agricultural policy

- **Parcels raster layers**

The third outputs are the rasters that are produced for both S2 and S1 data by tile, with the NewID as value. Only the pixels that have their centroid located in the corresponding buffer layers (5m for S2 and 10m for S1) have been assigned the NewID value of the parcel.

These layers:

- are .tif files;
- are produced by S2 tile;
- are projected in the WGS 84 / UTM zone {x} corresponding to the UTM zone of the S2 tile;
- value = NewID of the parcels;
- are generated in the folder /mnt/archive/lpis/site-name/.

4.4.4 Creating a Biophysical Indicators status (L3A) product

As outlined in section 2.2.1, the NDVI, LAI, FAPAR and FCOVER indicators are generated by the L3A processor. A more detailed description of the algorithms can be found in the corresponding ATBD document. These products are formatted according to the Product Specification Document (PSD).

4.4.4.1 Processors parameters

When a new L2A product is available or when a user requests a Biophysical Indicators product, the L3A Biophysical Indicators processor can be invoked with the parameters given in Table 4-7.

Table 4-7. L3A Biophysical Indicators processor parameters

Parameter	Description	Type
applocation	Path where the sen2agri is built	string
input	List of products descriptors (xml files)	file list
res	Resolution of the output image: - if the value is 0, the resolution of the input image will be kept; - if the value is 10 or 20, the image will be resampled at 10 or 20.	integer
outdir	Output directory	file
inlaimonodir	Directory where input mono-date LAI products are located (only for reprocessing) (optional)	file
rsrfile	RSR file (/path/filename)	file
rsrcfg	RSR configuration file specific to each mission (/path/filename)	file
tileid	Tile Id (optional)	string
modelsfolder	The forder where the models are located. If not specified, the outdir is considered. (optional)	file
generatemodel	Generate the model. (optional)	string (YES/NO)

Parameter	Description	Type
generatemonodate	Generate the mono-date LAI. (optional)	string (YES/NO)
genreprocessedlai	Generate the reprocessed N-Days LAI. (optional)	string (YES/NO)
genfittedlai	Generate the Fitted LAI. (optional)	string (YES/NO)
siteid	The site ID	string
useintermlaifiles	Specify if intermediate files or the final product files should be used (optional)	
minlai	Minimum value for LAI (optional – default "0.5")	string
maxlai	Maximum value for LAI (optional – default "5.0")	string
modlai	Mode value for LAI (optional – default "0.5")	string
stdlai	Standard deviation value for LAI (optional -default "1")	string
minala	Minimum value for ALA (optional – default "5.0")	string
maxala	Maximum value for ALA (optional - default "80.0")	string
modala	Mode value for ALA (optional – default "40.0")	string
stdala	Standard deviation for ALA (optional – default "20.0")	string
useinra	Generate LAI using INRA version (optional -default NO)	string (YES/NO)
fcover	Generate fCover in the output (optional - default YES)	string (YES/NO)
fapar	Generate fAPAR (optional - default YES)	string (YES/NO)
laiBandsCfg	LAI bands to be used configuration file (optional – default "/usr/share/sen2agri/Lai_Bands_Cfgs.cfg")	file
useSystemBVDistFile	Specifies if the system BV distribution file should be used or it should be generated at each execution. (optional)	string

4.4.4.2 Manual execution from terminal

The L3A processor can be launched manually from the terminal, by defining input L2A products (Sentinel-2 or Landsat 8).

First an output folder must be created:

```
mkdir /mnt/archive/temp/

mkdir /mnt/archive/temp/temp_l3a
```

Second the processing can be launched using a command similar to that here below.

```
lai_retrieve_processing.py --input /mnt/archive/maccs_def/cyl/l2a/*/*.xml --res 10 --outdir
/mnt/archive/temp/test_lai --rsrcfg /usr/share/sen2agri/rsr_cfg.txt --generatemonodate YES
--genreprocessedlai NO --generatemodel YES
```

IMPORTANT NOTE: The "input" parameter must be all the *MTD_ALL.xml files of the L2A products (Sentinel-2 or Landsat 8). The "generatemodel" argument is to be set to YES or an existing model must be choosen to run the LAI processing (i.e. --useinra YES uses the INRA LAI model but requires the related configuration file to be).

4.4.4.3 Manual execution from web interface
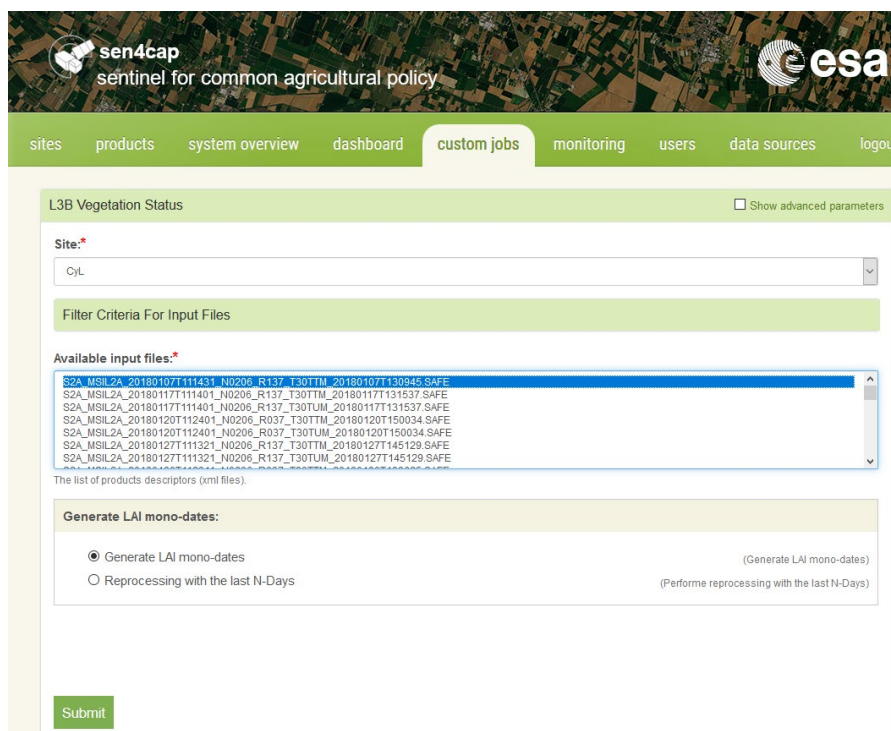
The L3A processor can also be executed manually from the web interface by accessing the "**Custom Jobs**" tab in the web interface and by selecting the "**L3A Biophysical Indicators processor**" tab.

This tab allows the user to obtain a L3A product (i) by using only a subset of the available L2A input products and/or (ii) by running it with other parameters than the default ones defined in the automated processing.

When the editor is open (Figure 4-30), the user has to select first the site and then, he has to define the following parameters:

- Set of L2A input products to be used: when the user selects his site of interest, the "Available input files" list is populated with the existing S2 and L8 L2A products available in the system; the user can select a subset of products and enable / disable one of the two sensors (S2 and L8);

- Spatial resolution ("res" in Table 4-7);

- Type of product that will be generated:
  - Mono-date product;
  - Reprocessing product with the last N-Days.

Figure 4-30. Manual execution of L3A processor from web interface

In the case of the LAI production with the last N-Days, the backward window can be defined by clicking on the the "**Show advanced parameters**" box of the L3A processor (Figure 4-31).

Figure 4-31. Manual execution of L3A processor from web interface (advanced parameters)

4.4.5 Creating a Markers Database (MDB) product

As explained in sections 2.2.2 and 4.2.9, four types of markers are processed by parcel, along the season. Each of these types correspond to a specific database name: MDB1, MDB2, MDB3 and MDB_L4B. Moreover, the MDB_L4B database is divided itself in four databases. The reference name (inside the system) of these different databases are given in section 4.2.9 and the list of markers in section 2.2.2.

IMPORTANT NOTE: in version 2.0, **only the markers from the Marker Database MDB1, MDB2 and MDB3 are processed by the MDB processor**. The markers from the Marker Database MDB_L4A are processed by the L4A crop type processor. Still, these MDB4 markers are available in the same way as the others, but they will only be available after having launched a L4A crop type job.

4.4.5.1 Processors parameters

The Markers Database processor is built in such a way that it is intended for automatic executions. It receives a small number of parameters especially for filtering the input products. These parameters are given in Table 4-8 and can be configured in the "sen4cap" database, in the "config" table.

Table 4-8. Markers Database processor's parameters

Parameter	Description	Type
processor.s4c_mdb1.input_amp	A filtered list of S1 Amplitude products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_mdb1.input_cohe	A filtered list of S1 Coherence products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_mdb1.input_ndvi	A filtered list of NDVI products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_mdb1.ndvi_enabled	Disable NDVI extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool
processor.s4c_mdb1.amp_enabled	Disable AMP markers extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool
processor.s4c_mdb1.cohe_enabled	Disable COHE markers extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool
processor.s4c_mdb1.lai_enabled	Disable LAI markers extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool
processor.s4c_mdb1.fapar_enabled	Disable FAPAR markers extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool

Parameter	Description	Type
processor.s4c_mdb1.fcover_enabled	Disable FCOVER markers extraction. Nevertheless, disabling this might lead to failures of other processors using MDB1. Default : 1	Bool

4.4.5.2   Manual execution from terminal

The Markers Database processor cannot be executed in this moment from the terminal in a generic way.

4.4.5.3   Manual execution from Web Interface

Although the Markers Database processor can be executed manually from the web interface by accessing the "**Custom Jobs**" tab in the Web Interface and by selecting the "**S4C Marker Database PR1**" tab, it is not recommended unless all the input products are selected. The reason is that the latest product is used by the HTTP REST API as the source of the data for the external requests. On the other hand, executing a custom job does not impact other high-level processors that use these data.

This editor allows the user running the Markers Database processor using only a subset of the available L3A and S1 Amplitude and Coherence input products by applying filters on the tiles and start and end dates, as in Figure 4-32.

The input products from the period need to be selected in order to submit the job.

Figure 4-32. Manual execution of Markers Database MDB1 from Website

4.4.6 Creating a crop type map (L4A) product

An overview of the processor is given in the section 2.2.2 and a more detailed description of the algorithms can be found in the corresponding ATBD document. This product is formatted according to the Product Specification Document (PSD). The processor also processes the parcels markers which are used in the crop type classification, and which are also described in the ATBD document. The way to access them is explained in Appendix F.5 Marker Database.

4.4.6.1 Processor parameters

The crop type map (L4A) processor is built in such a way that it can be run automatically. It receives a small number of parameters, which deal mainly with the selection of the monitorable parcels and the definition of the calibration and validation datasets.

When a user requests a crop type product (directly via the Custom Jobs tab or scheduled via the Dashboard tab), the crop type map (L4A) processor is invoked with the parameters given in Table 4-9. The full details concerning these parameters are given in the corresponding ATBD document.

Table 4-9. Crop type map (L4A) processor parameters

Parameter	Description	Type
-s	Site id	integer
--season-start	Start of the season	date
--season-end	End of the season	date
--out-path	Output product path (default ".")	string
--working-path	Working directory path (default ".")	string
--tiles	S2 tiles to be considered (default None)	list
--lc	LC classes to assess (default "1234")	string
--min-s2-pix	Minimum number of S2 pixels to consider a parcel (default 3)	int
--min-s1-pix	Minimum number of S1 pixels to consider a parcel (default 1)	int
--best-s2-pix	Minimum number of S2 pixels for parcels to be used in the training (default 10)	int
--pa-min	Minimum parcels to assess a crop type (default 30)	int
--pa-train-h	Upper threshold for parcel counts by crop type (default 4000)	int
--pa-train-l	Lower threshold for parcel counts by crop type (default 1333)	int
--sample-ratio-h	Training ratio for common crop types (default 0.25)	float
--sample-ratio-l	Training ratio for uncommon crop types (default 0.75)	float
--smote-target	Target sample count for SMOTE (default 1000)	int

Parameter	Description	Type
--smote-k	Number of nearest neighbours used by SMOTE process (default 5)	int
--num-trees	Number of Random Forest trees (default 300)	int
--min-node-size	Minimum node size of the Random Forest trees (default 10)	int
year	year	int

4.4.6.2 Manual execution from terminal

The crop type map (L4A) processor can be executed from a terminal.

First create an output and a temporary work directory.

```
mkdir /mnt/archive/crop-type-output

mkdir /mnt/archive/tmp
```

Then execute the crop type map.

```
crop-type-wrapper.py -s SITE_ID --season-start "2018-01-01" --season-end "2019-01-01" --
working-path /mnt/archive/tmp --out-path /mnt/archive/crop-type-output/ --tiles 31UWT 32UQR
```

4.4.6.3 Manual execution from web interface

The crop type map (L4A) processor can also be executed manually from the web interface by accessing the "**Custom Jobs**" tab and by selecting the "**L4A Crop Type**" tab.

This tab allows the user to run the crop type map (L4A) processor using only a subset of available L2A input products by applying filters on the tiles and on the start and end dates of the monitoring period, as showed in Figure 4-33.

Figure 4-33. Manual execution of the crop type map (L4A) processor from the web interface

By selecting the button "**Show advanced parameters**", the parameters described in the section 4.4.6.1 can also be modified before to submit the job (Figure 4-34).

Figure 4-34. Manual execution of the crop type map (L4A) processor from the web interface by changing the advanced parameters

4.4.6.4 Running the crop diversification use case

Running the crop diversification assessment adds to the results a conformity assessment at the parcel level, comparing the declared crop type with the predictions of the classification in the case of the classified parcels, as well as the reason why a parcel is not classified in the case of the not classified parcels. It is then using this assessment at the parcel-level to check the compliancy of each holding regarding the crop diversification rules. The method used to make this assessment is detailed in the L4A ATBD document.

To run the script, follow these steps in a terminal:

- Change the rights on the classification product directory to be able to create the outputs:

        sudo chmod -R 777 /mnt/archive/{site}/s4c_l4a/{product}/VECTOR_DATA/

- Go to the classification product directory:

        cd /mnt/archive/{site}/s4c_l4a/{product}/VECTOR_DATA/

- Run the crop diversification assessment, with the command:

        crop-diversification.py -i CropType.csv -l CropType_LUT.csv -o crop_div.csv

Two outputs will be created in the same folder:

- `crop_div.csv`: the results of the conformity assessment and the crop diversification assessment for each parcel;
- `crop_div_holding.csv`: the results of the crop diversification assessment for each holding.

4.4.7 Creating a Grassland Mowing (L4B) product

An overview of the processor is given in the section 0 and a more detailed description of the algorithms can be found in the corresponding ATBD document. This product is formatted according to the Product Specification Document (PSD).

4.4.7.1 Processors parameters

The L4B Grassland Mowing processor is built in such a way that is intended for automatic executions. It receives a small number of parameters especially for filtering the input products.

When a new L2 product is available or a when user requests a grassland mowing product, the L4C Grassland Mowing Processor can be invoked with the parameters given in Table 4-10.

Table 4-10. L4B Processor's parameters

Parameter	Description	Type
config_file	config file containing processing parameters, input data directories, and so on. (See config.ini in src_ini/)	file
input_shape_file	input shape file used to extract parcel geometries and the segmentation raster through the gdal rasterization process	shapefile
output_data_dir	output dir where results are written (it is automatically generated if it does not exist)	Directory
new_acq_date	date of the last acqsuistion (yyyymmdd)	Date

Parameter	Description	Type
older_acq_date	date of the older acqsuistion (yyyymmdd) (for run-time processing, it is equal to new_acq_date, else, for reanalysis it is a data previous new_acq_date)	date
tile_list_file	file with the list of the orbits to be considered. (See S1_orbit_list.ini in src_ini/)	List of tiles
seg_parcel_id_attribute	attribute name (string) used in input_shape_file for the parcel identifiers	String
outputShapeFile	output shape file generated/updated to report the mowing detections and the compliances	Shapefile
do_cmpl	True or False. "True" to perform finally the compliance analysis, "False" to skip the compliance	boolean

4.4.7.2 Manual execution from terminal

To generate the L4B Grassland Mowing input shapefile, the following commands can be executed:

First create a working directory and give writing permission to the folder to all users.

```
mkdir /mnt/archive/tmp/grassland_moving_wd/

chmod 777 -R /mnt/archive/tmp/grassland_mowing_wd/
```

The following command creates the shapefile.

```
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/generate_grassland_mowing_input_shp.sh -s 1
-y 2019 --path /mnt/archive/tmp/grassland_mowing_wd/
```

The L4B Grassland Mowing Processor can be executed from a terminal, for the S1 products, in the following way:

```
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/grassland_mowing.sh          --script-path
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_s1/S1_main.py --site-id 2 --config-file
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_ini/S4C_L4B_Default_Config.cfg  --input-
shape-file        /mnt/archive/orchestrator_temp/s4c_l4b/1152/43883-s4c-grassland-gen-input-
shp/SEN4CAP_L4B_GeneratedInputShp.shp                                  --output-data-dir
/mnt/archive/orchestrator_temp/s4c_l4b/1152/43885-s4c-grassland-
mowing/SEN4CAP_L4B_S1_OutputData --start-date 2019-04-15 --end-date 2019-05-30 --seg-parcel-
id-attribute   NewID   --output-shapefile   /mnt/archive/orchestrator_temp/s4c_l4b/1152/43886-
product-formatter/SEN4CAP_L4B_S1_S2_MowingDetection.shp --do-cmpl True --test True --season-
start 2019-01-01 --season-end 2019-12-31
```

The L4B Grassland Mowing Processor can be executed from a terminal, for the S2 products, in the following way:

```
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/grassland_mowing.sh          --script-path
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_s2/S2_main.py --site-id 6 --config-file
```

```
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_ini/S4C_L4B_Default_Config.cfg  --input-
shape-file
/mnt/archive/grassland_mowing_files/nld_2019/2019/InputShp/SEN4CAP_L4B_GeneratedInputShp.sh
p    --output-data-dir    /mnt/archive/    grassland_mowing_files/nld_2019/2019/
Output/SEN4CAP_L4B_S2_OutputData --start-date 2019-06-01 --end-date 2019-10-31 --seg-parcel-
id-attribute         NewID              --output-shapefile              /mnt/archive/
grassland_mowing_files/nld_2019/2019/Output/SEN4CAP_L4B_S1_S2_MowingDetection.shp  --do-cmpl
True --test True --season-start 2019-01-01 --season-end 2019-12-31
```

For the last two commands, instead of providing the --season-start and --season-end can be provided also the --input-products-list that explicitly defines the list of S2 or S1 products to be used like for example:

```
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/grassland_mowing.sh         --script-path
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_s2/S2_main.py --site-id 6 --config-file
/usr/share/sen2agri/S4C_L4B_GrasslandMowing/Bin/src_ini/S4C_L4B_Default_Config.cfg  --input-
shape-file
/mnt/archive/grassland_mowing_files/nld_2019/2019/InputShp/SEN4CAP_L4B_GeneratedInputShp.sh
p    --output-data-dir    /mnt/archive/    grassland_mowing_files/nld_2019/2019/
Output/SEN4CAP_L4B_S2_OutputData --start-date 2019-06-01 --end-date 2019-10-31 --seg-parcel-
id-attribute         NewID              --output-shapefile              /mnt/archive/
grassland_mowing_files/nld_2019/2019/Output/SEN4CAP_L4B_S1_S2_MowingDetection.shp  --do-cmpl
True                --test                True                --input-products-list
/CZE_2019_L2B/SEN4CAP_L2A_S2_V20181227T050828_20190102T050903_VH_022/SEN4CAP_L2A_S2_V201812
27T050828_20190102T050903_VH_022_AMP.tif
/CZE_2019_L2B/SEN4CAP_L2A_S2_V20181227T050828_20190102T050903_VV_022/SEN4CAP_L2A_S2_V201812
27T050828_20190102T050903_VV_022_AMP.tif ...
```

4.4.7.3 Manual execution from Web Interface

The L4B Grassland Mowing Processor can be also executed manually from the web interface by accessing the "**Custom Jobs**" tab in the Web Interface and by selecting the "**L4B processor**" tab.

As most of the parameters are configured in dedicated configuration files found on the server, this editor allows the user running the L4B processor using only a subset of the available L3A and S1 Amplitude and Coherence input products by applying filters on the tiles and start and end dates, as in Figure 4-35.
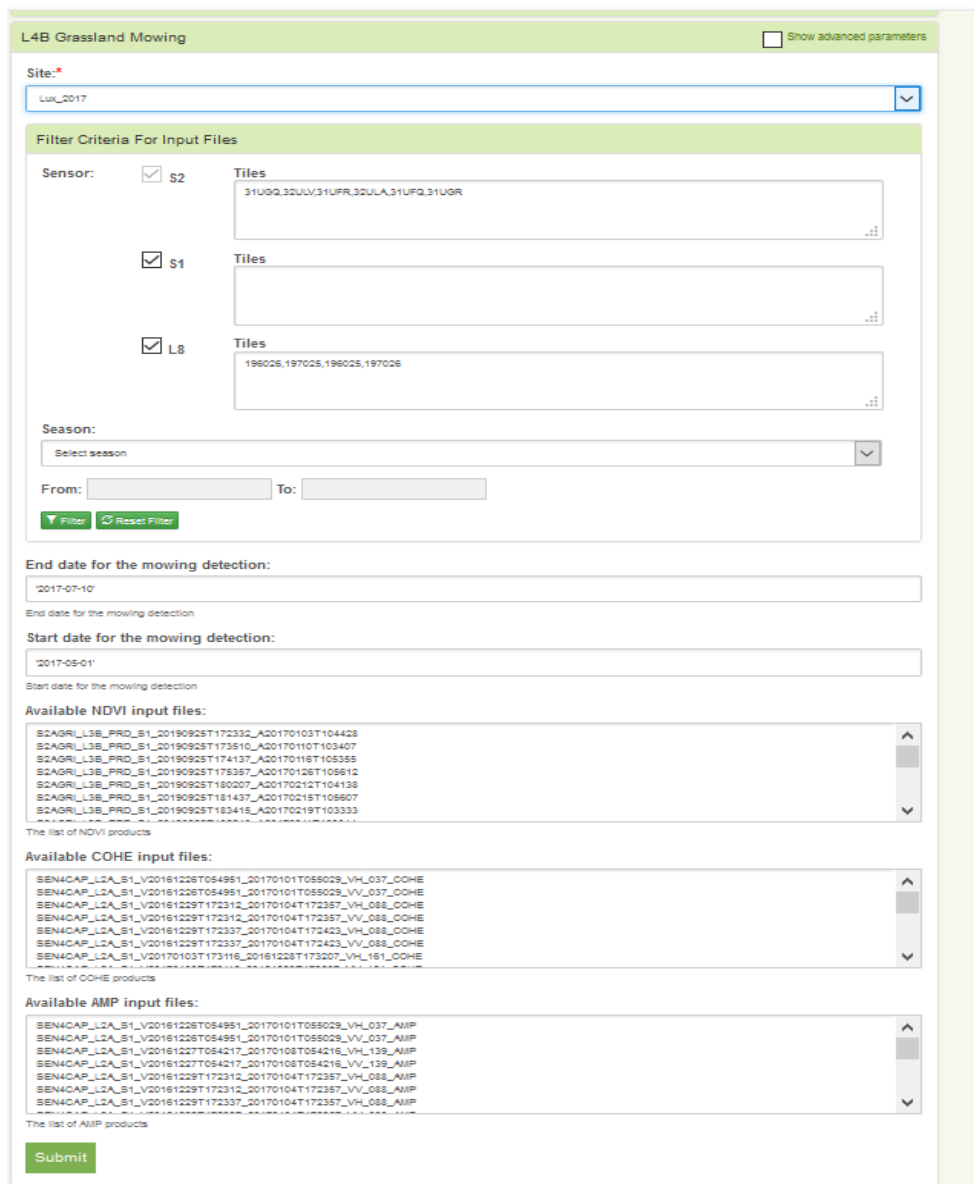
Figure 4-35. Manual execution of Grassland Mowing processor from Web Site

4.4.8 Creating an Agricultural Practices Monitoring (L4C) product

An overview of the processor is given in the section 2.2.5 and a more detailed description of the algorithms can be found in the corresponding ATBD document. This product is formatted according to the Product Specification Document (PSD).

4.4.8.1 Processors parameters

Normally, the L4C Agricultural Practices processor is built in such a way that is intended for automatic executions. It receives a small number of parameters especially for filtering the input products.

The Agricultural Practices Processor can be invoked with the parameters given in Table 4-11. These parameters are configured in the "sen4cap" database, in the "config" table.

Table 4-11. L4C Processor's parameters

Parameter	Description	Type
processor.s4c_l4c.cfg_dir	The default config directory, where the config files are searched. Default value "/mnt/archive/agric_practices_files/{site}/{year}/config/"	String
processor.s4c_l4c.data_extr_dir	Location on disk for the data extraction directory. Default value "/mnt/archive/agric_practices_files/{site}/{year}/data_extraction/{product_type}"	String
processor.s4c_l4c.ts_input_tables_dir	Location on disk where the input tables are searched. Default value "/mnt/archive/agric_practices_files/{site}/{year}/ts_input_tables/{practice}"	String
processor.s4c_l4c.filter_ids_path	Location on disk where the filter for parcel ids are created. Default value "/mnt/archive/agric_practices_files/{site}/{year}/ts_input_tables/FilterIds/Sen4CAP_L4C_FilterIds.csv"	String
processor.s4c_l4c.input_amp	A filtered list of S1 Amplitude products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_l4c.input_cohe	A filtered list of S1 Coherence products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_l4c.input_ndvi	A filtered list of NDVI products. This is just a placeholder and should not be modified in the database.	list
processor.s4c_l4c.sub_steps	The list of all possible practices. This is just a placeholder and should not be modified in the database.	list
processor.s4c_l4c.nrt_data_extr_enabled	Enable the data extraction triggered on an occurrence of an NDVI, AMP or COHE product. Experimental.	bool
processor.s4c_l4c.execution_operation	Default execution operation. This is just a placeholder and should not be modified in the database. Default: "ALL"	List
processor.s4c_l4c.cfg_upload_dir	Location where the config files are uploaded prior importing them in the final folder. Default value "/mnt/archive/upload/agric_practices_files/{site}/config"	String
processor.s4c_l4c.ts_input_tables_upload_root_dir	Location where the input table files files are uploaded prior importing them in the final folder. Default value "/mnt/data/archive/upload/agric_practices_files/{site}/ts_input_tables"	String
processor.s4c_l4c.prds_per_group	Number of the data extraction products to be extracted at once. Deprecated and should not be changed as it will be removed in future versions. Default value = 1	Int

Parameter	Description	Type
processor.s4c_l4c.default_config_path	A default configuration file that can be used if no config file was uploaded for a site. Default: "/usr/share/sen2agri/S4C_L4C_Configurations/S4C_L4C_Default_Config.cfg"	String
processor.s4c_l4c.practices	Default practices for a site. This is just a placeholder and should not be modified in the database as it will be created a new instance for each site. Default: "NA"	String
processor.s4c_l4c.country	Default country for a site. This is just a placeholder and should not be modified in the database as it will be created a new instance for each site. Default: "CNTRY"	String
processor.s4c_l4c.tsa_min_acqs_no	Minimum number of valid acquisitions in order to start the time series analysis for a parcel. Default: 15	int
processor.s4c_l4c.use_prev_prd	Specifies if the previous product should be used in the processing of the current one. If set, parcels detected as harvested in the previous product will not be re-evaluated. Default value : 1	Bool (0 or 1)
processor.s4c_l4c.sched_prds_hist_file	List of the products produced using scheduled jobs. Only these products (not the ones obtained from custom jobs) will be used in generating the M1-M5 markers database (MDB2). Default : "/mnt/data/archive/agric_practices_files/{site}/{year}/l4c_scheduled_prds_history.txt"	String
processor.s4c_l4c.year	Default country for a site year. This is just a placeholder and should not be modified in the database as it will be created a new instance for each site. Default: ""	String
processor.s4c_l4c.markers_add_no_data_rows	Specifies if in the resulted M1-M5 markers (MDB2) should be inserted also parcels having only NA. Default: 1	int

4.4.8.2 Inputs

This processor needs that the subsidy application dataset, the Agricultural Practices configuration file and the Agricultural Practices input tables are correctly imported (see sections 4.2.3.2, 4.2.3.5 and 4.2.3.6). Descriptions of these two last files are provided in the Appendices F.13 Agricultural Practices configuration file and F.14 Agricultural input tables file.

In the practices info file, the attributes to include are the following (view example in the Appendix F.13):

- "FIELD_ID" field ID (unique value corresponding to the original id from the subsidy application dataset, not containing a "/" character);
- "MAIN_CROP" "main crop" attribute or "NA";
- "VEG_START" start of a period of "main crop" growing (YYYY-MM-DD);
- "H_START" earliest start of "main crop" harvest period (YYYY-MM-DD);
- "H_END" latest end of "main crop" harvest period (YYYY-MM-DD);
- "PRACTICE" EFA practice (list of available EFA practices, e.g. "CatchCrop"; or "NA");
- "P_TYPE" EFA practice sub-type (list for each country/practice; or "NA");

- "P_START" start of EFA period (YYYY-MM-DD; or "NA");
- "P_END" end of EFA period (YYYY-MM-DD; or "NA").

4.4.8.3 Manual execution from terminal

The L4C Agricultural Practices Processor cannot be executed in this moment from the terminal in a generic way.

4.4.8.4 Manual execution from Web Interface

The L4C Agricultural Practices Processor can be executed manually from the web interface by accessing the "**Custom Jobs**" tab in the Web Interface and by selecting the "**L4C processor**" tab.

As most of the parameters are configured in dedicated configuration files found on the server, this editor allows the user running the L4C processor using only a subset of the available L3A and S1 Amplitude and Coherence input products by applying filters on the tiles and start and end dates, as in Figure 4-36.

The input products from the period need to be selected in order to submit the job.

IMPORTANT NOTE: The agricultural practices does not have in this moment advanced parameters. The advanced parameters can be configured using a configuration file that can be uploaded in the "Sites" section, as described in Section 4.2.3.

Figure 4-36. Manual execution of Agricultural Practices processor from Website, including the possibility to enable or disable the tillage detection

4.4.8.5   Exporting the output as a shapefile

The output of this processor, containing the parcels that were monitored and describing the assessed markers, week of harvest, and degree of compliancy to regulations, is currently produced in the GeoPackage (GPKG) format (that is recognized by almost all well-known applications). If the desired format is an ESRI shapefile, the GPKG file can be easily converted to an ESRI shapefile using a command similar with the following one, executed in the product folder:

```
ogr2ogr CatchCrop.shp CatchCrop.gpkg -lco ENCODING=UTF-8
```

4.4.8.6   L4C M1-M5 Markers Database (MDB2)

Please note that when an automatic generation of an L4C product (using a scheduled job), automatically an MDB3 product containing the M1-M5 markers contained in the corresponding L4C product will be created. These markers can be accessed either downloading the physical MDB3 product from the web interface or using the HTTP REST API, as described in section Appendix F.5 Marker Database.

4.4.9 Creating an Agricultural Practices Monitoring (L4C) product with tillage detection

The tillage detection processor is implemented as an additional option in the Agricultural Practices Monitoring processor (L4C).

Thus, the inputs for the tillage detection are the same as the ones described in section 4.4.8 Creating an Agricultural Practices Monitoring (L4C) product.

Nevertheless, additional parameters were added to the L4C processor that allow activating or deactivating the tillage detection and also to parametrize this option (Table 4-12).

Table 4-12. Specific parameter for tillage in the L4C Processor

Parameter	Description	Type
processor.s4c_l4c.tillage_monitoring	Activates or deactivates the tillage monitoring in the L4C processor. Default : 0	

4.4.9.1   Manual execution from terminal

As the Tillage Detection is an option of L4C Agricultural Practices Processor, it cannot be executed in this moment from the terminal in a generic way.

4.4.9.2   Manual execution from Web Interface

The L4C Agricultural Practices Processor can be executed, with Tillage Detection, manually from the web interface by accessing the "**Custom Jobs**" tab in the Web Interface and by selecting the "**L4C processor**" tab but by setting the "Enable or disable tillage monitoring" flag to value "1" instead of 0.

This way, the resulted L4C product will contain embedded also the columns corresponding to the tillage detection (see previous Figure 4-36).

5. Maintenance of the system

5.1 Uninstall procedure

The following steps should be run as *root* user in order to uninstall the Sen4CAP system from the computer.

- Stop and disable the running applications:

```
# systemctl stop sen2agri-orchestrator sen2agri-http-listener sen2agri-
services sen2agri-monitor-agent

# systemctl disable sen2agri-orchestrator sen2agri-http-listener sen2agri-
monitor-agent
```

- Uninstall the application packages:

```
# yum remove sen2agri-processors sen2agri-website sen2agri-downloaders-
demmaccs sen2agri-app

# rm -fR /usr/share/sen2agri/sen2agri-services
```

- Stop and disable SLURM and MUNGE:

```
# systemctl stop Slurmd Slurmctld Slurmdbd munge

# systemctl disable Slurmd Slurmctld Slurmdbd munge
```

- Uninstall SLURM and MUNGE:

```
# yum remove Slurm Slurm-munge Slurm-Slurmdb-direct Slurm-sql Slurm-torque
Slurm-sjstat Slurm-plugins Slurm-Slurmdbd Slurm-devel Slurm-sjobexit Slurm-
perlapi Slurm-pam_Slurm munge-devel munge-libs munge
```

- Stop and disable MariaDB:

```
# systemctl stop mariadb

# systemctl disable mariadb
```

- Uninstall MariaDB:

```
# yum remove mariadb-server mariadb-devel mariadb
```

- Remove the PostgreSQL database:

```
# su -l postgres -c 'psql -c "drop database sen4cap;"'
```

- Stop and disable PostgreSQL:

```
# systemctl stop postgresql-9.4

# systemctl disable postgresql-9.4
```

- Uninstall PostgreSQL:

```
# yum remove postgresql94-server postgresql94-contrib postgresql94
```

- Uninstall OTB and GDAL 2.0:

```
# yum remove otb gdal-local
```

- Stop and disable Apache:

```
# systemctl stop http

# systemctl disable http
```

- Uninstall Apache and PHP:

```
# yum remove httpd php
```

- Uninstall dependencies:

```
# yum remove boost tinyxml tinyxml-devel qt qt5-qtbase qt-x11 fftw gdal geos
libgeotiff libsvm muParser opencv openjpeg2 openjpeg2-tools proj proj-epsg
swig qt5-qtbase-postgresql gsl cifs-utils
```

- Uninstall MAJA:

Follow the guidelines available in the MAJA user manual [RD.2].

IMPORTANT NOTE: Other applications on the system might depend on some of the packages above. Before continuing, please make sure to double-check the list of packages to be removed.

5.2 Update procedure

Upgrading versions of the Sen4CAP system is possible.

All upgrades can be performed using the same script Sen4CAPDistribution/install_script/update.sh:

```
# cd /path/to/Sen4CAPDistribution/install_script/

# sudo ./update.sh
```

The MAJA installation is not performed automatically by the system and should be performed manually by the user according to the instructions described in the MAJA documentation.

Nevertheless, after a reinstall of MAJA, the following operations should be performed in the Sen4CAP system in order to use the new MAJA version:

```
$ rm -rf /mnt/archive/gipp_maja/*

Or

$ rm -rf /mnt/archive/gipp/maja/*

$ cp <SEN4CAP_ROOT>/Sen4CAPDistribution/gipp_maja/* /mnt/archive/gipp/maja

$ psql -U admin sen4cap

sen4cap=# update config set value = '/opt/maja/3.2.2/bin/maja' where key
='processor.l2a.maja.launcher';

sen4cap=# update config set value = '/mnt/archive/gipp/maja' where key
='processor.l2a.maja.gipp-path';

sen4cap=# \q
```

To change the MAJA version (for example, switching from MAJA 3 to MAJA 4) the following operations should be performed:

- Install the new MAJA version (MAJA 4, for example)
- Copy the MAJA 4 GIPP files into `/mnt/archive/gipp/maja4`
- Update the `'processor.l2a.maja.launcher'` and the `'processor.l2a.maja.gipp-path'` keys accordingly using the above commands.

IMPORTANT NOTE: Please note that the Sen4CAP system was validated using L2A products created with MAJA 3. Using MAJA 4 might lead to different results.

IMPORTANT NOTE: It is not advisable to switch from MAJA 3 to MAJA 4 while a season production is still in progress as unpredictable behavior might occur. The switching from MAJA 3 to MAJA 4 should be done only if all the sites were removed from the system or disabled.

6. The Sen4CAP Visualisation Tool and Sen4CAP WMS/ Statistical Info Services

6.1 Deployment of the Visualization Tool

The installation package of the Visualization tool is contained in a single zip file named **"GeopediaWorld.zip".** Installation is done in twp steps:

1) Unzip all the files from "GeopediaWorld.zip" into one directory of any http server, which can serve static files. This is for instance the case for Apache, Tomcat, IIS...

2) In any browser, go to the URL, representing the domain and the selected directory.

For example, if you unzipped the files into "mysites/geopedia", which is relative to the root directory of the web server, and the server is hosted at "http://www.myserver.org", then the URL is: http://www.myserver.org/mysites/geopedia.

After deployment, the Visualization Tool will connect to back-end services running on Geopedia's private server cluster.

6.2 How to use the Visualization Tool

6.2.1 How to log into the Visualization Tool

To log into the Visualization Tool, click on "Login" tab on the top right corner of the page. This will open a pop-up window where you can either log in with existing username and password or register (Figure 6-1).
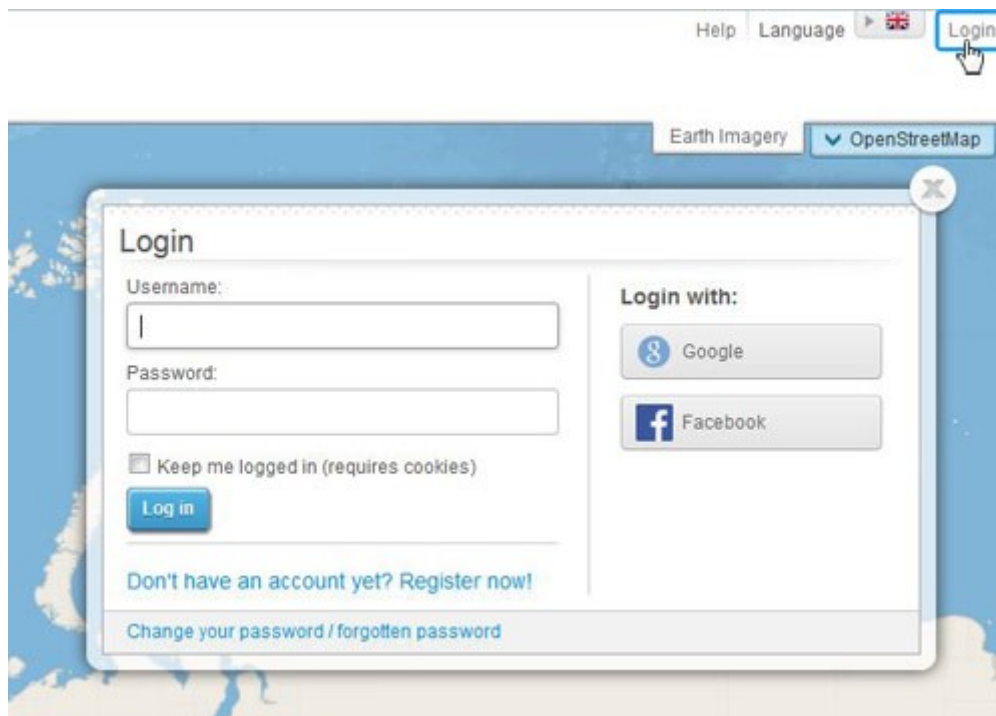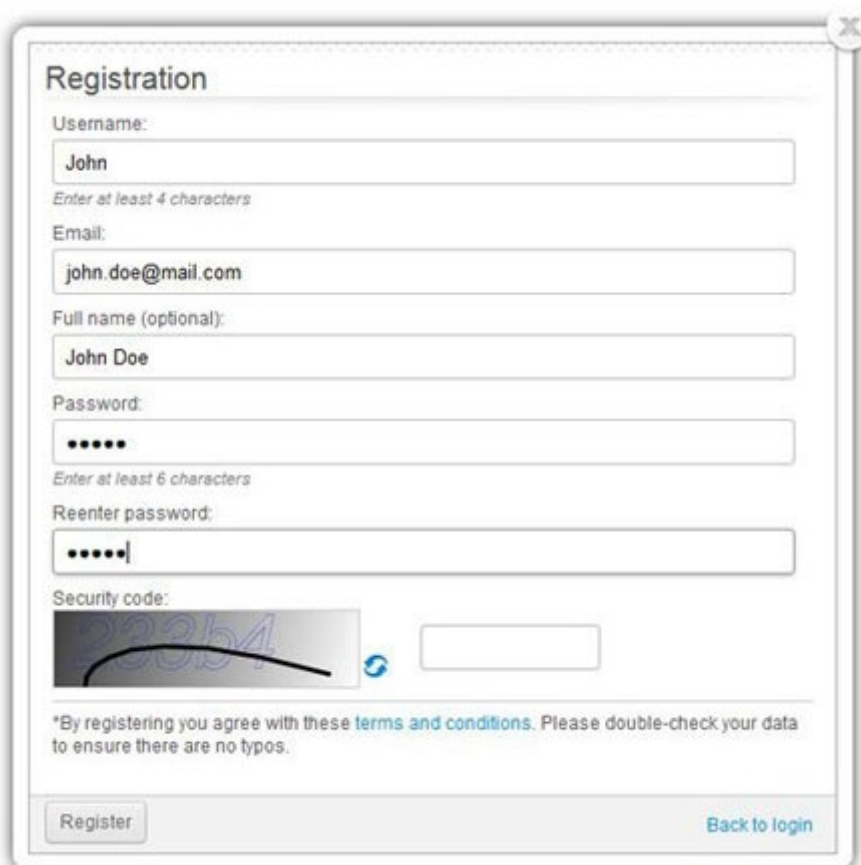
Figure 6-1. Dialog box to log in the Sen4CAP visualization tool

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
	2.0	96	07/01/2021	common agricultural policy

In the case you need to register, you have to fill in all mandatory fields marked with * (Figure 6-2). Activation mail will be sent to your e-mail address. By clicking on the link provided in this activation e-mail, you will activate your account.

Figure 6-2. Dialog box to register as a new user to the Sen4CAP visualization tool

6.2.2 How to create a layer or theme

To create a layer or a theme, you first need to log in. After log in, a PERSONAL tab will appear at the top left of your screen. By selecting this tab, you can access 2 buttons: one for creating a new layer and one for creating a new theme (Figure 6-3).

Figure 6-3. Personal tab of the Sen4CAP visualization tool, allowing creating new layer or theme

6.2.3 How to create a "Sentinel" layer

Click on the "**NEW LAYER**" button of your PERSONAL tab, and select the Sentinel layer type (Figure 6-4).

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
	2.0	97	07/01/2021	common agricultural policy

Figure 6-4. Adding a Sentinel layer in the Sen4CAP visualization tool

After clicking on the Sentinel layer button, click on the "**General**" tab and fill in the different fields as illustrated in Figure 6-5.
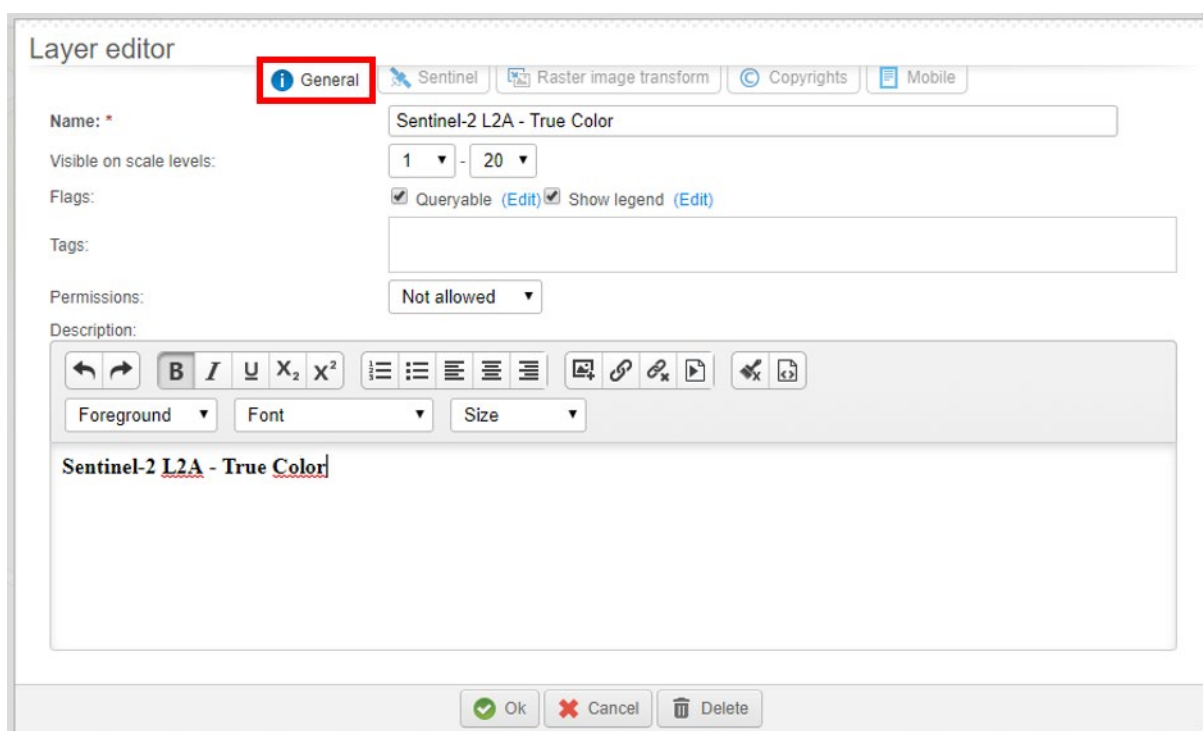
Figure 6-5. General information of the Sentinel layer to be added in the Sen4CAP visualization tool

In the "**Sentinel**" tab, enter the following WMS URL:

http://services.eocloud.sentinel-hub.com/v1/wms/c8d740dd-f9c0-4f6c-9a05-f2dabcfac982?

and click on the "**GO**" button.

Some parameters need to be set depending on the EO source you wish to add:
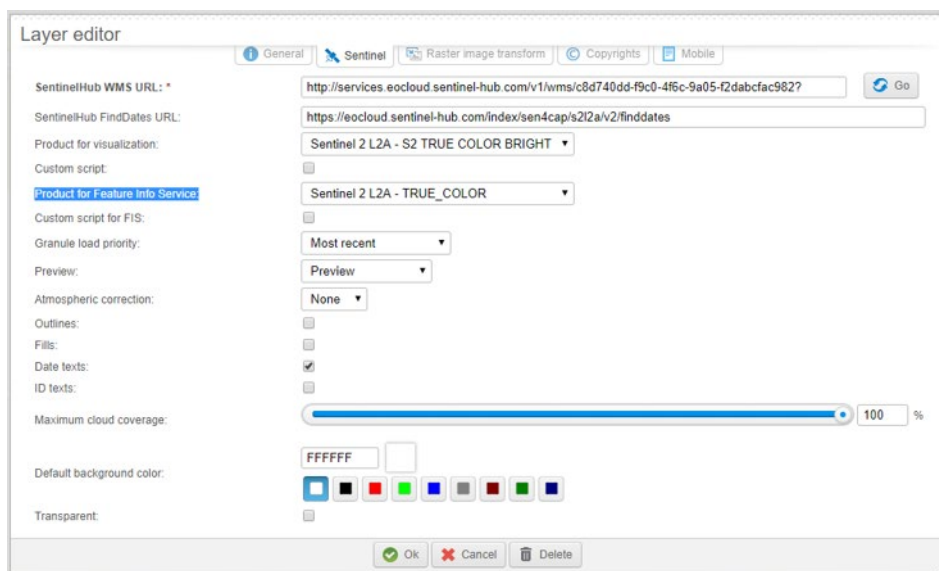
- **Sentinel-2 L2A - True Color** (Figure 6-6)

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/s2l2a/v2/finddates

Product for visualization: Sentinel 2 L2A – S2 TRUE COLOR BRIGHT

Product for Feature Info Service: Sentinel 2 L2A – S2 TRUE COLOR
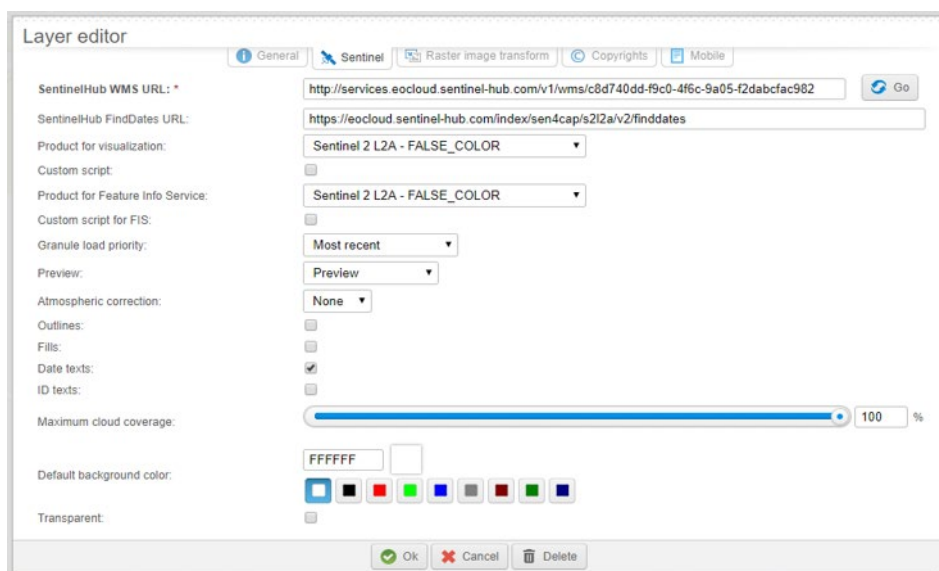
Figure 6-6. Settings for the Sentinel-2 L2A True Color layer

- **Sentinel-2 L2A - False Color** (Figure 6-7)

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/s2l2a/v2/finddates

Product for visualization: Sentinel 2 L2A – S2 FALSE_COLOR

Product for Feature Info Service: Sentinel 2 L2A – S2 FALSE_COLOR

Figure 6-7. Settings for the Sentinel-2 L2A False Color layer

- **Sentinel-2 L3A – NDVI** (Figure 6-8)

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/s2l3b/v2/finddates

Product for visualization: Sentinel 2 L3B – NDVI color

Product for Feature Info Service: Sentinel 2 L3B – NDVI
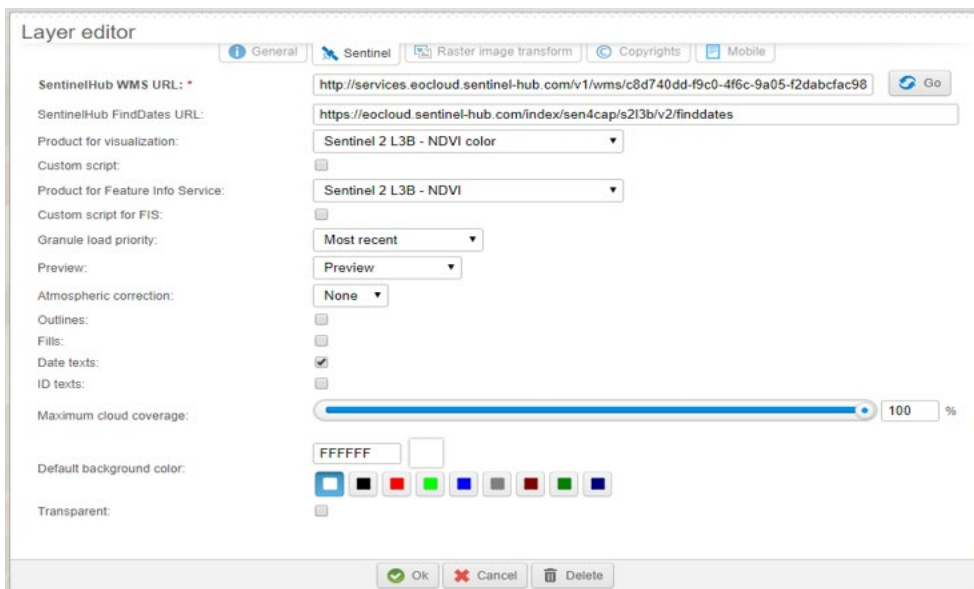
Figure 6-8. Settings for the Sentinel 2 L3A – NDVI layer

- **Sentinel-2 L3A – LAI** (Figure 6-9)

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/s2l3b/v2/finddates

Product for visualization: Sentinel 2 L3B – LAI color

Product for Feature Info Service: Sentinel 2 L3B – LAI
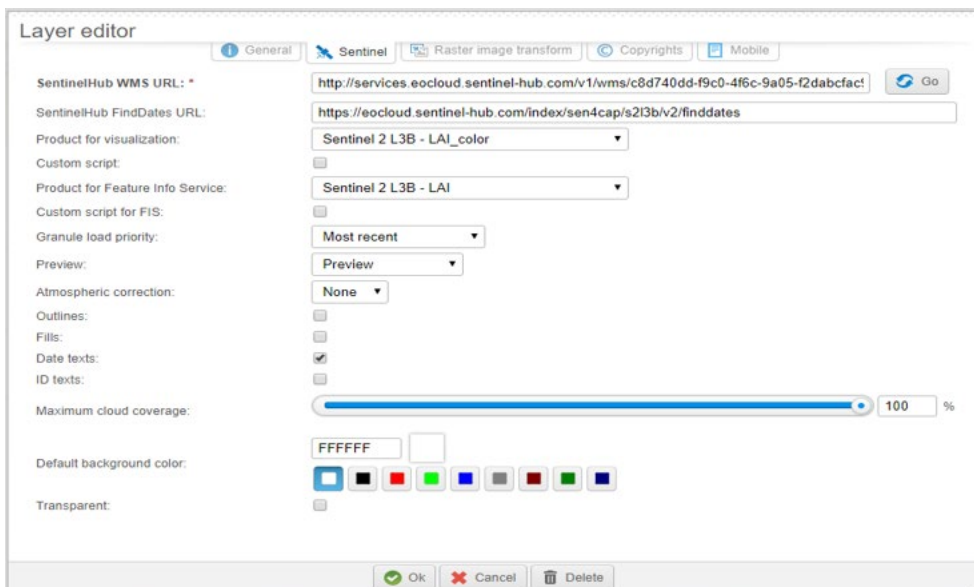
Figure 6-9. Settings for the Sentinel-2 L3A - LAI layer

- **Sentinel-2 L3A – FAPAR** (Figure 6-10)

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/s2l3b/v2/finddates

Product for visualization: Sentinel 2 L3B – FAPAR_ color

Product for Feature Info Service: Sentinel 2 L3B – FAPAR
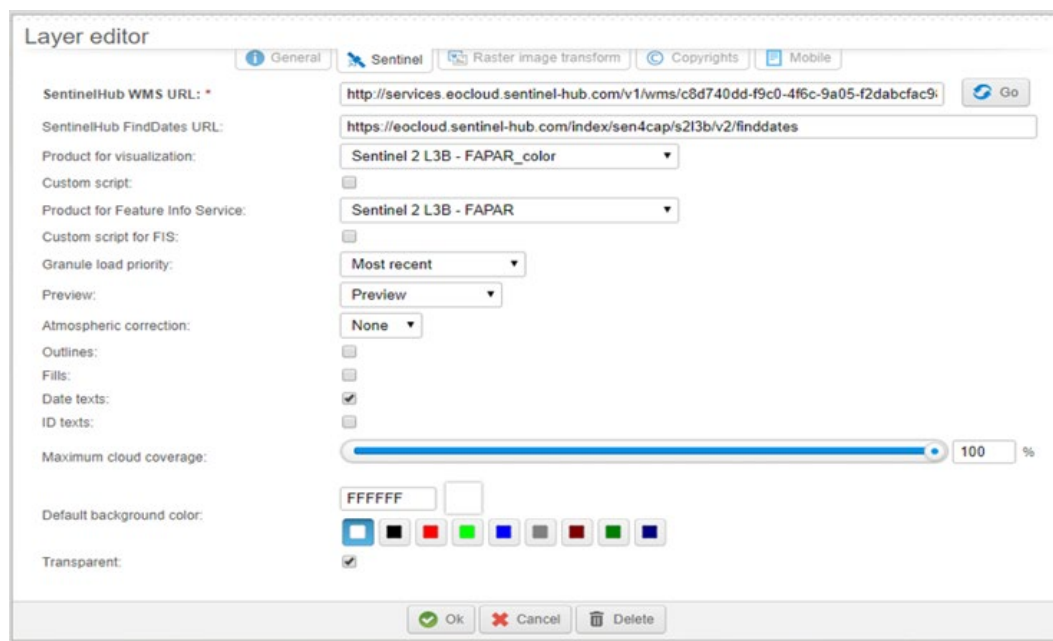
Figure 6-10. Settings for the Sentinel-2 L3A - FAPAR layer

- **Landsat 8 L2A - True Color**

SentinelHub FindDates URL: https://eocloud.sentinel-hub.com/index/sen4cap/l8l2a/v2/finddates

Product for visualization: Landsat 8 L2A – TRUE_COLOR

Product for Feature Info Service: Landsat 8 L2A – TRUE_COLOR
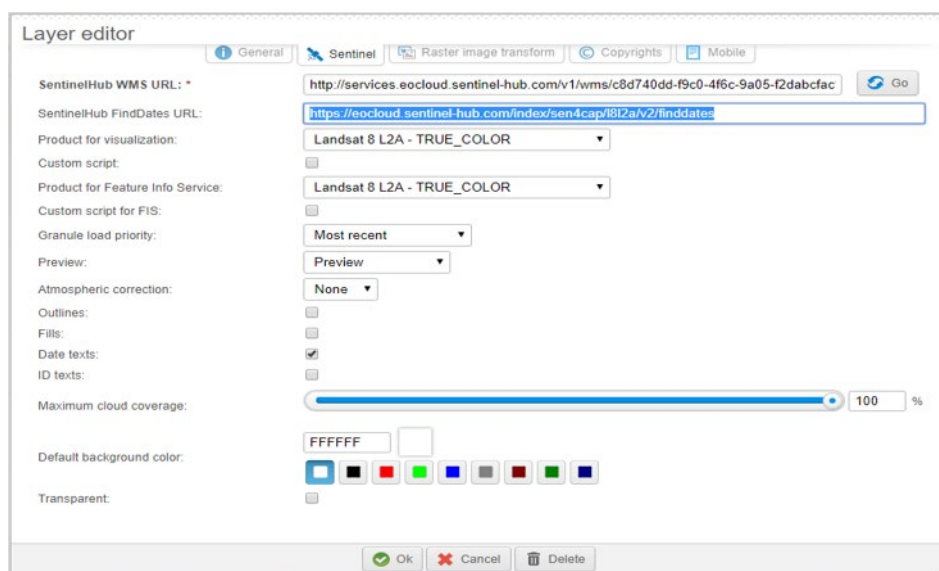
Figure 6-11: Settings for the Landsat 8 L2A True Color layer

6.3 Deployment of Sen4CAP WMS/Statistical Info Services

6.3.1 Installation package

The installation package is contained in the "sen4cap-master.zip" archive. It includes the following content:

- "**SentinelCommon**" folder;
- "**SentinelRenderer**" folder;
- "**SentinelWmsService**" folder;
- ExternalLibs.txt;
- SinergiseLibs.zip;
- COPYING;
- LICENSE;
- README.

6.3.2 Requirements for running

The requirements to run the Sen4CAP WMS / Statistical Info Services are:

- Eclipse 4.6 or higher;
- Java SDK 1.8;
- Libraries listed in file ExternalLibs.txt;
- Libraries provided in installation package (SinergiseLibs.zip);
- Source code (folders in sen4cap-master.zip);
- Jetty distribution package version 9.3.19 (https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-distribution/9.3.19.v20170502/jetty-distribution-9.3.19.v20170502.zip).

6.3.3 Installation procedure

6.3.3.1 Preparation

A list of external open source libraries is provided in the "**ExternalLibs.txt**" file included in the installation package. The information contained in this file has to be understood as follows, with for instance the library "*CloudServices/amazon/aws-java-sdk/1.10.49/aws-java-sdk-core-1.10.49.jar*":

1) *CloudServices* → Library group;
2) *amazon* → library owner (optional);

	Ref	Sen4CAP_SUM_v2.0		
.esa	Issue	Page	Date	**sen4cap**
	2.0	102	07/01/2021	common agricultural policy

3) *aws-java-sdk* → library name;

4) *1.10.49* → library version;

5) *aws-java-sdk-core-1.10.49.jar* → library file with (optional version).

Execute the following steps:

- Store all the libraries in one directory so that you are able to add them to the eclipse project.

- Extract source code to directory on disk.

- Store jetty distribution package on disk.

6.3.3.2 Eclipse setup

Execute the following steps:

- Start a new workspace;

- Go to Window → Preferences → Java → Build Path → User Libraries;

- Click "**New**" to create new libraries and set name for them;

- Click "**Add External JARs**";

- Select all jars (provided in the "SinergiseLibs" archive and in the External Libraries) and Open, then Apply and Close;

- Click on "**Import**" → Projects from Folder or Archive

- Select directory where sources are extracted

- Select all three projects (SentinelCommon, SentinelRenderer, SentinelWmsService)

- Under each of the project, add User Library
    - Right click → Build Path → Configure Build Path;
    - Add Library;
    - Select User Library and select previous created user libraries;

- Apply and Close.

After these steps, the project should rebuild itself and be ready to run.

6.3.3.3 Starting WMS Service

Execute the following steps:

- Open SentinelWmsService in eclipse;

- Right click on SentinelWmsService.launch;

- Run as → Run Configurations (required only the first time);

- Go to Jetty Webapp->SentinelWmsService;

- Under Options select jetty that was previously extracted and click apply;
- Click Run.

After few seconds, the application will announce jetty startup finished and you can access application on localhost port 8086.

Appendix A - Additional tools available in the system

Appendix A.1 Sen4CAP Executor parameters

The Sen4CAP Executor is based on the Sen2Agri Executor and is in charge of the execution of the processors applications, using SLURM for executing these applications.

Parameter Name	Description	Configuration Method
Executor IP Address	The address where the executor is found	Sen4CAP Configurator
Executor Port	The port where the executor is found	Sen4CAP Configurator

Appendix A.2 Sen4CAP Persistence Manager parameters

The Sen4CAP Persistence Manager offers an API for other components to access the database (configuration but also the execution tables).

Parameter Name	Parameter Value	Description	Configuration Method
Database/HostName	localhost	The name of the server where the database is hosted	Config file /etc/sen2agri/sen2agri-persistence.conf
Database/DatabaseName	sen4cap	The name of the Sen4CAP database	Config file /etc/sen2agri/sen2agri-persistence.conf
Database/UserName	Admin	The username used to connect to the database	Config file /etc/sen2agri/sen2agri-persistence.conf
Database/Password		The password used to connect to the database	Config file /etc/sen2agri/sen2agri-persistence.conf

Appendix A.3 Sen4CAP Services Download parameters

The Sen4CAP Services Download module is a component that manages the download of level-1 products (Sentinel-1, Sentinel-2 and Landsat8) from SciHub and USGS or from a local store. Also, if configured accordingly, the ESA level-2 products can be directly downloaded from SciHub or fetched from a local repository.

Parameter Name	Parameter Value	Description	Configuration Method
Write Directory for S2	/mnt/dwn_def/s2	The directory where the downloaded Sentinel-2 products will be saved	Sen4CAP Configurator
Write Directory for L8	/mnt/dwn_def/l8	The directory where the downloaded Landsat-8 products will be saved	Sen4CAP Configurator
Write Directory for S1	/mnt/dwn_def/s1	The directory where the downloaded Sentinel-1 products will be saved	Sen4CAP Configurator
site	Ex: "1";"NLD_test" "2"; "Italy"	The sites defined in the database, from which the downloaders are searching online products; the polygon of each site is projected in WGS84	Sen4CAP database, site table; this table is prefilled and the manual editing in the database is not advised.
shape_tiles_s2	Shapes defined for the acquisition plan	The shapes defined for the acquisition plan will be intersected with the polygon of each site, and the resulting shapes will be interrogated by the downloader	Sen4CAP database, shape_tiles_s2 table; this table is prefilled and the manual editing in the database is not advised.
shape_tiles_l8	Shapes defined for the acquisition plan of Landsat 8	The shapes defined for the acquisition plan will be intersected with the polygon of each site, and the resulting shapes will be interrogated by the landsat downloader	Sen4CAP database, shape_tiles_l8 table; this table is prefilled and the manual editing in the database is not advised.

satellite	"1";"sentinel2" "2";"landsat8" "3"; "sentinel1"			Sen4CAP database, site table; this table is prefilled and the manual editing in the database is not advised
downloader.use.esa.l2a	false	Specifies if the ESA L2A products should be downloaded from SciHub instead of downloading L1C and pre-process them with MAJA.		Sen4CAP Configurator
downloader.start.offset	2	Specifies for the S2/L8 products the interval before the start of the season where L1C/L1T need to be still downloaded for the MAJA training. To be set to 0 if L2A products are directly downloaded/fetched (no MAJA pre-processing).		Sen4CAP Configurator

The Sen4CAP Services Download is able to query for the list of products available from one source and to download the products from another source. For example, the list of available products for a season can be queried from SciHub but the download of the products to be performed from a local store. The "datasources" table contains the data sources used for querying and for downloading level-1 or ESA level-2 products. These parameters can be edited from the "Datasource" tab in the system web interface as described in section 4.2.1. The structure of the table is the following:

Column name	Description
satellite_id	The satellite ID for which the datasource is configured
name	The name of the datasource
scope	The scope of the datasource and may have the following values: 1 = query, 2 = download, 3 = query & download
fetch_mode	The product extraction mode: 1 - OVERWRITE - Products are downloaded from the remote site and the corresponding local product, if exists, is overwritten 2 - RESUME - Products are downloaded from the remote site and, if a corresponding local product exists, the download is resumed from the current length of the local product

	3 - COPY - Products are copied from a local (or shared) folder into the output folder. No remote download is performed
	4 - SYMLINK - Only a symlink to the product file system location, into the output folder, is created. No remote download is performed
	5 – Direct link to product – The path of the product is directly inserted into the database. No remote download or symbolic linking is performed
username	The user used for the remote connection
password	The password used for the remote connection
download_path	The path where the products are downloaded (should not be changed)
specific_params	Specific parameters for the datasource (should not be changed)
maximum_connexions	Maximum connections to be used at a moment to the datasource
local_root	The local root repository in the case of the products already downloaded in a local store
enabled	Specifies if the datasource is enabled or not
site_id	Binds the record to a specific site. This allows using a different data source (than the one globally configured for a satellite) for the specified site

Some of the parameters can also be edited in the /usr/share/sen2agri/sen2agri-services/config/ services.properties.

In the case of using a local repository that does not have the structure in the format .../2018/01/08, the following keys can be added/updated in the configuration file:

```
AWSDataSource.Sentinel2.usePadding=false
AWSDataSource.Landsat8.usePadding=false
```

In this way, repositories having the folder structure like ".../2018/1/8" will be considered.

Other keys that can be configured via the services.properties are given below.

Parameters that should not be modified unless administration is performed:

```
server.port = 8081
```

➔ This is the port on which the server listens for calls from the interface. Should not be changed unless port is already taken by another application. In this case, the port should be also updated in the file /var/www/html/ConfigParams.php

```
spring.datasource.url
```

➔ This key gives the connexion string to the database

```
spring.datasource.username=admin
spring.datasource.password=sen2agri
```

➔ Credentials for the sen2agri database

```
resolve.links.locally = false
```

➔ If set to "true", it "expands" symbolic links by copying locally the files and folders

Parameters that can be optionally changed:

```
database.config.polling=15
```

➔ Interval in minutes for checking for changes in the database and for execution jobs (ex. queries for datasources to check for new products or retry products interval). 0 means disabled

```
network.connexions.timeout=30
```

➔ If some datasources are responding very slow, this parameter can be set to a higher value. Pay attention that this applies to all datasources and might impact the performances of the system. To be increased if it is absolutely necessary.

IMPORTANT NOTE: Changing one of the keys in the "services.properties" file will require restarting the application using:

```
sudo systemctl restart sen2agri-services
```

Also, after changing datasources parameters in "services.properties" and restarting the application, should be checked the correct update of the "datasource" table using:

```
psql -U admin sen4cap -c "select * from datasource;"
```

If the parameters do not match, manual update of the datasource table might be necessary and restart the application.

IMPORTANT NOTE: In the sen2agri-services, compared with the old downloaders, the philosophy of downloading changed as searches are not performed anymore each time from the beginning of the season(s) but instead it searches in an incremental manner, from the last downloaded/aborted product.

With the sen4cap-services, in order to try to retry the aborted products they should be set to status 3 (FAILED) and in the same time reset the no_of_retries column to 0 (there is a retry job in the application that will perform all the retries automatically).

Nevertheless, in order to force a query from the beginning of the season (in the case of missed products), a small plugin was added to the application in order to be able to perform this operation.

This option should be activated only in exceptional situations or if the user really wants this and is aware about the disadvantages (much more requests to datasources, more processing time consuming on machine, possible higher delays in product availability and incompatibility with the near-realtime mode). This is why, this option is disabled by default.

To use this option, add in the "config" table the key "scheduled.lookup.all_products.enabled = true".

Once you notice that the products are up-to-date, you should remove (or set to false) the key from the database as it is not compatible with the near-realtime download mode.

In order to change the logging level of the sen2agri-services, the file /usr/share/sen4cap/sen2agri-services/config/application.properties can be edited for changing especially the following keys:

```
logging.level.ro.cs.tao=TRACE
```

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
	2.0	109	07/01/2021	common agricultural policy

```
logging.level.org.esa.sen2agri=TRACE
```

Where the possible values are for log levels are: ERROR, WARN, INFO, DEBUG, or TRACE.

Appendix A.4 Sen4CAP Services Sentinel-1 Pre-processing Parameters

The Sen4CAP Services S1 Pre-processing module is a component that is doing the amplitude and coherence extraction from the Sentinel-1 SLC products.

Some of its parameters are found in the "config" database table:

Parameter Name	Description	Possible Values	Default Value
processor.l2s1.acquisition.delay	The number of days to expect between the acquisition date and the availability of orbit files	<integer>	2
processor.l2s1.compute.amplitude	Enable backscatter preprocessing	true|false	true
processor.l2s1.compute.coherence	Enable coherence preprocessing	true|false	true
processor.l2s1.copy.locally	Copy L1 products in the working directory before preprocessing	true|false	false
processor.l2s1.crop.nodata	Further crop L2 products to reduce the NoData values on sides	true|false	true
processor.l2s1.crop.output	Crop the L2 products to the extent of the SLC intersection (for coherence)	true|false	true
processor.l2s1.enabled	Enable or disable S1 preprocessing	true|false	true
processor.l2s1.extract.histogram	Extract histogram from L2 products	true|false	true
processor.l2s1.gpt.parallelism	How many threads SNAP will use	<integer>	8
processor.l2s1.gpt.tile.cache.size	SNAP tile cache size in MB (influences the memory consumption)	<integer>	256
processor.l2s1.interval	Interval in minutes betweem running jobs	<integer>	60
processor.l2s1.join.amplitude.steps	Combine the amplitude steps in a single SNAP graph (not advisable)	true|false	false
processor.l2s1.join.coherence.steps	Combine the coherence steps in a single SNAP graph (not advisable)	true|false	false
processor.l2s1.keep.intermediate	Do not delete intermediate products and files	true|false	false
processor.l2s1.master	Master product for pair processing	S1A|S1B|OLDEST|NEWEST	S1B
processor.l2s1.min.disk	Minimum required available disk space for a job step (in MB)	<integer>	16384
processor.l2s1.min.intersection	Minimum percentage of intersection for consecutive SLC scenes	<float> between 0.0 and 1.0	0.05
processor.l2s1.min.memory	Minumum required available memory for SNAP job steps (in MB)	<integer>	8192

processor.l2s1.min.s2.intersection	Minimum percentage of intersection of a Sentinel-2 tile intersecting a SLC product	<float> between 0.0 and 1.0	0.05
processor.l2s1.output.extension	Extension of the L2 products (for SNAP)	<string>	.tif
processor.l2s1.output.format	Format of the L2 products (for SNAP)	GeoTIFF|BigTIFF-GeoTIFF|NetCDF-CF4	GeoTIFF
processor.l2s1.overwrite.existing	If true, the existing L2 products will be overwritten	true|false	false
processor.l2s1.parallel.steps.enabled	Enable the execution in parallels for some steps	true|false	true
processor.l2s1.parallelism	How many processings can be done in parallel	<integer>	1
processor.l2s1.path	Output location of the preprocessor	<string>	/mnt/archive/{site}/l2a-s1
processor.l2s1.pixel.spacing	Spatial resolution of the L2 products	10.0|15.0|20.0|30.0	20
processor.l2s1.polarisations	Polarisations to be processed	VV|VH|VV;VH	VV;VH
processor.l2s1.process.newest	Process starting with most recent acquisitions	true|false	false
processor.l2s1.projection	The projection of L2 products, either as WKT1 or as EPSG code	<string>	EPSG:3035
processor.l2s1.resolve.links	If true, the symlinks will be resolved to absolute paths	true|false	false
processor.l2s1.step.timeout	Maximum time, in minutes, a step can execute before being killed	<integer>	60
processor.l2s1.temporal.offset	The distance, in days, between S1 acquisitions	6|12	6
processor.l2s1.version	Preprocessor version	1|2	1
processor.l2s1.work.dir	Working directory location	<string>	/ssd
dem.name	The name (as known by SNAP) of the DEM to be used	<string>	SRTM 1Sec HGT
disk.monitor.interval	The interval, in seconds, to obtain disk usage statistics. 0 means disabled	<integer>	0
docker.gdal.image	The GDAL Docker image name	<string>	osgeo/gdal
docker.snap.image	The SNAP Docker image name	<string>	sen4cap/snap
downloader.s1.write-dir	The folder in which L1 products were downloaded	<string>	/mnt/archive/dwn_def/s1/default
plugins.use.docker	TAO plugins (info, quicklook) should use gdal from a Docker container	true|false	true
snap.dem.location	The folder into which the DEM is found	<string>	
snap.use.docker	Instruct the SNAP usage from a Docker container	true|false	true

Other parameters are found in the sen4cap services configuration file (services.properties):

Parameters that should not be modified unless administration is performed:

Parameter Name	Description	Possible Values	Default Value
dem.name	The name (as known by SNAP) of the DEM to be used	<string>	SRTM 1Sec HGT
docker.gdal.image	The GDAL Docker image name	<string>	osgeo/gdal
docker.snap.image	The SNAP Docker image name	<string>	sen4cap/snap
plugins.use.docker	TAO plugins (info, quicklook) should use gdal from a Docker container	true|false	true
snap.dem.location	The folder into which the DEM is found	<string>	
snap.use.docker	Instruct the SNAP usage from a Docker container	true|false	true

Parameters that can be optionally changed:

```
ScheduleManager.forced.site = <site_id>
```

> ➔ If present and not empty, it forces the download and Sentinel-1 processing just for the given site_id. Usually it is not set.

Appendix A.5 Using Sentinel1 and Sentinel2 Download Services with Local L1C Store

If Sentinel-1, Sentinel-2 and Landsat8 low-level products are already downloaded and available in a store , the Download Services can be configured in order to use directly these stores instead of downloading the low-level products from SciHub and USGS. In this case, after the successful installation of the Sen4CAP system, the "*datasource*" table can be edited in order to specify the store.

In order to activate the local store mode, the line corresponding to the download datasource should be updated. For example, for the S2 products (having the column satellite_id = 1 and the scope=2) :

- Set the "local_root" column to the root directory where the L1C products are stored (see "<PATH_TO_LOCAL_EO_DATA>" below);
- Set the "fetch_mode" column value to 4 (symlink mode) or 3 (copy products mode)

The "<PATH_TO_LOCAL_EO_DATA>" should be the root directory where the L1C products are stored in a hierarchical folder structure yyyy/MM/dd/<Products> like in the following example, where the first level is the year, the second level of folders is the month of the year, the third level of folders is the day of the month:

```
2015
2016
2017
   01
```

```
02

03

   01

   02

   03

   04

      S2A_MSIL1C_<xxxx1>.SAFE

      S2A_MSIL1C_<xxxx1>.SAFE
```

These parameters can also be edited in the /usr/share/sen4cap/sen2agri-services/config/ services.properties by setting, for example, for IPT repository and the option of creating symlinks:

```
AWSDataSource.Sentinel2.local_archive_path=/eodata/Sentinel-2/MSI/L1C

AWSDataSource.Sentinel2.fetch_mode=4
```

The general settings for a DIAS have the following form:

```
<DataSourceClassName>.<Satellite>.path.builder.class                    =
org.esa.sen4cap.dias.<dias>.Sentinel1PathBuilder

<DataSourceClassName>.<Satellite>.local.archive.path.format = <tokens>

<DataSourceClassName>.<Satellite>.path.suffix = none|.SAFE|<other>

<DataSourceClassName>.<Satellite>.product.format = folder|zip
```

The configuration file of the services contain sections for all the DIASes (except CreoDIAS, which doesn't need one). It is enough to uncomment the items from a section (and comment the others).

After changing configuration file or the database, the changes normally should be refreshed in maximum 1 minute otherwise the Sen2Agri Services service should be restarted using:

```
sudo systemctl restart sen2agri-services
```

Appendix A.6 Using AsterDEM for S1 pre-processing

To use AsterDEM for the S1 pre-processing the following steps should be performed:

1. Ensure that SNAP 8 docker container is installed on the processing machine. In order to do this, run the update script from the installation package with the following parameters:

```
# cd /path/to/Sen4CAPDistribution/install_script/

# sudo ./update.sh --snap8-docker true
```

Or simply:

```
# sudo ./update.sh --snap8-docker true
```

2. Download the archive ASTER_Northern_Europe.zip and unzip it in a location like the following:

```
# mkdir -p /mnt/archive/dem/ASTER_Northern_Europe/
```

```
# unzip ASTER_Northern_Europe.zip -d /mnt/archive/dem/ASTER_Northern_Europe/
```

3. Edit the file /usr/share/sen2agri/sen2agri-services/config/services.properties in order to add/update the following keys in the following manner:

```
# SNAP 8 docker container
snap.use.docker = true
docker.snap.image = sen4cap/snap


# SNAP 8 ASTER DEM
dem.name = ASTER 1sec GDEM v3
snap.dem.location = /mnt/archive/dem/ASTER_Northern_Europe/


#physical location to be mapped for .snap from within container
snap.home.location = /mnt/archive/snap_tmp/
```

Ensure that there are no other keys like the ones above that are uncommented.

4. Please ensure that the folder /mnt/archive/snap_tmp/ exists and has write access for everybody:

```
# sudo chmod -R 777 /mnt/archive/snap_tmp/
```

5. Stop the sen4cap services:

```
# sudo systemctl stop sen2agri-services
```

6. Reset the status of any failed S1 SLC products in order to be reprocessed :

```
# sudo psql -U admin sen4cap -c "delete from l1_tile_history where
downloader_history_id in (select id from downloader_history where
satellite_id = 3 and status_id = 6 and site_id = <site_id>)"

# sudo psql -U admin sen4cap -c "update downloader_history set status_id
= 2 where satellite_id = 3 and status_id = 6 and site_id = <site_id>)"
```

where <site_id> is the site you want to process using AsterDEM. If all the sites need to be pre-processed with AsterDEM, the part "and site_id = <site_id>" could be skipped from the above commands.

7. Start the sen4cap services using the command :

```
# sudo systemctl start sen2agri-services
```

Appendix A.7 Using other DEM for S2 L2A pre-processing

The system can use different DEMs for the Sentinel-2 L2A pre-processing.
Currently, the following ones are supported:
- SRTM 3 arc-second, uncompressed, e.g. "srtm_69_11.tif"
- SRTM 1 arc-second, uncompressed or compressed, e.g. "N44E023.SRTMGL1.hgt.zip" or "N44E023.hgt"
- ASTER GDEM v3, uncompressed or compressed, e.g. "ASTGTMV003_N69W023.zip" or "ASTGTMV003_N69W023_dem.tif"
- EU-DEM v1.1, uncompressed, e.g. "eu_dem_v11_E60N20.TIF"

The DEM type is automatically detected by enumerating the files in the configured directory.
Combining different DEMs in the same directory is not supported.

To use, for example, EU-DEM for the S2 pre-processing the following steps should be performed:

1. Copy the EU-DEM files into a directory, for example, "/mnt/archive/EU-DEM"
2. If you want EU-DEM to be used for all the sites in the system, update the key "processor.l2a.srtm-path" in the "config" table for the new location:

```
# psql -U admin sen4cap -c "update config set value = '/mnt/archive/EU-
DEM' where key = 'processor.l2a.srtm-path'"
```

3. If you want EU-DEM to be used only for a single site in the system, insert the key "processor.l2a.srtm-path" in the "config" table for the new location and the desired site:

```
# psql -U admin sen4cap -c "insert into config (key, site_id, value)
values ('processor.l2a.srtm-path', <your_site_id>, '/mnt/archive/EU-DEM')"
```

Where <your_site_id> is the site id that you want to process using EU-DEM.

Appendix B - Sen4CAP database tables for manual configuration

The table below presents the main configuration tables that might present interest for the advanced users.

Table name	Description
Site	The table contains all sites that are used in the system
Processor	The table contains all processors that are used in the system
Config_category	The table contains the configuration for the type of applications that can be configured. Examples: Archiver, Executor, L4A Crop Mask Processor, etc.
Config_metadata	The table contains the parameters that can be configured for each type of application contained in Config_category table. The following columns can be modified: - the friendly name for the parameter - the type of the parameter - is_advanced – specifies if the parameter can be configured by an user with admin role - config_category_id – the id from Config_category table
Config	The table contains default values for the parameters of each configured parameters: - key – the name of the parameter - site_id – the id of the site, if the parameter is configured for a specific site - value – the default value of the parameter - last_updated – the time when the parameter was last updated

Apart from these tables, there are also the execution tables that are used by the system but these tables not recommended to be edited by the users.

In order to access the database, the following command can be issued in order to access to the PostgreSQL console:

```
$ psql -U admin sen4cap

sen4cap=#
```

The prompt "sen4cap=#" can be used to issue SQL commands like the following that displays the content of the table "config":

```
sen4cap=# select * from config;
```

Please note that all commands in this console should end by a semicolon.

Another option, if the Linux was installed with an X server, is to use the much easier to use application, *pgadmin3*. In order to install and run this application, the following commands can be executed:

```
$ sudo yum install -y pgadmin3

$ pgadmin3 &
```

In this application, a new connexion to the database server can be created using the parameters from the picture below (database password is *"sen2agri"*):

Appendix C - System performance example

System performances on an IPT machine used as calculation basis (8 vCores, 128 gb ram)

	Products	Tile(s) / Orbit(s)	Duration
MAJA	1	1	**30 min**
S1 Preprocessing	1	1	**30 min**
LAI Mono date	1	1	**20 min**
CropType			
GrasslandMowing			
Agricultural Practices	**2057 NDVI Tiles** **1751 Coherences** **1261 Amplitudes**	**14 S2 tiles** **8 S1 orbits**	**10 hours**

Appendix D - Building from source

In order to build the system, the script Sen2AgriBuildAll.sh (from the git sources sen2agri/packaging) can be used. This script presents to the user the following menu:

```
*********************************************

** 1) Install BUILD PREREQUISITES

** 2) BUILD Sen2AgriPlatform : OTB and GDAL

** 3) BUILD Sen2AgriProcessors, Downloaders and Demmaccs

** 4) BUILD Sen2AgriApplication

** 5) BUILD Sen2AgriWebsite

** 6) BUILD Selective: Enter list of options (ex: platform, processors,
website, app)

** 7) BUILD Sen2Agri All Components

*********************************************

Please enter a menu option and enter or enter to exit.
```

The user can choose one of the option in order to build the desired component(s).

In the next 2 paragraphs are presented the options to build the system. Both of these options can be also implemented using the above interface and by selecting the corresponding menus.

Building the core components

For building the core components (used for manual execution) run the steps 1, 2 and 3 from the general installation script Sen2AgriBuildAll.sh.

Building the components for automated usage

Run step 7 from the general installation script Sen2AgriBuildAll.sh.

Preparing the distribution

In order to create a Sen4CAP system distribution the following steps should be followed:

- Execute Sen2AgriBuildAll.sh

Execute install_platform/sen2agriCreateDistribution.sh by providing the previously created folder with the Sen4CAP system RPM files.

Appendix E – Advanced system configuration items

Appendix E.1 Processing System Configuration Utility

The Sen4CAP Configurator is based on the Sen2-Agri Configurator and is a Linux desktop application used to configure the system parameters. In order to be launched, the X server graphical interface should have been installed during the operating system installation.

If CentOS is already installed without a graphical interface, this can be installed with the following commands:

```
$ sudo yum groupinstall "X Window System" "Desktop" "Desktop Platform"

$ sudo yum install gdm
```

If you would like this to be the default runlevel, the file "/etc/inittab" can be edited and set the default level to 5 instead of 3:

```
$ sudo vi /etc/inittab
```

Change:

```
id:3:initdefault:

To:

id:5:initdefault:
```

The Sen4CAP configurator can be launched with the following command:

```
$ sen2agri-config &
```

The parameters that will appear in the Sen4CAP Configurator are managed through the Sen4CAP database and, depending on their type, they can be modified either by an admin or non-admin user. All parameters values that are modified are validated by the configurator (see Figure 0-1).
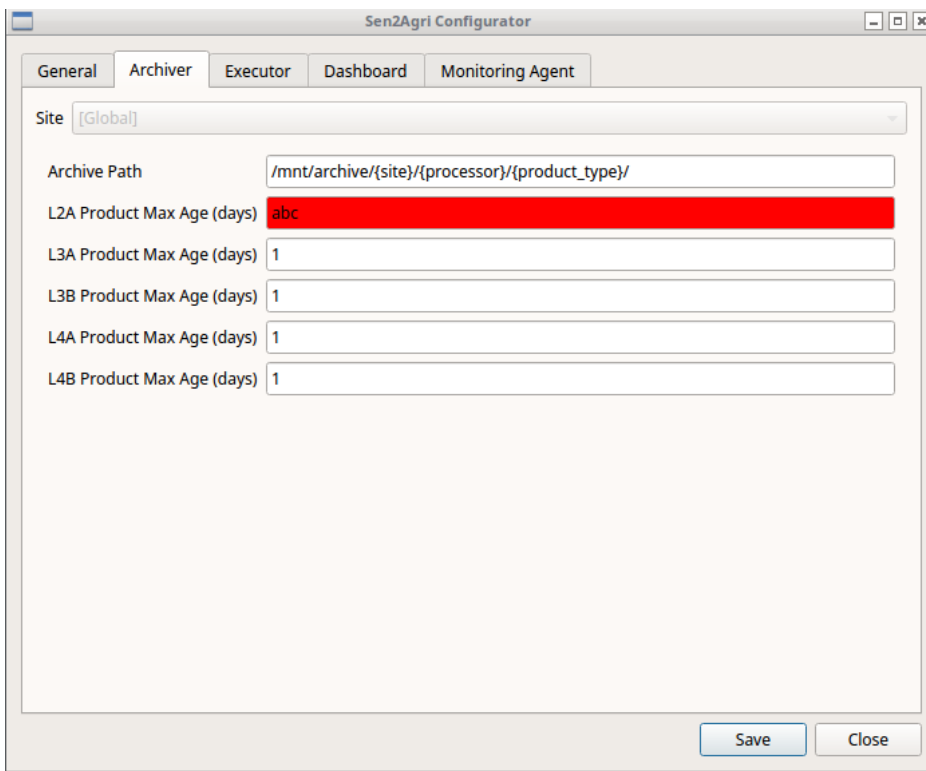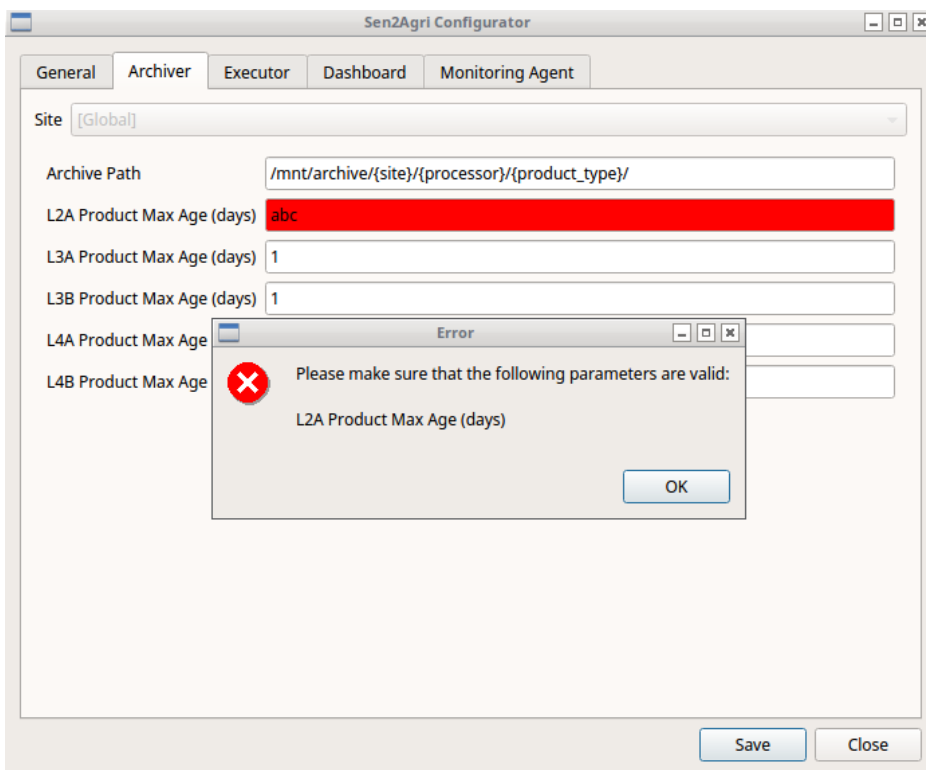
Figure 0-1. Sen4CAP Configurator – parameter validation

If the user tries to save when an invalid parameter value was filled in, an error message will inform the user about the error and the values are not saved.

Figure 0-2. Sen4CAP Configurator – parameter validation – error message

Some of the parameters are site specific while other ones are system specific. For the site specific parameters, the user has the possibility to select the site for which he is configuring the parameter (Figure 0-3).
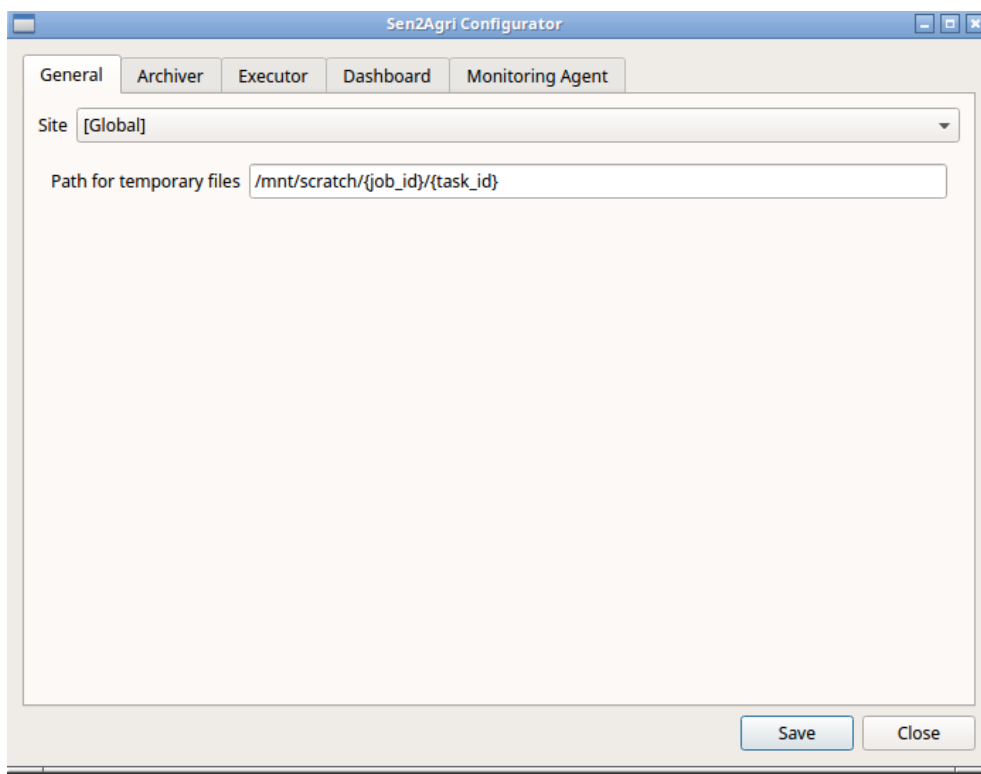
Figure 0-3. Sen4CAP Configurator – general tab – user with admin role

The modification of the value of a parameter can be done either by site (by selecting a site and then modifying the value) or for all sites (by selecting **Global** and then modifying the value). In order to modify the value of a parameter only for a site, the **Customize** button will be used for values that were not set before for this site. To reset the value for the current site back to the global value, the **Reset** button can be used. This is illustrated in Figure 0-4.
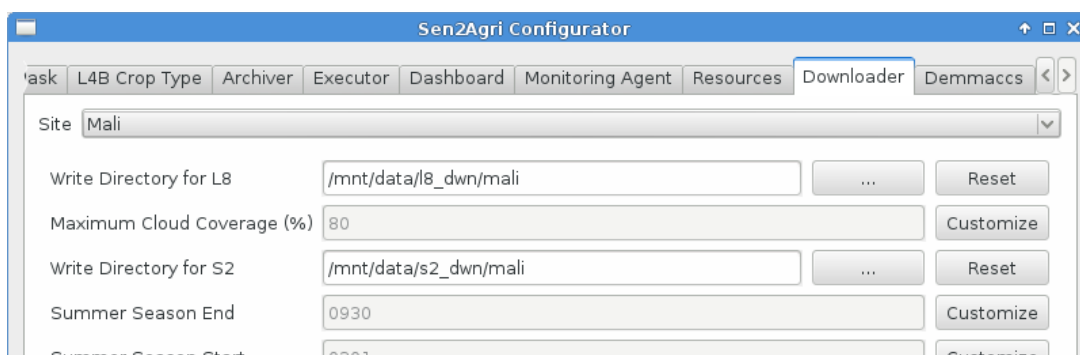
Figure 0-4. Sen4CAP Configurator – customize parameter value

If a parameter was already defined for a specific site, changing the parameter for **Global** will not affect the already customized specific parameter.

	Ref	Sen4CAP_SUM_v2.0		
.esa	Issue	Page	Date	**sen4cap**
	2.0	122	07/01/2021	common agricultural policy

A non-admin user will only be able to see the information in the Sen4CAP interface but will not be able to modify it (Figure 0-5).
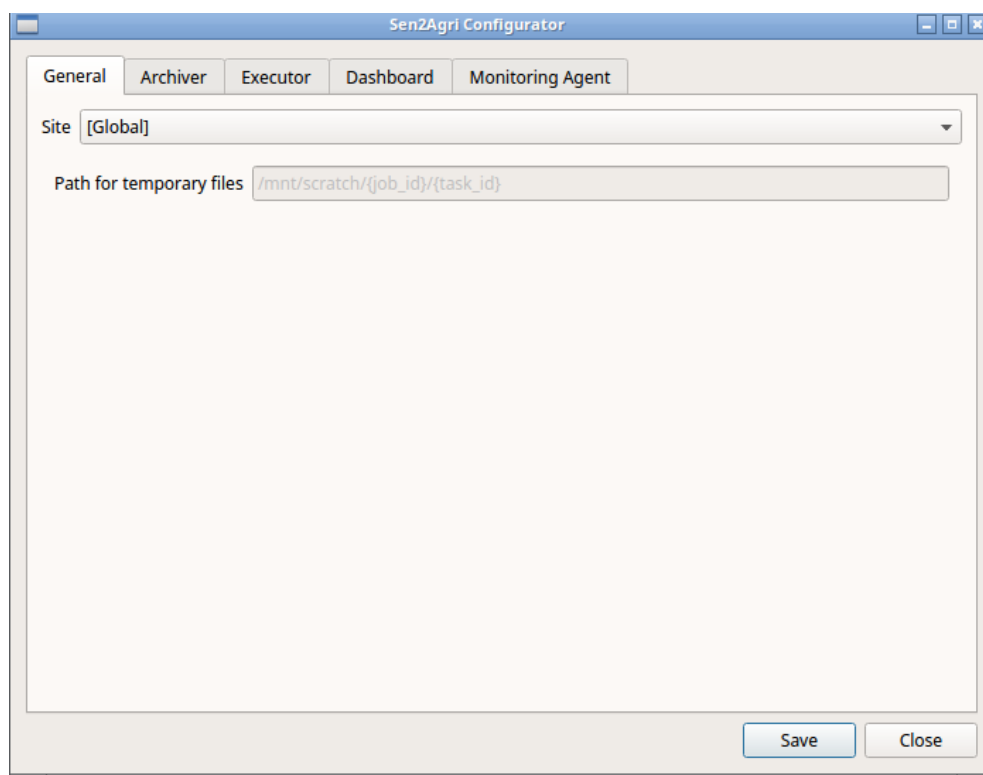
Figure 0-5. Sen4CAP Configurator – general tab – user without admin role

Appendix E.2 Sen4CAP Archiver

The Sen4CAP Archiver is a system service which interrogates the execution database for products that need to be archived. It will receive a list of products, each product having a current path (source directory) and a destination path (destination directory).

The Configurator contains a tab dedicated to configure the Archiver (Figure 0-6):
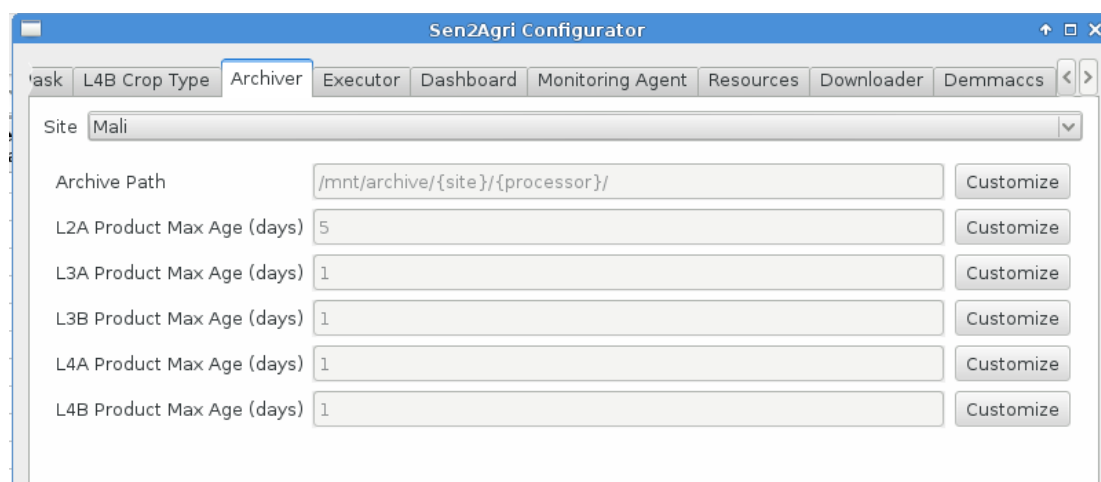
Figure 0-6. Sen4CAP Configurator – Archiver tab

For details about the parameters of Sen4CAP Archiver, see Appendix E.9 All advanced parameters table.

Appendix E.3 Sen4CAP HTTP Listener

The Sen4CAP HTTP Listener is a system service used to offer the information displayed in the Execution Dashboard and to communicate with the Monitor Agent to receive the information that it will send to the dashboard.

The Configurator contains a tab dedicated to the HTTP Listener configuration, called Dashboard (Figure 0-7).
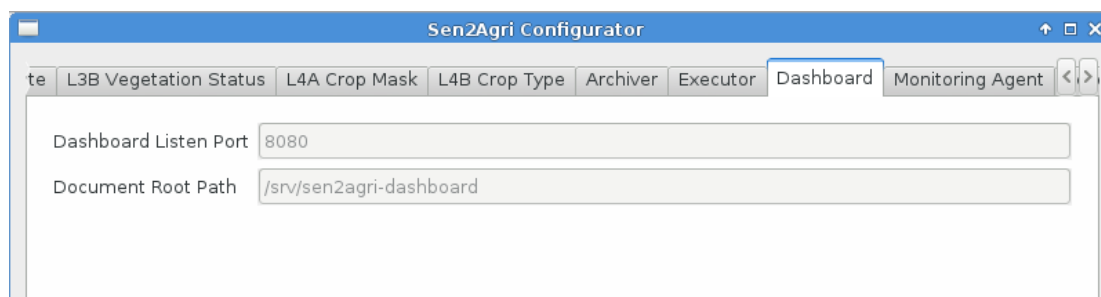
Figure 0-7. Sen4CAP Configurator – Dashboard tab

For details about the parameters of Sen4CAP HTTP Listener, see Appendix E.9 All advanced parameters table.

Appendix E.4 Sen4CAP Executor

The Sen4CAP Executor is a system service that uses SLURM to execute the processors.

The Configurator contains a tab dedicated to the Executor configuration (Figure 0-8).
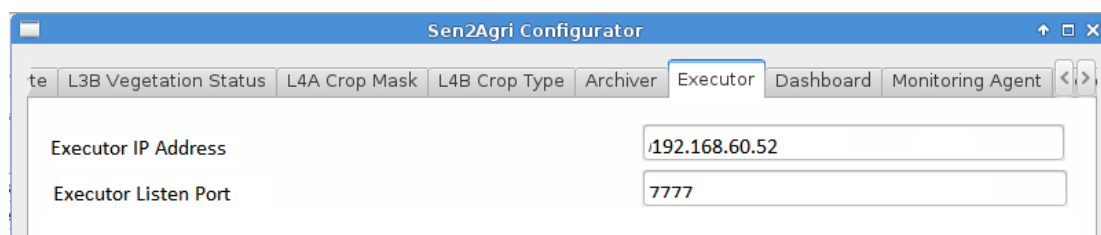
Figure 0-8. Sen4CAP Configurator – Executor tab

The parameters used by the Executor are described in Appendix A.1 Sen4CAP Executor parameters.

Appendix E.5 Sen4CAP Monitor Agent

The Monitor Agent service is used to monitor the Sen4CAP processing nodes and sends metrics to the HTTP Listener in order to be displayed in the Execution Dashboard.

The Configurator contains a tab dedicated to configure the Monitor Agent (Figure 0-9).

Figure 0-9. Sen4CAP Configurator – Monitoring Agent tab

Find the parameters used by the Monitor Agent Service in XXX.

Appendix E.6 Sen4CAP Persistence Manager

The Persistence Manager exchanges information between the database and the other system components (Executor, Monitor Agent, Dashboard and Archiver).

The parameters used by the Persistence Manager are described in Appendix A.2 Sen4CAP Persistence Manager parameters.

Appendix E.7 Sen4CAP Downloader

The S2 L1C and L8 L1T products are downloaded automatically by the downloader processor installed and configured by the installation script described in section 3.3. This processor is scheduled to run periodically (hourly), as configured by the installation script. Its job is to download the available products/tiles from the ESA SciHub for S2 products and from https://ers.cr.usgs.gov/login (for L8 products) according to the parameters described in Appendix A.3 Sen4CAP Services Download parameters.

The configuration parameters (from **config** table) can have values for each site id, and also a global value, for all sites. In the case the key is missing for a certain site, the global value is considered. Even if the database parameters are described here, they can be changed also from the Sen4CAP Configurator application, so the direct database manipulation is not advised.

Some manual configuration should be performed in order to set up a proxy configuration if needed, please see 3.3.3.1.

To configure the application by using the Sen4CAP Configurator, just launch the configurator as shown in Figure 0-10.
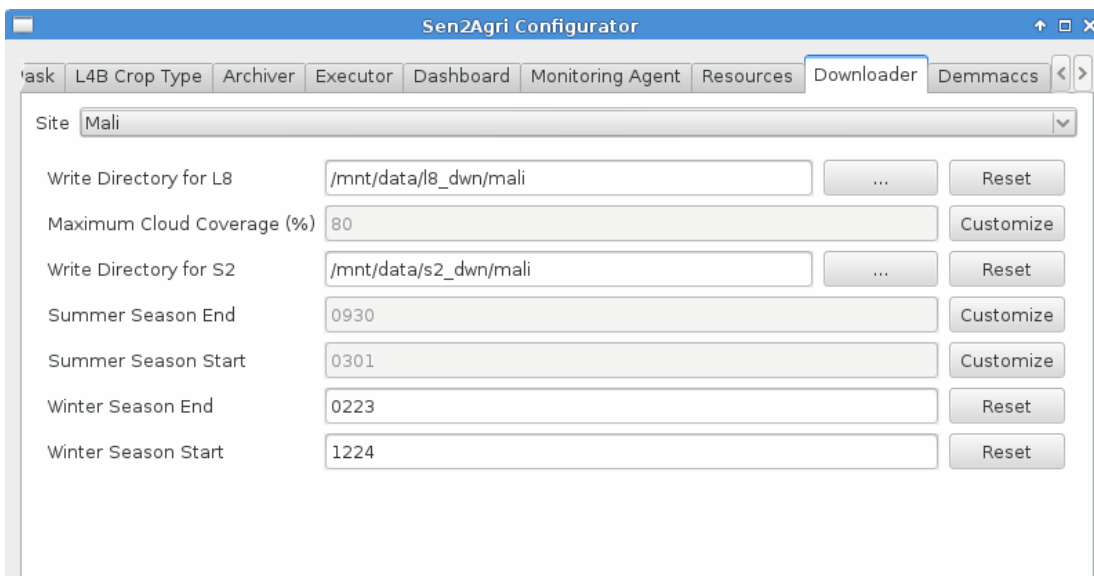
Figure 0-10. Sen4CAP Configurator – Downloader tab

When the downloader processor is launched, it interrogates the database to intersect the polygons of the sites with the shapes defined in the sentinel tiles (**shape_tile_s2**) table (see Appendix B - Sen4CAP database tables for manual configuration for database tables). For the resulting tiles, the downloader interrogates scihub site https://scihub.copernicus.eu/apihub/search?q= and the landsat site https://ers.cr.usgs.gov/login about the available products for the defined season period for each configured site. From the available products, the downloader checks which of them were already downloaded (this information is located in table **downloader_history**). After the new products are downloaded, they are added in the **downloader_history** table, with the status **not processed**. Later on, the demmaccs processor (see Appendix E.8 Sen4CAP Demmaccs_launcher) will handle all the **not processed** products found in the **downloader_history** table.

Appendix E.8 Sen4CAP Demmaccs_launcher

The demmaccs_launcher processor is also installed and configured by the installation script described in section 3.3. It handles all the fully downloaded products from the **downloader_history** table with the status **not_processed** and its main goal is to check the database, to create a list with the unprocessed L1C and L1T products, to launch with the proper arguments the **demmaccs** processor which in its turn will launch the dem and maccs processors (see section 4.4.1) and to write the results in the database in both **downloader_history** and **product** tables. The demmaccs processor is able to launch the dem and maccs processors in parallel to minimize the processing time (see section 4.4.2.1 *–processes-number-dem* and *–processes-number-maccs* input arguments).

Appendix E.9 All advanced parameters table

The following configurations are used in the automatic mode of the Sen4CAP system. They can be changed using the Procesing System Configuration Utility described in paragraph Appendix E.1 Processing System Configuration Utility.

The advanced processor parameters are prefixed with:

- LAI: "processor.l3b.*". These parameters can be changed in the "L3A Vegetation Status" of the System Cofiguration Utility;

	Ref	Sen4CAP_SUM_v2.0		
esa	Issue	Page	Date	sen4cap
	2.0	126	07/01/2021	common agricultural policy

- CropType: "processor. s4c_l4a.*". These parameters can be changed in the "L4A Crop Type" of the System Cofiguration Utility;

- GrasslandMowing: "processor.s4c_l4b.*". These parameters can be changed in the "Grassland Mowing " of the System Cofiguration Utility;

- AgriculturalPractices: "processor.s4c_l4c.*". These parameters can be changed in the "L4C Agricultural practices" of the System Cofiguration Utility;

Appendix E.10 Advanced useful scripts

- **delete_site.py** – allows deleting an existing site from the system

Examples:

```
delete_site.py -s site_short_name

OR

delete_site.py -s site_short_name -d false -e false -a false -l false -m false -t false
```

The "site_short_name" is the short name of the site from the "Sites" page of the Sen4CAP GUI. The first example above removes only the site from the database without removing the files on disk. The second example removes also the files from the disk (downloaded or processed).

Use `delete_site.py –help` to see the list of options and details.

- **filter_site_download_tiles.py** – allows filtering, enabling or disabling the downloads for a certain site.

Example 1 to disable the download for all L8 product tiles for the site "test". The value -t 2 is for L8, if you want to disable S2 use –t 1:

```
filter_site_download_tiles.py -t 2 -s test -e false
```

Example 2 to enable the download for all L8 product tiles that intersect the area defined for the site "test".

```
filter_site_download_tiles.py -t 2 -s test -e true
```

Example 3 to filter during downloading the S2 tiles for the site "test" to only the two specified by the parameter "–l" (they need to be comma followed by a space separated).

```
        filter_site_download_tiles.py -t 1 -s test -e true -l "34RFP, 43FP"
```

IMPORTANT NOTE: The specified tiles need to be tiles that intersect the site extent otherwise the downloader will download nothing. This is a filter parameter and not a manner to replace the site extent. It should be used only to eliminate the tiles that have only small areas intersecting with your site and to keep only the relevant ones.

The script allows to the user controlling individually the download of each site. For example, you can disable the download of L8 for 2 sites but not for the 3rd. Nevertheless, after configuring these filters (especially if you enable per site downloads), you need eventually to restart the downloaders using *"sudo systemctl restart sen2agri-services"*.

IMPORTANT NOTE: The scripts should use them with care, only if you really need them and you fully understood how they are functioning (see also the help for the parameters). If incorrectly used, you might delete your site and your products or your downloader will not work anymore.

Use `filter_site_download_tiles.py –help` to see the list of options.

- **insert_l2a_product_to_db.py** – allows inserting into a site the L2A products processed in another location. See the help of the script for details.

Appendix E.11 Crop code LUT file structure

A crop type look-up table (LUT) is used to provide extra information about the crop codes present in the subsidy application dataset. It is used during the preparation step to set a unique crop code (integer) by declared crop type (and a corresponding name) and to define a broad land cover category to which each crop type belongs:

- 0: other natural areas;
- 1: annual crop;
- 2: permanent crop;
- 3: grassland;
- 4: fallow land;
- 5: greenhouse and nursery.

It is a CSV file that must contain the following columns:

Field name	Role	Default value [format]
Ori_crop	The initial crop code from the subsidy application dataset	[integer or string]
CTnum	The new crop type code (each Ori_crop being associated to a unique CTnum)	[integer]
CT	The name of the crop type in English	[string]
LC	The main land cover class of the crop type: 0: other natural areas 1: annual crop 2: permanent crop 3: grassland 4: fallow land 5: greenhouse and nursery	[integer]
CTnumL4A	The new crop type code resulting of the possible grouping of the CTnum for the classification	[integer]
CTL4A	The crop type name associated to CTnumL4A	[string]
CTnumDIV	The crop code specific for the crop diversification assessment	[integer]
CTDIV	The crop name specific for the crop diversification assessment	[string]
EAA	Eligible Agricultural Area: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
AL	Arable Land: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
PGrass	Permanent grassland: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]

TGrass	Temporary grassland: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
Fallow	Fallow land: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
Cwater	Crop under water: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]

IMPORTANT NOTE: It is recommended before importing the LUT file to convert it into a UTF-8 encoding file.

Appendix E.12 Grassland mowing configuration file

In order to execute, the Grassland mowing shall be configured using an ini format configuration file. The system provides a default version of this file but some of the fields need to be customized according to each country. In the default file were preserved the values for each pilot country, as an example.

```
[GENERAL]
; Specify here the CTnum filter, if any
CTNUM_FILTER =

 [S1_constants]
; for all countries
S1_time_interval = 6
SAR_spacing = 20
        ; meters
cohe_ENL = 100
        ;20*5
min_cohe_var = 0.024
        ; ML=2*8; s1_res=3.0*22.; s1_smp=2.3*17.4; cohe_var.cohe_var(0.2, ML*s1_smp/s1_res)
locAcqTimeASC = 18:00:00
locAcqTimeDESC = 06:00:00

[S2_constants]
; for all countries
S2_time_interval = 5
        ; meters
locAcqTime = 10:30:00

[S2_model]
; for all countries
apply_model = False
modelDir = model
NDVI_nomow_model_perc = 95.
sampling_days = 1.
```

```
minimum_parcels_th = 20
model_temporal_range_str = ('20190101 00:00:00', '20191231 23:59:59')
start_params = (0.05, 50., 0.05, 340.)
bounds = ([0.02, 0., 0.02, 180.], [1, 180., 1, 360.])


[S1_processing]
; for all countries
pol_types = VH
data_types = COHE
invalid_data = nan
saturate_sigma = True
pfa = 3.0e-7
stat_smpl_n = 5
non_overlap_interval_days = 28
    ; days
options_layer_burning = 'ALL_TOUCHED=False'
    ; list of gdal options comma separated
erode_pixels = 0


[S2_processing]
; NDVI
; Czech Rep
prod_type_list = SNDVI
sc_fact = 1000
corrupted_th = 0.1
invalid_data = -10000
decreasing_abs_th = 0.12
decreasing_rate_th = -0.000001
increasing_rate_th = 0.05
high_abs_th = 0.72
low_abs_th = 0.4


; Lithuania
; prod_type_list = SNDVI
; sc_fact = 1000
; corrupted_th = 0.1
; invalid_data = -10000
; decreasing_abs_th = 0.12
; decreasing_rate_th = -0.000001
; increasing_rate_th = 0.06
; high_abs_th = 0.75
; low_abs_th = 0.5
```

```
; Netherlands
; prod_type_list = SNDVI
; sc_fact = 1000
; corrupted_th = 0.1
; invalid_data = -10000
; decreasing_abs_th = 0.12
; decreasing_rate_th = -0.000001
; increasing_rate_th = 0.9
; high_abs_th = 0.75
; low_abs_th = 0.5


; Spain
; prod_type_list = SNDVI
; sc_fact = 1000
; corrupted_th = 0.1
; invalid_data = -10000
; decreasing_abs_th = 0.15
; decreasing_rate_th = -0.000001
; increasing_rate_th = 0.9
; high_abs_th = 0.75
; low_abs_th = 0.5


; Romania
; prod_type_list = SNDVI
; sc_fact = 1000
; corrupted_th = 0.1
; invalid_data = -10000
; decreasing_abs_th = 0.15
; decreasing_rate_th = -0.000001
; increasing_rate_th = 0.9
; high_abs_th = 0.75
; low_abs_th = 0.5


; Parameters currently not in use
; decreasing_rate = - [doy-1]
; LAI
;prod_type_list = SLAIMONO
;sc_fact = 1000
;corrupted_th = 0.1
;invalid_data = -10000
```

```
;decreasing_abs_th = 0.8

;decreasing_rate_th = -0.01

;; decreasing_rate_th [doy-1]

;

;prod_type_list = SFAPARMONO

;sc_fact = 1000

;corrupted_th = 0.1

;invalid_data = -10000

;decreasing_abs_th = 0.014

;decreasing_rate_th = -0.01

;; decreasing_rate_th [doy-1]

;

; other parameters

erode_pixels = 0

options_layer_burning = ['ALL_TOUCHED=False']

no_mowing_after_det = 60

non_overlap_interval_days = 28

stat_smpl_n = 0


[compliancy]


crop_codes =

crop_time_intervals =

crop_rule =


; Czech Rep

; crop_codes = 315, 350, 3001

;  crop_time_intervals = ('01/04/2019', '31/07/2019'), ('01/04/2019', '31/07/2019'), ('01/04/2019',
'31/07/2019')

; crop_rule = 0, 0, 0


; Lithuania

; crop_codes = GPZ, DGP, GPA, EPT, MNP, SPT, 5PT-2, MNS

;  crop_time_intervals = ('01/04/2019', '31/07/2019'), ('01/04/2019', '31/07/2019'), ('01/04/2019',
'31/07/2019'), ('01/05/2019', '30/10/2019'), ('01/07/2019', '01/10/2019'), ('15/07/2019', '15/10/2019'),
('15/07/2019', '01/03/2020'), ('01/08/2019', '01/10/2019')

; crop_rule = 0, 0, 0, 0, 0, 0, 0, 0


; Netherlands

; crop_codes = 265, 331, 336, 266, 332, 333, 334, 370, 372, 383, 1921, 3506, 3509, 3512, 3513, 3519,
3522, 3523, 3805, 3807, 3808

;  crop_time_intervals = ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019',
'31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'),
('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019',
'31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'),
```

```
('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019',
'31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019'),
('01/04/2019', '31/10/2019')
```

; crop_rule = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

; Spain

; crop_codes = 2, 85

; crop_time_intervals = ('01/04/2019', '31/10/2019'), ('01/04/2019', '31/10/2019')

;crop_rule = 0, 0

; Romania

; crop_codes = 450, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 660, 661, 662, 663, 671

```
;  crop_time_intervals = ('01/05/2019',  '31/10/2019'), ('01/05/2019',  '31/10/2019'), ('01/05/2019',
'31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'),
('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019',
'31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'),
('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019'), ('01/05/2019', '31/10/2019')
```

; crop_rule = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

[validation]

fwd_tol_days = 10

    ; days

bkw_tol_days = 10

    ; days

validation_temporal_range_str = ('20190401 00:00:00', '20190815 23:59:59')

Appendix E.13 Agricultural practices configuration file

The Agricultural Practices processor can be configured with a file whose default content is given below. Some of the parameters are advanced and normally do not change but some of them can be customized. Within the application can be found a list of files that were used for the pilot countries.

The structure of the configuration file is presented below:

```
[COMMON]

# Nothing here yet


[DEFAULT_TIME_SERIES_ANALYSIS_PARAMS]

# IMPORTANT: Please do not remove any of these keys from this section. Just set them to a value
or to nothing

OPTTHRVEGCYCLE=350

NDVIDW=300

NDVIUP=350

NDVISTEP=5

OPTTHRMIN=100

COHTHRBASE=0.05

COHTHRHIGH=0.15
```

```
COHTHRABS=0.75

AMPTHRMIN=0.1

CATCHMAIN=""

CATCHPERIOD=56

CATCHPERIODSTART=""

CATCHCROPISMAIN=""

CATCHPROPORTION=""

EFANDVITHR=325

EFANDVIUP=400

EFANDVIDW=300

EFACOHCHANGE=0.2

EFACOHVALUE=0.7

EFANDVIMIN="-10000"

EFAAMPTHR="-10000"

STDDEVINAMPTHR=1

OPTTHRBUFDEN=4

AMPTHRBREAKDEN=6

AMPTHRVALDEN=2

FLMARKSTARTDATE=

FLMARKSTENDDATE=


[CC_TIME_SERIES_ANALYSIS_PARAMS]

# Overwrite the needed default values here by adding the prefix CC


[FL_TIME_SERIES_ANALYSIS_PARAMS]

# Overwrite the needed default values here by adding the prefix FL


[NFC_TIME_SERIES_ANALYSIS_PARAMS]

# Overwrite the needed default values here by adding the prefix NFC


[NA_TIME_SERIES_ANALYSIS_PARAMS]

# Overwrite the needed default values here by adding the prefix NA (like the ones below)


NA_EFANDVITHR="-10000"

NA_EFANDVIUP="-10000"

NA_EFANDVIDW="-10000"

NA_EFACOHCHANGE="-10000"

NA_EFACOHVALUE="-10000"
```

	Ref	Sen4CAP_SUM_v2.0		
.esa	Issue	Page	Date	**sen4cap**
	2.0	134	07/01/2021	common agricultural policy

Appendix E.14 Agricultural practices table file

The agricultural input table files need to be provided for each configured practice for a site.

These files provide information about the main crop code and vegetation and practices dates. An example for the Catch Crop input table file to be uploaded in the system can be found below. Please note that these files are ingested by the system and preprocessed by the system, generating derived files that are used after that by the Agricultural Practices processor (the idea is that these files are not used directly by the processor itself).

IMPORTANT NOTE: the timestamp in this file are expected to be in the format YYYY-MM-DD.

```
FIELD_ID,MAIN_CROP,VEG_START,H_START,H_END,PRACTICE,P_TYPE,P_START,P_END
31.0000003730124.001,1933,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003487830.001,242,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004049812.001,3506,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003758331.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003832018.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003465920.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003525102.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003870820.002,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003456194.001,1932,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003525217.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004201395.001,1933,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003536452.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_3,2019-10-15,NA
31.0000003647133.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003730114.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003903829.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004085345.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003944796.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003558141.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003483300.001,3736,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004060852.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003525117.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003843449.001,1933,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003647129.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004023006.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004183370.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003835241.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003772407.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003446401.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004201360.001,383,2019-05-20,2019-07-01,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003880664.001,382,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003695251.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003910406.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
```

```
31.0000003776926.001,236,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004161526.001,262,2019-05-20,2019-08-01,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003801401.002,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004073838.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003871497.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004219766.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004077392.001,236,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003977038.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003685668.002,235,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003971560.001,236,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003972791.001,266,2019-05-20,2019-06-03,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003735293.001,383,2019-05-20,2019-07-01,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004018872.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_2,2019-10-15,NA
31.0000003570176.002,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003869569.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003497925.001,383,2019-05-20,2019-07-01,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003873564.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000004162925.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003508694.001,233,2019-05-20,2019-07-15,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003843451.001,2014,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003515828.001,383,2019-05-20,2019-07-01,2019-10-15,CatchCrop,CatchCrop_1,2019-10-15,NA
31.0000003587143.001,236,2019-05-20,2019-06-10,2019-10-15,CatchCrop,CatchCrop_3,2019-10-15,NA
```

Appendix E.15 Processors execution from docker or locally

By default, processors are configured to be executed from a docker container. In this moment, there are still some scripts from the processors that need to be executed from the local system but this might change in the future.

The following keys are involved in the configuration of the execution of the processors in a docker:

Key	Description
general.orchestrator.use_docker	Enables or disables globally the usage of the docker container for the processors steps execution. Default : 1
general.orchestrator.docker_image	The global docker image name for the processors step execution, in case docker is activated (either globally, at processor level or at step level). Default: "lnicola/sen2agri-processors"
general.orchestrator.docker_add_mounts	Additional mounts to be added in particular cases when users are using custom paths from where products are, for example, imported in the system with the "Direct link to product" option. In case a value like the following should be added in this key: /eodata/Sentinel-2/MSI/L2A:/eodata/Sentinel-2/MSI/L2A

	If the ESA Sen2Cor L2A products are directly imported from a DIAS
	Or :
	/my/l2_custom/dir/:/my/l2_custom/dir/
	If products with paths in this custom directory was imported in database.
	Multiple directories can be configured, separated with comma:
	/dir1:/dir1/,/dir2:/dir2/,/dir3:/dir3/

All the keys in the above table, can be also overwritten at processor and step level like, for example:

general.orchestrator.s4c_l4b.use_docker = 0

⇨ This means that all the steps executed in the L4B processor will be executed locally.

general.orchestrator.s4c-grassland-gen-input-shp.use_docker = 0

⇨ This means that only the s4c-grassland-gen-input-shp step of the L4B processor will be executed locally but not the others.

general.orchestrator.s4c-grassland-gen-input-shp.docker_image = other/processors-image2

⇨ This means that if the step s4c-grassland-gen-input-shp is configured (explicitly or implicitly) to be executed using docker, it will be executed in this particular docker container.

The inheritance of the settings is the following:

- If at step level there is no configuration, the configuration from the processor (if any) or at global level is inherited.
- If at step level there is a configuration, it will be used, regardless the configuration at the higher levels.
- If at step level there is no configuration, the configuration at global level is inherited (if no particular configuration at step level).
- If at processor level there is a configuration, it will be used, regardless the configuration at the global level (if no particular configuration at step level).
- docker_add_mounts configurations are accumulated for global, processor and step.

Appendix F – Endpoint Services description

Appendix F.1 Data source configuration

GET: /downloader/sources/

This method allows the user to retrieve the list of predefined configurations of data sources modules.

Note: There are no mandatory or optional parameters for this function.

Example: /downloader/sources/

GET: /downloader/sources/{satellite}

This method allows the user to retrieve the list of predefined configurations of data sources modules for the given satellite (*satellite*).

Parameter	Description
satellite: *short*	The unique identifier of a sentinel; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);

Example: /downloader/sources/1

GET: /downloader/sources/{satellite}/{name}

This method allows the user to retrieve the configuration of the given data source (*name*) for the given satellite (*satellite*).

Parameter	Description
satellite: *short*	The unique identifier of a sentinel; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);
name: *String*	The name of the data source from which the user wants to download data.

Example: /downloader/sources/1/SciHub

POST: /downloader/sources/{satellite}/{name}

This method allows the user to update a certain configuration of the given data source (*name*) for the given satellite (*satellite*).

Parameter	Description
satellite: *short*	The unique identifier of a sentinel; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);
name: *String*	The name of the data source from which the user wants to updata data.

Example: /downloader/sources/1/SciHub

GET: /refresh/

This method allows the user to refresh the configuration.

Note: There are no mandatory or optional parameters for this function.

Example: /refresh/

GET: /refresh/info

This method allows the user to view informations regarding the memory, threads and jobs.

Note: There are no mandatory or optional parameters for this function.

Example: /refresh/info

GET: /parcel

This method allows the user to retrieve information about specific parcel. This method performs a verification of all the input parameters, in order to provide the correct information.

Parameter	Description
site: *String*	The name of the site for which we want to retrieve information about the parcel.
id: *String*	The id of the parcel for which we want to retrieve information.
practice: *Practice*	
year: *int*	

Example: /parcel

Appendix F.2 Download control

GET: /downloader/

This method returns information about all the downloads in progress.

Note: There are no mandatory or optional parameters for this function.

Example: /downloader/

GET: /downloader/{id}

This method returns information about the downloads in progress for a specific site (*id*).

Parameter	Description
id: *short*	The unique identifier of a site; if the value of the parameter is: • id = 0: it returns the list with all the downloads in progress; • id != 0: it returns the list with all the downloads in progress for the given site;

Example: /downloader/1235

GET: /downloader/{code}/{satellite}

This method returns information about the download in progress for a specific site (*id*) and a specific satellite (*satellite*).

Parameter	Description
code: *String*	The unique identifier of a site; if the value of the parameter is: • id = 0: it returns the list with all the downloads in progress; • id != 0: it returns the list with all the downloads in progress for the given site;
satellite: *String*	The name of the sentinel from which the user wants to retrieve the information regarding the status of the download in progress.

Example: /downloader/netherlands/S2

GET: /downloader/{id}/count

This method retrieves information about the estimated number of products to download for a specific site (*id*)

Parameter	Description
id: *short*	The unique identifier of the site for which the user wants to retrieve the information regarding the estimated number of products to download.

Example: /downloader/1235/count

GET: /downloader/stop

This method stops all the downloads and marks the downloader as disabled for all sites.

Note: There are no mandatory or optional parameters for this function.

Example: /downloader/stop

GET: /downloader/stop/{id}

This method stops all the downloads and marks the downloader as disabled for a specific site (*id*).

Parameter	Description
id: *short*	The unique identifier of the site for which the user wants to stop all the downloads; if the value of the parameter is: • id = 0: it stops all the downloads in progress; • id != 0: it stops all the downloads in progress for the given site;

Example: /downloader/stop/1235

GET: /downloader/stop/{id}/{satelliteId}

This method stops all the downloads and marks the downloader as disabled for the specific site (*id*) and specific sensor (*satelliteId*).

Parameter	Description
id: *short*	The unique identifier of the site for which the user wants to stop all the downloads for the given sensor (*satelliteId*);
satelliteId: *short*	The unique identifier of the sensor from which the user stops all the downloads, for the given site.

Example: /downloader/stop/1235/2

GET: /downloader/start

This method starts all the downloads and marks the downloader as enabled for all sites.

Note: There are no mandatory or optional parameters for this function.

Example: /downloader/start

GET: /downloader/start/{id}

This method starts all the downloads and marks the downloader as enabled for a specific site (*id*).

Parameter	Description
id: *short*	The unique identifier of the site for which the user wants to enable all the downloads; if the value of the parameter is: • id = 0: it enables all the downloads in progress; • id != 0: it enables all the downloads in progress for the given site;

Example: /downloader/start/1235

GET: /downloader/start/{id}/{satelliteId}

This method starts all the downloads and marks the downloader as enabled for a specific site (*id*) and a specific satellite (*satelliteId*).

Parameter	Description
id: *short*	The unique identifier of the site for which the user wants to start all the downloads, for the given sensor (*satelliteId*);
satelliteId: *short*	The unique identifier of the sensor from which the user enables all the downloads, for the given site (*id*).

Example: /downloader/start/1235/3

GET: /downloader/forcestart

This method forces the downloader for the specific job (*job*), site (*siteId*) and sensor (*satelliteId*) to start from the beginning of the first defined season.

Parameter	Description
job: *String*	The unique identifier of the job type that the user wants to force to restart;
siteId: *short*	The unique identifier of a site; if the value of the parameter is: • siteId = 0: it forces the downloader to start from the beginning for all the sites; • siteId != 0: it forces the downloader to start from the beginning for the given site;
satelliteId: *short*	The unique identifier of the sensor from which the user forces the restart of the downloader;

Example: /downloader/forcestart

Appendix F.3 Import

GET: /products/import/l1

This method allows the user to import data into the system.

Parameter	Description
siteId: *short*	The name of the site for which we want to import L1 products into the system.
folder: *String*	The relative folder where we want to upload L1 products.
satelliteId: *short*	The name of the site for which we want to upload L1 products.
link: *boolean*	If: • true: is linking folder to target path; • false: is copying the folder to target path;

Example: /products/import/l1

GET: /import/lpis

This method allows the user to import the subsidy application dataset into the system. At least one subsidy application dataset shapefile or one LUT (Look-Up-Table) file must be present.

Parameter	Description
siteId: *short*	The name of the site for which we want to import the subsidy application dataset.
parcelColumns: *String[]*	The parcel division of the site.
holdingColumns: *String[]*	The site holding.

cropCodeColumn: *String*	The crop code for each parcel.
year: *Optional* *<Integer>*	
lpisFile: *Optional* *<String>*	
lutFile: *Optional* *<String>*	
mode: *Optional* *<String>*	

Example: /import/lpis

GET: /import/{siteShortName}

This method allows the user to check if a certain site exists into the system and, if so, update its data. This method performs a file size check in order validate the verification.

Parameter	Description
siteShortName: *String*	The name of the site for which we want update data.
file: *String*	The file with new data, which we want to add to the system.
type: *DataType*	The file type (*LPIS* or *DECLARATION*).
offset: *Optional* *<Long>*	

Example: /import/ netherlands_1

GET: /import/{siteShortName}/{year}

This method allows the user to check if a certain site exists into the system and, if so, update its data. This method performs a file size check in order validate the verification.

Parameter	Description
siteShortName: *String*	
year: *int*	
file: *String*	
type: *Optional* *<DataType>*	

crs: *Optional* *<String>*	
offset: *Optional* *<Long>*	

Example: /import/ netherlands_1/2018

GET: /s1l2/config

This method allows the user to retrieve the actual configuration values for the Sentinel-1 L2 processor.

Note: There are no mandatory or optional parameters for this function.

Example: /s1l2/ config

GET: /s1l2/

This method allows the user to retrieve the actual metadata of the product specified in the given path.

Parameter	Description
path: *String*	The path of the product for which the user wants to retrieve the metadata.

Example: /s1l2/

GET: /s1l2/reprocess

This method allows the user to trigger the reprocessing of the Sentinel-1 L1 SLC acquisition, for a given site and for a given polarisation.

Parameter	Description
siteCode: *String*	The path of the product for which the user wants to retrieve the metadata.
product: *String*	The name of the acquisition.
polarisation: *Optional <String>*	The name of the polarisation for which the user wants to do the reprocessing. By default: VV and VH.

Example: /s1l2/

GET: /s1l2/import/l2/{site}

This method allows the user to import in the system database the L2 products from a specific folder.

Parameter	Description
site: *String*	The site name for which we trigger the import.
folder: *String*	The path of the folder from which we want to import data.

Example: /s1l2/import/l2/ netherlands_1

Appendix F.4 Output products

GET: /sites/

This method allows the user to retrieve the list of all existing sites.

Note: There are no mandatory or optional parameters for this function.

Example: /sites/

POST: /sites/

This method allows the user to create a new site; it returns the unique identifier of the created site (*siteId*).

Parameter	Description
name: *String*	The name of the site we want to create.
zipFilePath: *String*	The relative path of the site shapefile.
enabled: *boolean*	Provides information concerning the site status (enabled/disabled).

Example: /sites/

DELETE: /sites/

This method allows the user to delete an existing site and all related products.

Parameter	Description
siteId: *short*	The unique identifier of the site we want to delete.

Example: /sites/

GET: /sites/seasons/{id}

This method allows the user to retrieve the list of all seasons for a specific site (*id*).

Parameter	Description
siteId: *String*	The unique identifier of the site for which we want to retrieve of all seasons.

Example: /sites/seasons/15

POST: /login

This method allows the user to connect to the application using a valid username and password.

Parameter	Description
user: *String*	The unique identifier of the name/entity/person who wants to use the application.

pwd: *String*	The unique identifier that allows the user (*user*) to access the application.

Example: /login

GET: /products/

This method allows the user to retrieve the list of all the existing products.

Parameter	Description
userId: *int*	The unique identifier of the logged user.
siteId: *int*	The unique identifier of the site for which we want to retrieve the products.
productTypeId: *int*	The unique identifier of the product types we want to retrieve.
token: *String*	The unique identifier that authenticates the users who attempt to log into the application.

Example: /products/

GET: /products/disable/{satellite}/{id}

This method allows the user to disable the downloading or processing products for a specific satellite (*satellite*) and site (*id*).

Parameter	Description
satellite: *short*	The enum value of sensor we want to disable; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);
id: *short*	The unique identifier of the site we want to disable; if the value of the parameter is: • id = 0: it disables the downloads or processing in progress for all sites; • id != 0: it disables the downloads or processing in progress for the given site;

Example: /products/disable/3/1235

GET: /products/enable/{satellite}/{id}

This method allows the user to enable the downloading or processing products for a specific satellite (*satellite*) and site (*id*).

Parameter	Description
satellite: *short*	The enum value of sensor we want to enable; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);

id: *short*	The unique identifier of the site we want to enable; if the value of the parameter is: • id = 0: it enables the downloads or processing in progress for all sites; • id != 0: it enables the downloads or processing in progress for the given site;

Example: /products/enable/3/1235

GET: /products/enable/status/{satellite}/{id}

This method allows the user to view the status (enable/disable) of a specific satellite (*satellite*) and site (*id*).

Parameter	Description
satellite: *short*	The enum value of sensor for which we want to check the status; possible values: • 1 = S2 (Sentinel 2); • 2 = L8 (Landsat8); • 3 = S1 (Sentinel 1);
id: *short*	The unique identifier of the site for which we want to check the status.

Example: /products/enable/status/3/1235

GET: /products/objectstorage/get

? This method returns information about the product types for all stored objects

Note: There are no mandatory or optional parameters for this function.

Example: /products/objectstorage/get

GET: /products/objectstorage/set/{productsTypeIds}

This method returns information about a specific product type.

Parameter	Description
productsTypeIds: *String[]*	The unique identifier of the site products types we want to retrieve.

Example: /products/objectstorage/get/12;15

GET: /products/types

This method returns a list with all the existing product types

Note: There are no mandatory or optional parameters for this function.

Example: /products/types

GET: /products/download

	Ref	Sen4CAP_SUM_v2.0		
eesa	Issue	Page	Date	sen4cap
	2.0	147	07/01/2021	common agricultural policy

This method allows the user to download a specific product (*name*) for a specific site (*siteId*), in ".zip" format.

Parameter	Description
name: *String*	The name of the product we want to download.
siteId: *int*	The unique identifier of the site for which we want to download the given product.

Example: /products/download

POST: /products/upload

This method allows the user to upload files into the system.

Parameter	Description
siteId: *int*	The name of the site for which we want to upload a given file.
file: *MultipartFile*	The file we want to upload.
folder: *String*	The relative folder we want to upload the file.

Example: /products/upload

GET: /objectstorage/productTypes

This method allows the user to retrieve the product type ids that should be moved to Object Storage buckets.

Note: There are no mandatory or optional parameters for this function.

Example: /objectstorage/productTypes

POST: /objectstorage/productTypes

This method allows the user to set the product type ids that should be moved to Object Storage buckets.

Note: There are no mandatory or optional parameters for this function.

Example: /objectstorage/productTypes

GET: /objectstorage/delete

This method allows the user to retrieve the number of deleted objects.

Parameter	Description
container: *String*	The Object Storage bucket.

filter: *String*	The name of the applied filter.

Example: /objectstorage/delete

GET: /objectstorage/copy

This method allows the user to copy the given file, associated to the given site and representing the given product type, to Object Storage.

Parameter	Description
siteId: *short*	The unique identifier of the site we want to copy.
productType: *ProductType*	The product type identifier.
year: *int*	The year of the product.
file: *String*	The file representing the product.

Example: /objectstorage/copy

GET: /objectstorage/move

This method allows the user to move the given file, associated to the given site and representing the given product type, to Object Storage.

Parameter	Description
siteId: *short*	The unique identifier of the site we want to move.
productType: *ProductType*	The product type identifier.
year: *int*	The year of the product.
file: *String*	The file representing the product.

Example: /objectstorage/copy

Appendix F.5 Marker Database

GET: /markers/names

This method will allow the user to retrieve the marker names of a given marker type, available for a site and a year.

Parameter	Description
site: *String*	The site short name as it appears in the sites visualization tab.
productType: *ProductType*	The markers product type identifier. Available marker product types: s4c_mdb1 – simple standard deviation and mean values for AMP, COHE, NDVI, LAI, FAPAR, FCOVER s4c_mdb2 – simple standard deviation and mean values for Ampl Ratio VH/VV

	s4c_mdb3 – L4C M1-M5 markers
	s4c_mdb_l4a_opt_main - L4A Optical Main Features
	s4c_mdb_l4a_opt_re - L4A Optical Red-Edge Features
	s4c_mdb_l4a_sar_main - L4A SAR Main Features
	s4c_mdb_l4a_sar_temp - L4A SAR Temporal Features
year: *int*	The year of the product.

Example: /markers/names?site=test_site&productType=s4c_mdb2&year=2020

GET: /markers

This method will allow the user to retrieve the values of the selected markers, available for a product type, a site and a year.

Parameter	Description
site: *String*	The site short name as it appears in the sites visualization tab.
productType: *ProductType*	The markers product type identifier.
year: *int*	The year of the product.
markers: *String*	Comma separated list of markers names obtained with a /marker/names query.
from: *String*	Filtering start date for markers values retrieval
to: *String*	Filtering end date for markers values retrieval
roi: *String*	ROI as WKT for filtering parcels
parcels: *String*	Comma separated string for filtering parcels

Example:
/markers?site=test_http&productType=s4c_mdb2&year=2020&markers=mean_AMP_VVVH_022, stdev_AMP_VVVH_022

A special case of the markers is the L4C M1-M5 markers values that normally have the values TRUE, FALSE, NA, NA1 or NR (please see the Agricultural Practices ATBD for the meaning of these values). These values are translated in the marker returned values using the following mapping to integer values:

L4C Value	Marker Value
TRUE	1
FALSE	0
NA	-1
NA1	-2
NR	-3