
# D4.1 - Demonstration of data collection, aggregation of Earth Observations, weather/climate models and in-situ environmental sensors for forest fire risk/threat assessment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 101037247


| | |
|---|---|
| **Project Acronym** | SILVANUS |
| **Grant Agreement number** | 101037247 (H2020-LC-GD-2020-3) |
| **Project Full Title** | Integrated Technological and Information Platform for Wildfire Management |
| **Funding Scheme** | IA – Innovation action |

**DELIVERABLE INFORMATION**

| | |
|---|---|
| **Deliverable Number:** | D4.1 |
| **Deliverable Name:** | Demonstration of data collection, aggregation of Earth Observations, weather/climate models and in-situ environmental sensors for forest fire danger |
| **Dissemination level:** | Public |
| **Type of Document:** | Demonstrator |
| **Contractual date of delivery:** | 31/03/2023 (M18) |
| **Date of submission:** | 31/03/2023 |
| **Deliverable Leader:** | CMCC |
| **Status:** | Version with reviewers' comment |
| **Version number:** | 1.0 |
| **WPLeader/ TaskLeader:** | CMCC/DELL/CTL |
| **Keywords** | Earth Observations; Weather/Climate Forecasting; Internet of Things |
| **Abstract** | This document reports the implementation and demonstration of the systems (interfaces, methods, and tools) to collect and aggregate data from heterogenous sources (Earth Observations, Weather and Climate Modes, in-situ devices) and subsequent pre-processing capabilities to detect fire and to quantify the forest fire danger. It also outlines the integration of those systems with the SILVANUS Platform. |

| | |
|---|---|
| **Deliverable Leader:** | CMCC |
| **Lead Author(s)** | Marco Mancini |
| **Reviewers** | EXUS: Aris Bonanos, George Diles<br>RINI: G.Markarian, J.Levak<br>WUT: Wojciech Mazurczyk, Krzysztof Cabaj, Natan Orzechowski, |

| | |
|---|---|
| | Przemysław Szary |

Disclaimer

All information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose.

The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors ‘view.

| Document History | | | |
|---|---|---|---|
| **Version** | **Date** | **Contributor(s)** | **Description** |
| V0.1 | 23.01.2023 | Marco Mancini (CMCC) | Proposal of ToC |
| V0.2 | 21.03.2023 | All contributors | Ready for the review |
| V0.3 | 30.03.2023 | All contributors | Version with reviewers' comment |
| V1.0 | 31.03.2023 | All contributors | Ready for submission |

**List of Contributors**

| Partner | Author(s) |
|---|---|
| CERTH | Yiannis Kouloglou |
| CMCC | Marco Mancini, Giusy Fedele, Ilenia Manco, Giuliana Barbato, Giuseppe Giugliano, Paola Mercogliano, Pasquale Schiano, Carmen Alvarez Castro, Malik Aljabu |
| DELL | Matthew Keating |
| CTL | Maria Maslioukova, Georgios Prokopiou, Konstantinos Avgerinakis |
| TP | Ivo Gama, Jorge Palma |
| FINC | Marco Saltarella, Antonio Santovito, Marcello Paolo Scipioni |
| EAI | Maria Serafina Cefarelli |
| EXUS | Aris Bonanos |

**List of beneficiaries**

| No | Partner Name | Short name | Country |
|---|---|---|---|
| 1 | UNIVERSITA TELEMATICA PEGASO | PEGASO | Italy |
| 2 | ZANASI ALESSANDRO SRL | Z&P | Italy |
| 3 | NETCOMPANY-INTRASOFT SA | INTRA | Luxembourg |
| 4 | THALES | TRT | France |
| 5 | FINCONS SPA | FINC | Italy |
| 6 | ATOS IT SOLUTIONS AND SERVICES IBERIA SL | ATOS IT | Spain |
| 6.1 | ATOS SPAIN SA | ATOS SA | Spain |
| 7 | EMC INFORMATION SYSTEMS INTERNATIONAL | DELL | Ireland |
| 8 | SOFTWARE IMAGINATION & VISION SRL | SIMAVI | Romania |
| 9 | CNET CENTRE FOR NEW ENERGY TECHNOLOGIES SA | EDP | Portugal |
| 10 | ADP VALOR SERVICOS AMBIENTAIS SA | ADP | Portugal |
| 11 | TERRAPRIMA - SERVICOS AMBIENTAIS SOCIEDADE UNIPESSOAL LDA | TP | Portugal |
| 12 | 3MON, s. r. o. | 3MON | Slovakia |
| 13 | CATALINK LIMITED | CTL | Cyprus |
| 14 | SYNTHESIS CENTER FOR RESEARCH AND EDUCATION LIMITED | SYNC | Cyprus |
| 15 | EXPERT SYSTEM SPA | EAI | Italy |
| 16 | ITTI SP ZOO | ITTI | Poland |
| 17 | Venaka Treleaf GbR | VTG | Germany |
| 18 | MASSIVE DYNAMIC SWEDEN AB | MDS | Sweden |
| 19 | FONDAZIONE CENTRO EURO-MEDITERRANEOSUI CAMBIAMENTI CLIMATICI | CMCC F | Italy |
| 20 | EXUS SOFTWARE MONOPROSOPI ETAIRIA PERIORISMENIS EVTHINIS | EXUS | Greece |
| 21 | RINIGARD DOO ZA USLUGE | RINI | Croatia |
| 22 | Micro Digital d.o.o. | MD | Croatia |
| 23 | POLITECHNIKA WARSZAWSKA | WUT | Poland |
| 24 | HOEGSKOLAN I BORAS | HB | Sweden |
| 25 | GEOPONIKO PANEPISTIMION ATHINON | AUA | Greece |
| 26 | ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS | CERTH | Greece |
| 27 | PANEPISTIMIO THESSALIAS | UTH | Greece |

| No | Partner Name | Short name | Country |
|---|---|---|---|
| 28 | ASSOCIACAO DO INSTITUTO SUPERIOR TECNICO PARA A INVESTIGACAO E DESENVOLVIMENTO | IST | Portugal |
| 29 | VELEUCILISTE VELIKA GORICA | UASVG | Croatia |
| 30 | USTAV INFORMATIKY, SLOVENSKA AKADEMIA VIED | UISAV | Slovakia |
| 31 | POMPIERS DE L'URGENCE INTERNATIONALE | PUI | France |
| 32 | THE MAIN SCHOOL OF FIRE SERVICE | SGPS | Poland |
| 33 | ASSET - Agenzia regionale Strategica per lo Sviluppo Ecosostenibile del Territorio | ASSET | Italy |
| 34 | LETS ITALIA srls | LETS | Italy |
| 35 | Parco Naturale Regionale di Tepilora | PNRT | Italy |
| 36 | FUNDATIA PENTRU SMURD | SMURD | Romania |
| 37 | Romanian Forestry Association - ASFOR | ASFOR | Romania |
| 38 | KENTRO MELETON ASFALEIAS | KEMEA | Greece |
| 39 | ELLINIKI OMADA DIASOSIS SOMATEIO | HRT | Greece |
| 40 | ARISTOTELIO PANEPISTIMIO THESSALONIKIS | AHEPA | Greece |
| 41 | Ospedale Israelitico | OIR | Italy |
| 42 | PERIFEREIA STEREAS ELLADAS | PSTE | Greece |
| 43 | HASICSKY ZACHRANNY SBOR MORAVSKOSLEZSKEHO KRAJE | FRB MSR | Czechia |
| 44 | Hrvatska vatrogasna zajednica | HVZ | Croatia |
| 45 | TECHNICKA UNIVERZITA VO ZVOLENE | TUZVO | Slovakia |
| 46 | Obcianske zdruzenie Plamen Badin | PLAMEN | Slovakia |
| 47 | Yayasan AMIKOM Yogyakarta | AMIKOM | Indonesia |
| 48 | COMMONWEALTH SCIENTIFIC AND INDUSTRIAL RESEARCH ORGANISATION | CSIRO | Australia |
| 50 | FUNDACAO COORDENACAO DE PROJETOS PESQUISAS E ESTUDOS TECNOLOGICOS COPPETEC | COPPETEC | Brazil |

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

**LIST OF ACRONYMS**

| ACRONYM | DESCRIPTION |
|---|---|
| AOI | Area of Interest |
| CLC | Corine Land Cover |
| CSV | Comma Separated Values |
| DEM | Digital Elevation Model |
| DDS | Data Delivery System |
| EEA | European Environment Agency |
| EVI | Enhanced Vegetation Index |
| EO | Earth Observation |
| FWI | Fire Weather Index |
| GUI | Graphic User Interface |
| HDF | Hierarchical Data Format |
| IoTED | Internet of Things Edge Device |
| IoT | Internet of Things |
| JSON | JavaScript Object Notation |
| LST | Land Surface Temperature |
| LoRa | Long Range |
| MMU | Minimum Mapping Unit |
| MSG | Meteosat Second Generation |
| NetCDF | Network Common Data Form |
| NCAR | National Center for Atmospheric Research |
| NDMI | Normalized Difference Moisture Index |
| NDVI | Normalized Difference Vegetation Index |
| NDWI | Normalized Difference Water Index |
| NIR | Near Infrared |
| NWP | Numerical Weather Prediction |
| OSAVI | Optimized Soil Adjusted Vegetation Index |
| OSM | Open Street Map |
| PBF | *Protocol buffer binary format* |
| PII | Personally Identifiable Information |
| QF | Quinta da França farm |
| SAL | Storage Abstraction Layer |
| S3 | (Amazon) Simple Storage Service |
| SAVI | Soil Adjusted Vegetation Index |
| TCD | Tree Cover Density |
| TIFF | Tagged image file format |
| URI | Uniform Resource Identifier |
| UN | United Nations |
| UIF | User Interface Frontend |
| WRF | Weather Research and Forecasting Model |

**EXECUTIVE SUMMARY**

This document reports the implementation and demonstration of the systems - interfaces, methods, and tools - to collect and aggregate data from heterogenous sources (Earth Observations, Weather and Climate Modes, in-situ devices) and subsequent pre-processing capabilities to detect fire and to quantify the forest fire danger. It also outlines the integration of those systems with the SILVANUS Platform in order to provide data to the SILVANUS downstream application and services.

The datasets collected and produced in SILVANUS are related to heterogeneous sources providing different data formats and different ways to access data. It is important to define ingestion procedures that allow services and applications to easily retrieve data necessary for analysis and processing to detect fire and quantify the fire risk.

This document reports the ingestion procedures, the pre-processing and post-processing steps for each dataset that has been identified in Deliverable D8.1 related to the SILVANUS Platform Architecture in order to make them available to the SILVANUS downstream User Products (as reported in D8.1). In particular, the ingestion pipelines reported in this document are related to the following datasets:

***Satellite Data***: European Copernicus Programme and EUMETSAT Data Centre provide a vast amount of Earth Data – satellite and in-situ observations - that will be used by SILVANUS User Products. Through the Copernicus Open Access Hub and EUMETSAT Platform different Satellite data products will be ingested and pre-processed in the SILVANUS Platform. Satellite images will be used for computing derived indices related to the earth's surface and for the nowcasting (0-3 hours) of fires. In SILVANUS, different satellite derived products (e.g. NDVI, Active Fire Monitoring) will be ingested from EUMETSAT and Copernicus repositories or will be computed starting from Sentinel-2 and EUMETSAT data products.

***Topographic, Road and Population data***: In SILVANUS, different User Products needs to access to static reference data such as Digital Elevation Model, road density and population density data. This data will be ingested in the SILVANUS Platform from different repositories such as Copernicus Land Monitoring Service for DEM, OpenStreetMap for road density and Worldpop for population data.

***Weather and Climate data***: SILVANUS will provide high resolution, high quality and reliable data from weather forecast models in the short-term range (up to 72 hours), also using statistical analysis tools to remove biases and to perform downscaling. Short-term weather prediction will be used to provide fire weather indices). SILVANUS will produce also probabilistic fire weather indices based on the ensemble seasonal forecasts produced by Copernicus C3S. Those data will be provided by CMCC Data Delivery System and will be ingested in the SILVANUS Platform to feed downstream User Products such as the Fire Spread and Fire Danger Forecast.

***Real-time IoT data***: SILVANUS will collect, pre-process and aggregate IoT In-situ data which are generated by different types of devices installed in the different pilots, such as cameras, environmental sensors like soil and/or leaf humidity, biosensors and others.

This document provides also extensive description on the pre-processing and post-processing steps of observational– satellite and in-situ - and modelling data to make them available to the SILVANUS Downstream User Products.

# 1. Introduction

This document provides a description of the implementation and demonstration of the SILVANUS systems (interfaces, methods, and tools) to collect and aggregate data from heterogenous sources (i.e., Earth Observations, Weather and Climate Models, In-situ devices) and associated pre-processing functionalities

- to detect fires
- to quantify the forest fire danger.

It also outlines the integration of those systems within the SILVANUS Platform as reported in Figure 1.

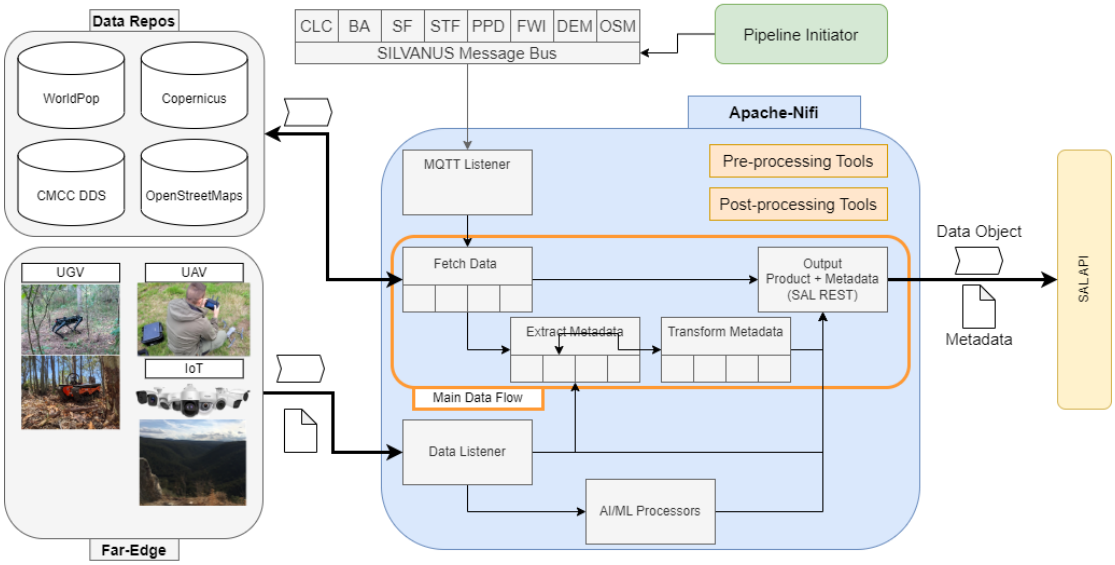



**Figure 1 – Ingestion Data flow from source to SILVANUS Storage Abstraction Layer**

The SILVANUS system for data collection, aggregation and pre-processing uses Apache Nifi [1] as a common pipeline framework to ingest, transform, and store data into the SILVANUS Storage Abstraction Layer (SAL) (an S3-like storage solution). In the Nifi framework, each ingestion flow is composed of different blocks each representing a Processor, which carries out an individual unit of work or function. Each Processor is characterized by an input, a function, and an output step; each of these steps operates on FlowFiles, where a Flowfile refer to the Nifi's representation of a main data object body (e.g. JSON, png etc.) and attached metadata attributes. By chaining these blocks together, it is possible to perform relevantly complex tasks. Leveraging the Nifi framework, we have been able to work efficiently in parallel in a production ready rapid development environment. The ingestion process is initiated by publishing a message on the SILVANUS message bus based on RabbitMQ.

The document is structured as follows:

*Section 2: Earth Observations* is related to the used Earth Observations datasets (i.e., satellite images and/or derived products) and their ingestion process, including the pre-processing and post-processing tools.

*Section 3: Weather and Climate data* is related to the weather and seasonal forecast models that have been developed to provide forecast about the fire weather index.

*Section 4: In-situ Devices* describes the IoT devices deployed in the different SILVANUS pilot sites, their ingestion process and how they are used for fire detection.

*Section 5: Integration with SILVANUS Platform* describes in details the integration between the pipelines developed for the ingestion of the data sources described in the previous sections and the SILVANUS Platform.

*Section 6: Conclusions* provides the overall conclusions of the deliverable

# 2. Earth Observations

## 2.1. Repositories & Ingestion Flows

This section is related to the different Earth Observation datasets that have been processed and ingested in the SILVANUS platform. In the following paragraphs, EO datasets and their ingestion process are described.

### 2.1.1. Digital Elevation Model

The Digital Elevation Model (DEM) [2] is a static dataset provided by Copernicus Land Monitoring Service, describing the height/ elevation for a region within Europe. Products are split into tiles, which are sub-sections that undergo a pre-processing step from raw satellite sensor data to a Digital Elevation Tile. Each Tile has a metadata descriptor attached, which describes the identifier, region/ spatial context, as well as other relevant indexing datapoints.

The dataset format comprises TIFF objects, a file format intended to encode both raw image data as well as geospatial and other relevant metadata. The average file size for a single DEM tile at 25m spatial resolution is ~5GB, covering a total area of 1,000x1,000km (an example is given in Figure 2). The data encoding and visualization is based on a grey-scale image, with values 0-1 representing lower to higher elevation above sea level.

Within the SILVANUS platform, this dataset is to be leveraged as a data feature for several machine learning models, such as the Fire Spread Model. This kind of model provides a prediction to users based on a number of parameters on how a wildfire is predicted to spread over time. Topographic and elevation data plays a key role due to the fact that the slope of a given physical area is crucial in determining how a fire spreads, i.e., given ideal conditions

(wind, moisture), a fire will accelerate and “prefers” to move to a higher elevation from a lower point; the slope angle of this movement effects the speed at which this effect occurs.

**Figure 2 - Copernicus DEM Tile covering northern Italy region**

The ingestion process of DEM files is initiated with a message (an example is reported in Table 1) that contains the key parameters needed to initiate the pipeline to pre-processing and output of a relevant data.

**Table 1 - DEM Sample ingestion message**

| |
|---|
| Queue: ingest.dem<br>Message: {"pilot": "gargano", "type": "dem"} |

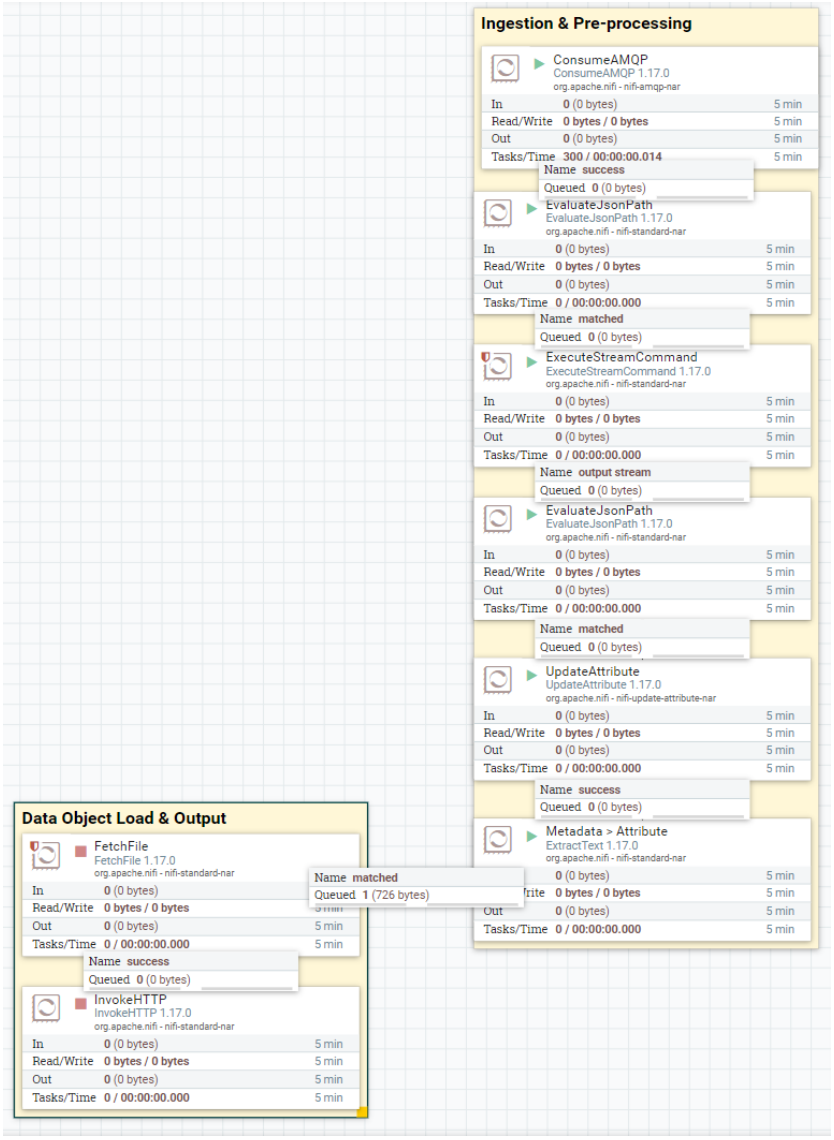



**Figure 3 - Apache Nifi Ingestion Pipeline for DEM Source**

Figure 3 shows the DEM ingestion pipeline using the Apache Nifi's GUI. The pipeline flows from top to bottom. Initiated by a new message published from the relevant ingestion queue, the body parameters are extracted via a Nifi processor before passing the contents to a custom Python-based ingestion script. This script allows greater control on how the data is pre-processed before reaching the SILVANUS SAL where the requested files related to DEM datasets are stored. One key benefit is that DEM data storage requirements can dramatically decrease, by extracting only relevant areas of interest from the larger dataset. Once this pre-processing script has been completed, the TIFF image object is temporarily stored into a local directory, and then the relevant metadata are indexed, and finally the output are stored in the SAL.

### 2.1.2. OpenStreetMap

The location and characteristics of road and railway networks have been identified in deliverable D8.1 as an input feature to be used for several SILVANUS user-products. For example, the localization of infrastructures provides a key input to ML models such as the Fire Spread Model, since the position and type of road or railway within a given spatial region can

act as an effective firebreak and thus must be considered in the calculation and in the prediction of wildfire movement.

Given this requirement, we have selected the OpenStreetMaps (OSM) platform [3], which provides open-source access to global geographic data.

OSM is a community-driven initiative that aims to create a free, editable map of the world. The map is built using data collected by volunteers through various methods, including GPS devices, aerial photography, and local knowledge. Data is uploaded to the OSM database and is freely available for anyone to use and modify.

OSM data is openly licensed, allowing developers to use it in their own applications and services without restriction. OSM data includes information on roads, buildings, parks, lakes and other geographic features, as well as attributes such as addresses, phone numbers, and opening hours. In addition to traditional map data, OSM also includes information on hiking trails, ski runs, and other recreational amenities. OSM provides raw geographic data in a variety of formats and all the feature types are tagged and can be filtered from the original raw dataset. Datasets offered directly by OSM leverage crowd-sourced data; this means that OSM XML formatted datasets contain a considerable amount of Personally Identifiable Information (PII) (Usernames, IDs) for many features which are contributed by users, such as roads and placenames. In SILVANUS, we will ingest OSM data in *protocolbuffer binary format* (*PBF*) since OSM PBF datasets clean original data by removing all fields that may contain PII, and is a compressed form of the original OSM XML data format accessible via an API [4].

The SILVANUS ingestion pipeline (as shown in Figure 4) currently implements an end-to-end data flow for ingestion of raw OSM data, including extraction of region of interest, filtering of area by feature and finally output processed data objects.

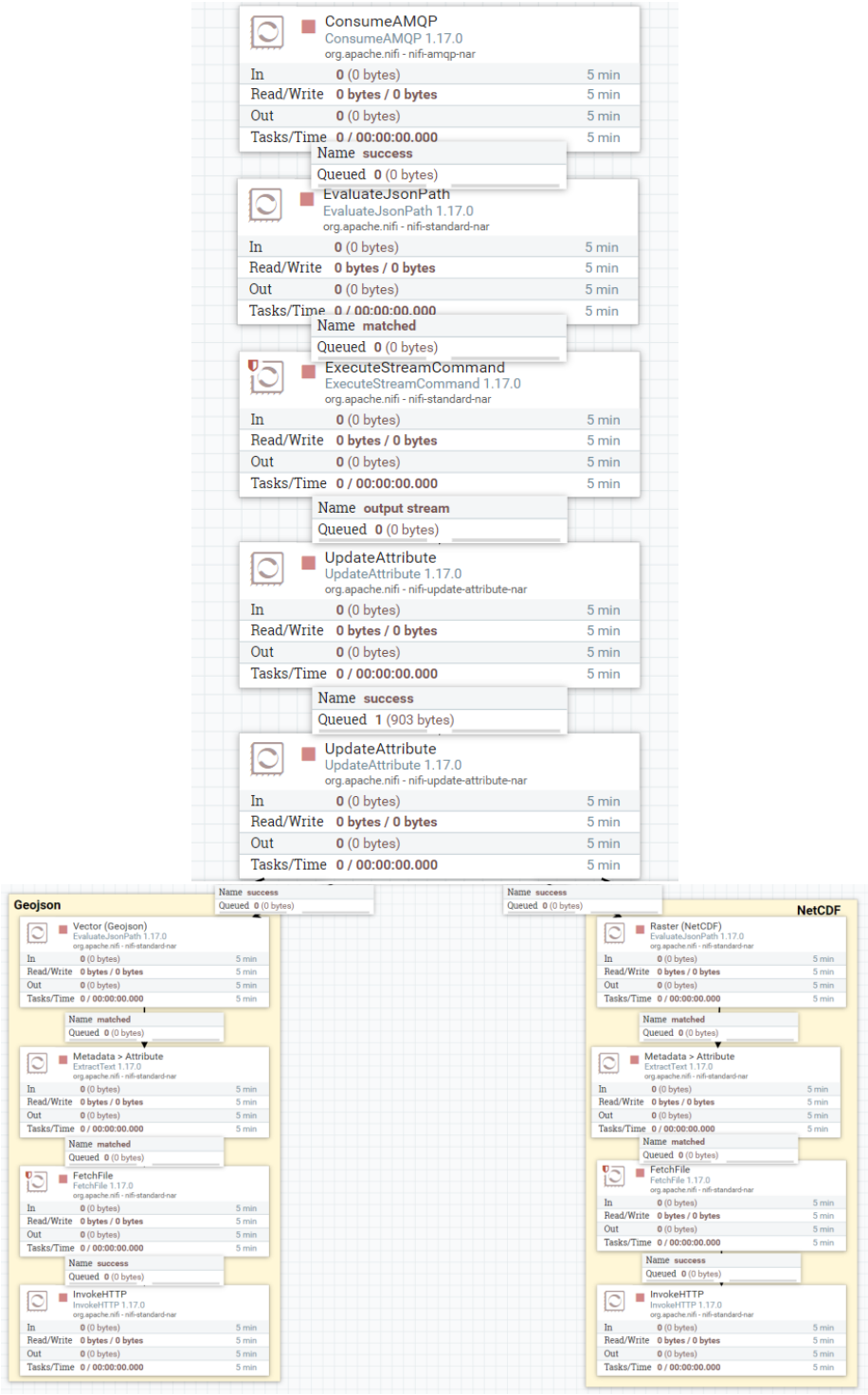



**Figure 4 - Apache Nifi Ingestion Pipeline for OSM features**

The pipeline is initiated by consuming a RabbitMQ message containing relevant parameters to retrieve and process OSM Feature data (Table 2).

**Table 2 - OSM Sample ingestion message**

| |
|---|
| Queue: ingest.osm<br>Message:<br>{ |

```
  "pilot": "gargano",
  "type": "road",
  "resolution": "100",
  "bbox": <GeoJSON Polygon>
}
```

After consuming a new message, the following parameters are extracted and passed to a custom ingestion and pre-processing script (detailed in Section 2.3.2):

- *Pilot*: relevant pilot of interest.
- *Type*: feature type to be extracted (road, rail, footprint, landcover).
- *Resolution*: output resolution (meters).
- *Bbox*: optional parameter to define region of interest (i.e. bounding box) within the pilot area.

After pre-processing script has completed, two data objects are created in the following formats:

- GeoJSON: vector format serving as input to visualization/ mapping layers (Figure 5);
- NetCDF: raster format serving as feature input to ML models (Figure 6);

and the related metadata are indexed. As shown in Figure 4, the ingestion pipeline routes these objects into two separate flows. Before storing the outputs to the SILVANUS SAL, a HTTP request is generated for each product with the object indexing metadata attached via request headers.

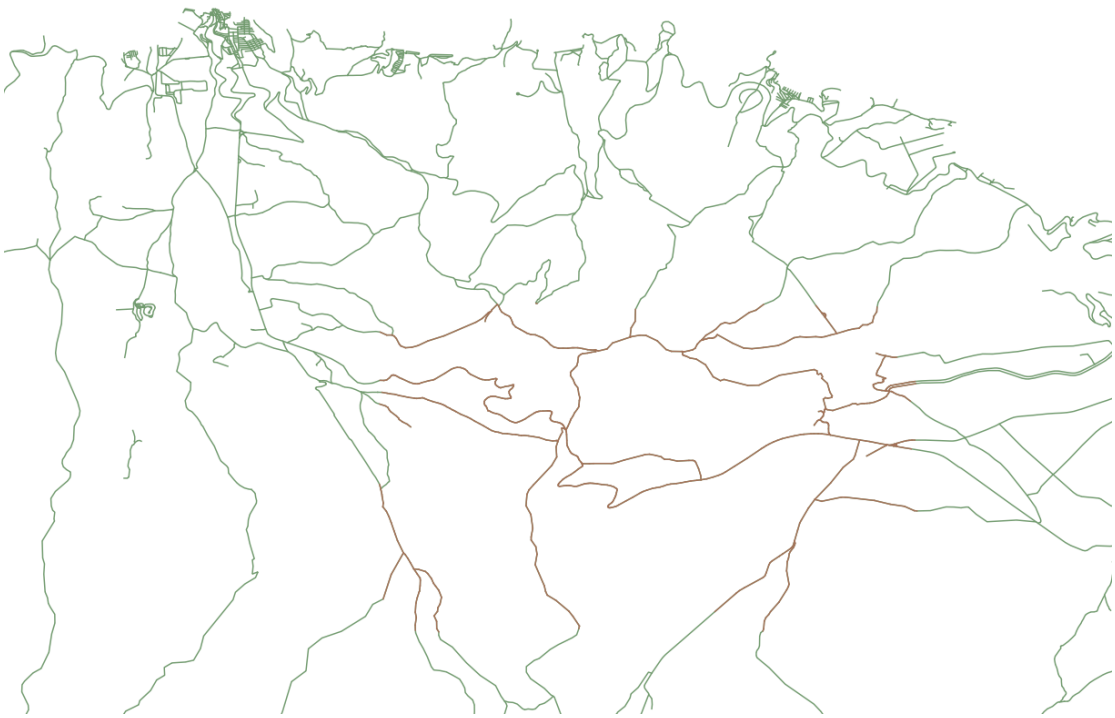


**Figure 5 - GeoJSON Road network northern Gargano region (green) and extracted ROI (brown)**

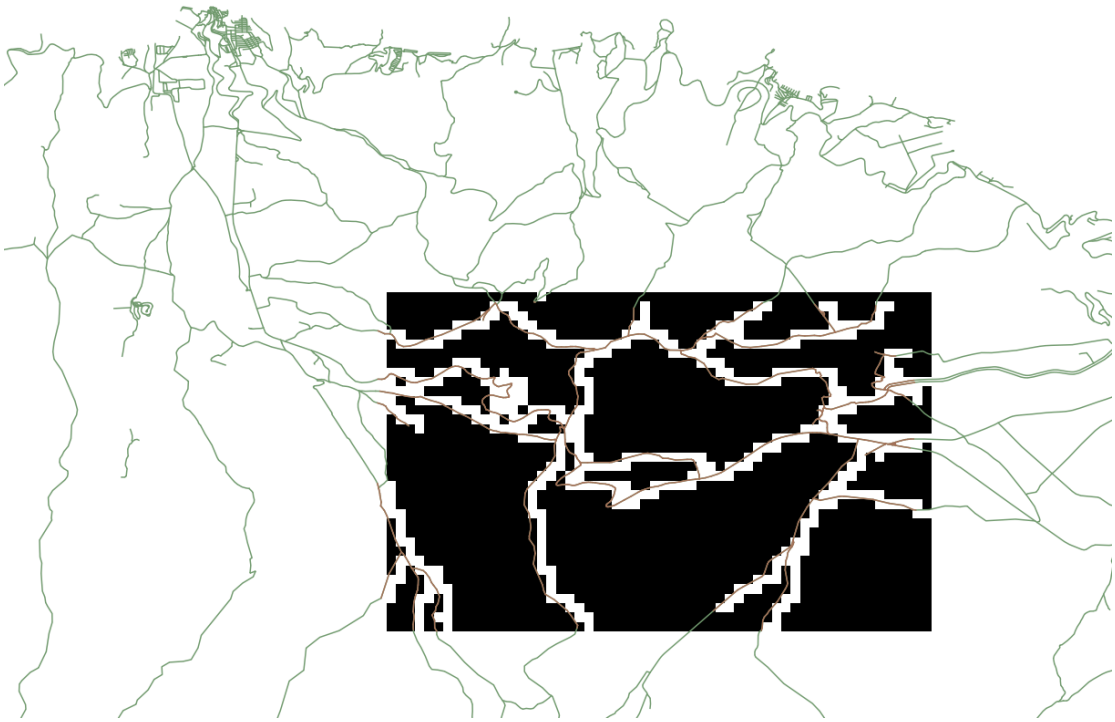


**Figure 6 - NetCDF visual overlay for extracted ROI**

### 2.1.3. Satellite Earth Observation Data

The Copernicus Programme [5] provides a vast array of Earth Observation data (satellite and in-situ observations) that can be used for a range of applications. To access this data, users can take advantage of the Copernicus Open Access Hub [6], which provides a platform for searching, discovering, and downloading the different Sentinel data products available (Sentinel-1, Sentinel-2, Sentinel-3 and Sentinel-5P). One way to interact with the Copernicus Open Access Hub within SILVANUS platform is to use a JSON input to request data (an example can be seen in the Figure 7).

```
{
    "discovery_settings": {
        "start_date":"01-01-2014" ,
        "end_date":"01-02-2022",
        "search_aoi": "POLYGON((21.95375 40.20245,21.95395 40.20245,21.95395 40.202650000000006,21.95375 40.202650000000006,21.95375 40.20245))",
        "platform": "Sentinel-2",
        "filters": ["(cloudcoverpercentage:95)"]
        }
}
```

**Figure 7 - JSON example for Copernicus Open Access Hub input within SILVANUS platform**

The JSON input file allows users to specify the search criteria for the data products they require. The following fields are included in the JSON input file to facilitate the search:

- *Start date and end date*: these fields allow the user to specify the time for which data products are needed.
- *Area of interest* (AOI): this field allows the user to define the geographical area of interest for the data search, using a polygon.

- *Platform*: this field allows the user to specify the satellite platform from which the data products are required. For example, the user may require Sentinel-2 data products.
- *Filters*: various filters are available on the Copernicus Open Access Hub Advanced Search site [7], such as cloud coverage percentage or the processing level of the data. These filters can be used to refine the search results and ensure that only the relevant data products are returned.

Once the JSON input file has been created and submitted to the SILVANUS Message Bus, the ingestion pipeline searches for and identifies the relevant data products that match the specified search criteria using the Copernicus Open Access Hub. The SILVANUS platform then generates a list of URI links to the specific data products that meet the search criteria. These URI links can be used to download the data products in various formats, such as NetCDF [8] or GeoTIFF.

### 2.1.4. Population Density

The population density data is provided by WorldPop [9] which produces different types of gridded population count datasets. Specifically, for the population density, the dataset is available in GeoTIFF and ASCII XYZ (CSV) format at a resolution of 30 arc-seconds (approximately 1km at the equator), spanning the years 2000-2020. The data are derived from the population count dataset dividing the number of people in each pixel by the pixel surface area and are produced using the unconstrained top-down modelling method [10]. The datasets are also available in the United Nation (UN) adjusted version, where the initial population count data is adjusted to match the official UN population estimates [11].

The Apache NiFI flow implemented to ingest these data in the SILVANUS platform is depicted in Figure 8.

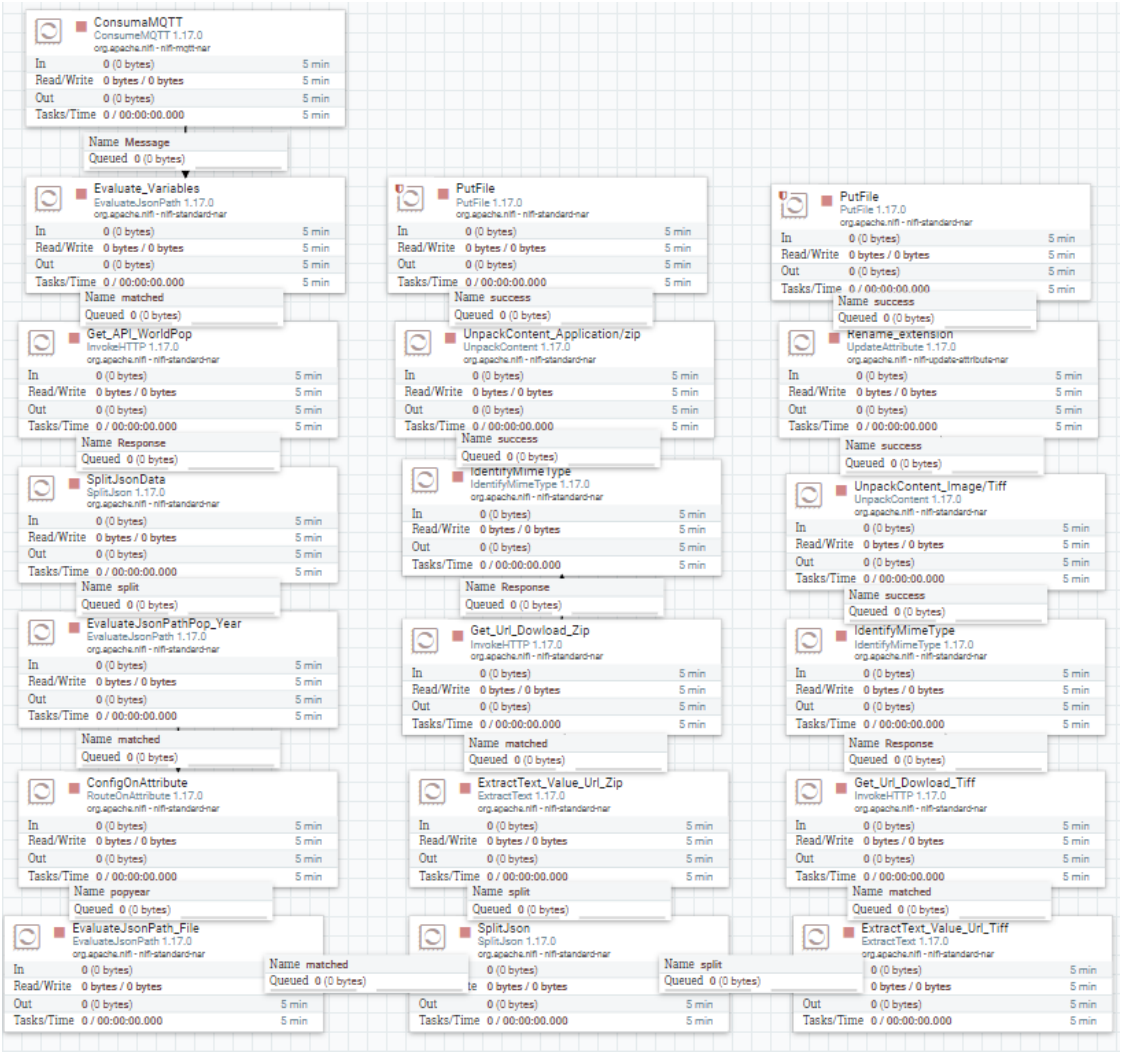



**Figure 8 – Apache NiFi ingestion pipeline for Population Density**

The flow expects in input a JSON message containing the following variables as reported in Table 3:

**Table 3 - JSON Variables for Population Density**

| Variable | Description |
|---|---|
| pop_year | Year of the dataset to be downloaded |
| country | Country of which we want the data, in ISO 3166-1-alpha3 format |
| unadj | pd_ic_1km_unadj Specify if the requested data has to be UN adjusted |

Using these parameters, a request to the WorldPop API is made, which returns a JSON containing the links to the CSV and TIFF files for each year for the specified country. By using the *EvaluatedJSONPath* processor, we can extract the requested *pop_year* and download the country yearly dataset using the provided links. The CSV is compressed in a zip file, so the *UnpackContent* processor is required to uncompress the data. The results are then stored in the SILVANUS SAL. Figure 8 and 9 depicts an example of the population density dataset for the Italian peninsula.

**Figure 9 - Tiff of the Italian Population Density dataset year 2000**

| X | Y | Z |
|---|---|---|
| 12.1945832852870453 | 47.0954166656310278 | 2.94151782989501953 |
| 12.2029166185870448 | 47.0954166656310278 | 2.47266912460327148 |
| 12.2112499518870443 | 47.0954166656310278 | 1.94547045230865479 |
| 12.2195832851870456 | 47.0954166656310278 | 3.54073429107666016 |
| 12.1362499521870451 | 47.0870833323310265 | 1.81413447856903076 |
| 12.1445832854870446 | 47.0870833323310265 | 1.87228679656982422 |
| 12.1529166187870459 | 47.0870833323310265 | 1.76710116863250732 |

**Figure 10 - ASCII Gridded XYZ (CSV) Italian Population Density dataset year 2000**

### 2.1.5. Burned Area

The Copernicus burned areas dataset [12] provides critical information on the extent and severity of wildfires across different regions. This dataset is available in two versions: Version 1 and Version 3. Version 1 was released in 2015 and provides data on the burned area extent at a spatial resolution of 20 meters. This version uses a semi-automatic algorithm to identify burned areas based on spectral changes in the satellite images. Version 3 was released in 2021 and represents a significant improvement over the earlier version. This dataset provides more accurate and detailed information on the extent and severity of wildfires at a spatial resolution of 10 meters. The algorithm used in this version incorporates more advanced techniques, including machine learning, to identify burned areas more accurately (an example of this version is shown in Figure 11 for the northern Evia area, where the non-black tiles provide the necessary information). Within the SILVANUS platform, the dataset of burned areas can be used as a data feature for several machine learning models, such as the Fire Danger Index.

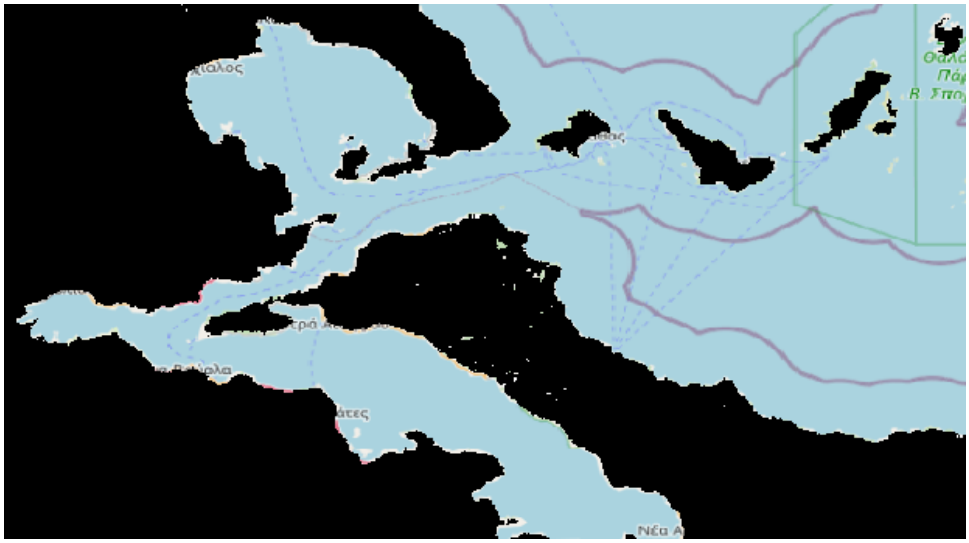


**Figure 11 - Copernicus Burned Areas Tile covering northern Evia.**

The Apache Nifi ingestion pipeline related to the Copernicus Burned Area datasets is initiated with a message (as reported in Table 4) published on the SILVANUS message bus.

**Table 4 - Burned area Sample ingestion message**

| |
|---|
| Queue: ba.nc<br>Message: {"version": "V3"} |

To initiate the process, the script requires users to input the version of the burned areas dataset they want to work with - either V1 (version 1) or V3 (version 3) arguments.

As shown using in Figure 12, the top-to-bottom flow gets initiated by the published message using the ba.nc queue. After the initiation of the pipeline, the first step is to extract the body parameters using a NiFi processor, and subsequently pass the extracted content to a Python script that downloads the netCDF files related to the burned area requested using the manifest files provided by the Copernicus Land Monitoring Service. Once the data has been downloaded, the Apache NiFi processor forwards it to the SILVANUS SAL for further processing and analysis.

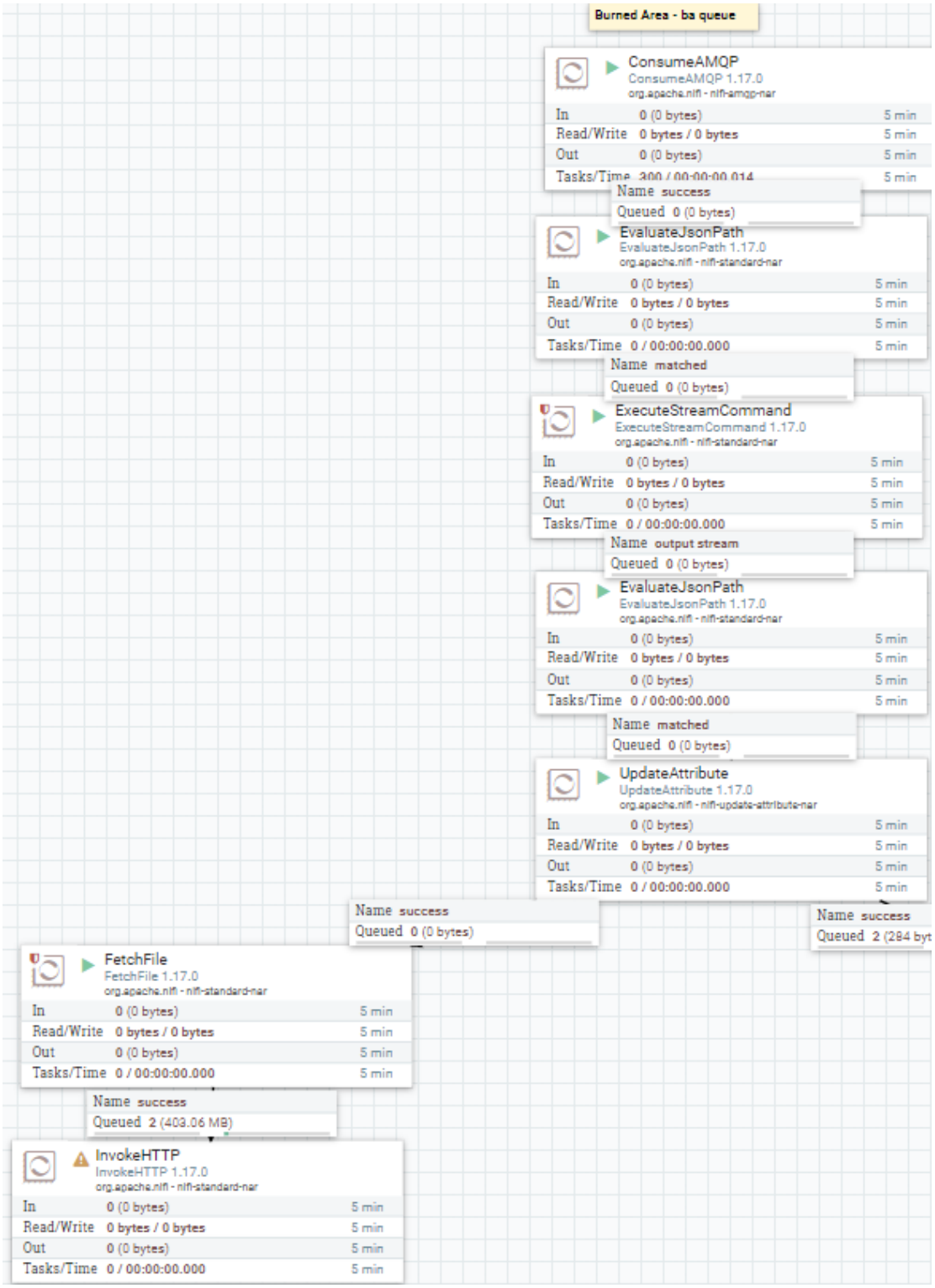



**Figure 12 - Apache Nifi Ingestion Pipeline for Burned areas**

### 2.1.6. Land Surface Temperature

The Copernicus Land Surface Temperature (LST) products [13] offer valuable information about the thermal characteristics of the Earth's surface, which is crucial for monitoring and managing various environmental issues. The spatial resolution of the Copernicus Land Surface Temperature product is 1 km, which means that the product provides information on the

surface temperature of every 1 km x 1 km pixel. The three main categories of Copernicus LST products are:

- The first product, hourly LST from instantaneous observations, provides real-time information on the surface temperature. Figure 13 shows an example of hourly LST, where the thermal characteristics are shown in colored tiles.
- The second product, LST10-DC, provides a 10-day composite of the Land Surface Temperature with a daily cycle.
- The third product, LST10-TCI, provides a Thermal Condition Index that is calculated using a 10-day composite of the Land Surface Temperature.

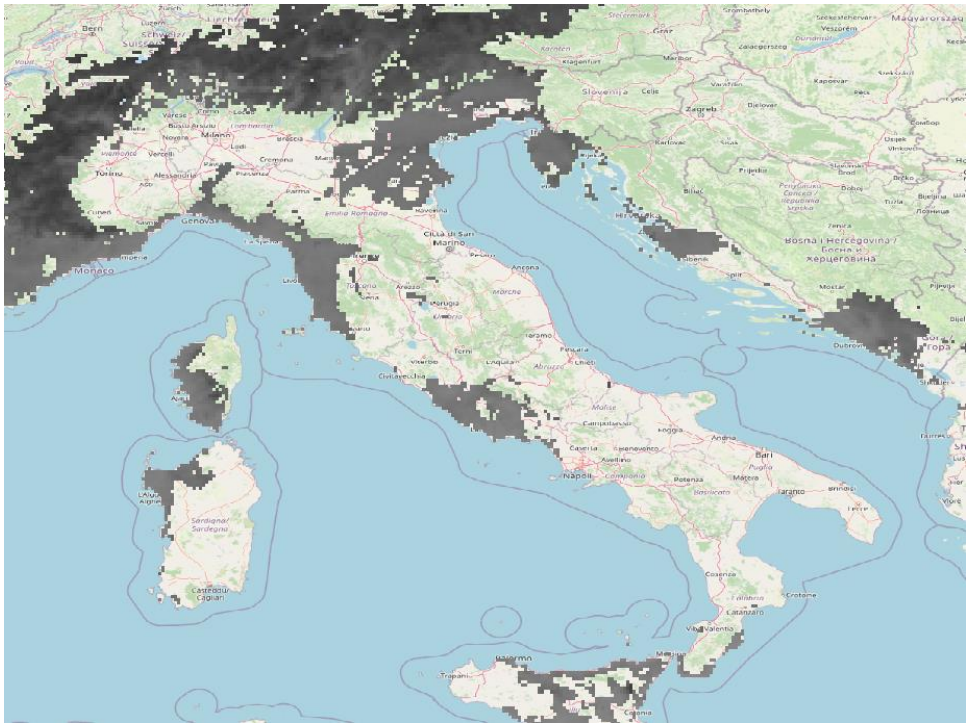


**Figure 13 - Copernicus Land Surface Temperature for Italy – hourly LST**

The Apache Nifi ingestion pipeline related to the Copernicus Land Surface Temperature datasets is initiated with a message (as reported in Table 5) published on the SILVANUS message bus.

**Table 5 - LST Sample ingestion message**

| |
|---|
| Queue: lst.nc<br>Message: {"type": "H"} |

Figure 14 depicts the top-to-bottom Apache NiFi flow; once the ingestion is initiated the first step is to extract the body parameters from the message using a NiFi processor that passes the product type of LST dataset – “H” for hourly LST, “DC” for the LST10-DC and “TCI” for the LST10-TCI - to a Python script. The script, based on the product type (H, DC or TCI), identifies the relevant manifest files from the Copernicus services and downloads the netCDF file containing the requested LST data. Once the data has been downloaded, the Apache NiFi processor forwards it to the SILVANUS SAL.

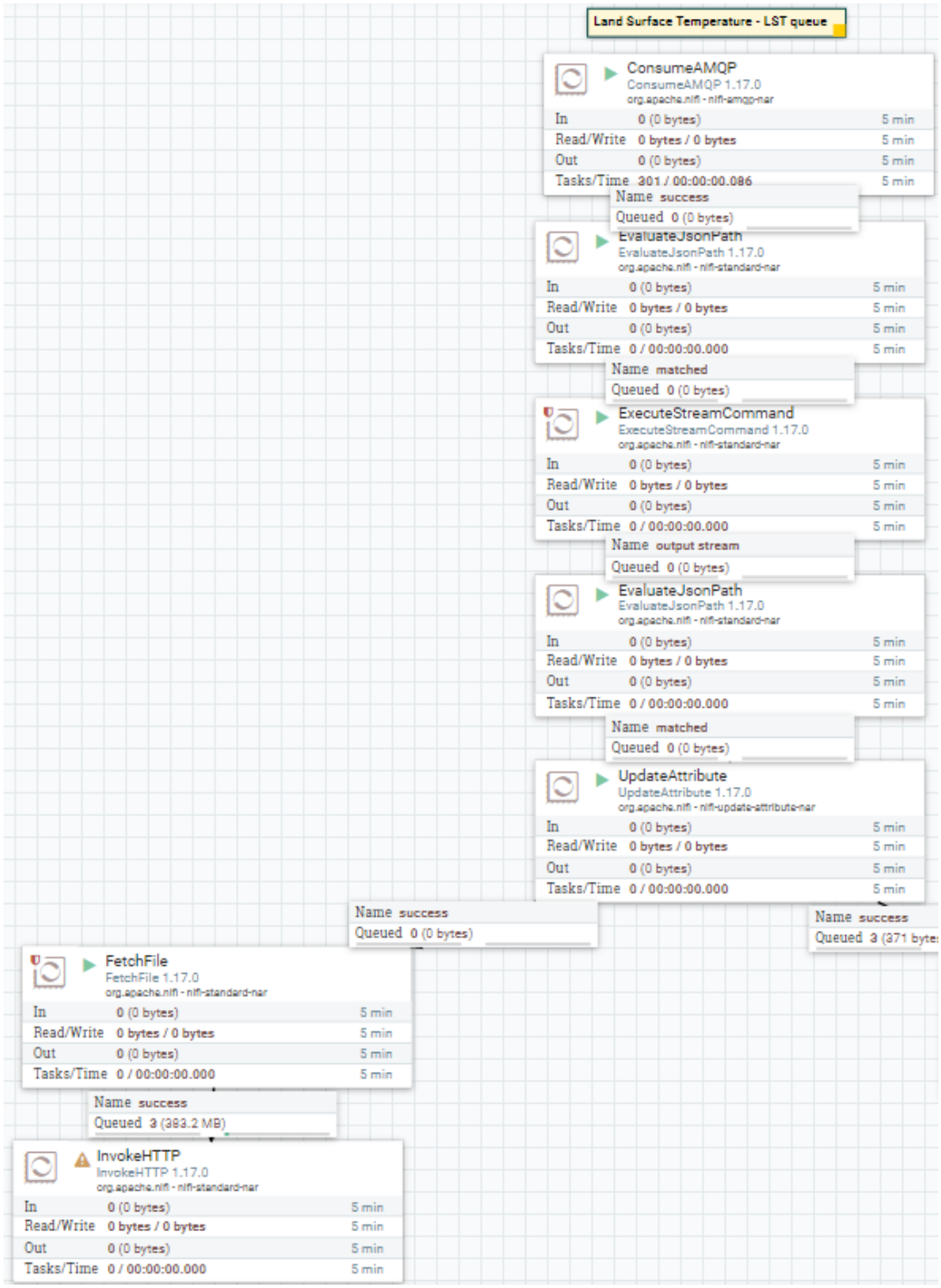



**Figure 14 - Apache NiFi Ingestion Pipeline for LST**

### 2.1.7. Corine Land Cover

Corine Land Cover (CLC) [14] is a land cover classification system that was developed by the European Environment Agency (EEA) in the 1980s. It provides a standardized and detailed mapping of the land cover across Europe and is widely used to monitor changes in land cover and land use over time, distribution of habitats and ecosystem, impacts of climate change and a lot more key use. The Corine Land Cover system divides the land into 44 different classes, ranging from artificial surfaces such as urban areas and transport networks, to natural and

semi-natural vegetation, wetlands, and water bodies. The CLC employs a Minimum Mapping Unit (MMU) of 25 hectares for spatial features and a minimum width of 100 meters for linear features. The mapping is carried out using satellite imagery and other remote sensing techniques, and is updated approximately every 6 years, the latest version of CLC is the 2018 dataset (an example is shown in Figure 15).

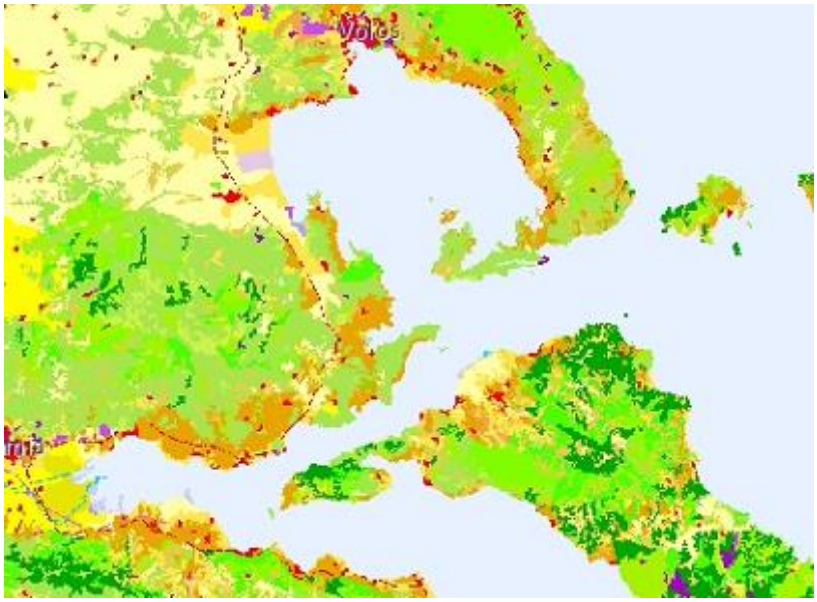


**Figure 15 - Copernicus Corine Land Cover - covering northern Evia [13]**

To download the Corine Land Cover dataset from the SILVANUS platform, an Apache NiFi ingestion pipeline will be developed that consists of several processors and components. The pipeline will be designed to handle multiple datasets obtained from the CMCC Data Delivery System, including CLC, climate data, and forest inventory data.

### 2.1.8. Tree Cover Density

The HRL Forest 2018 primary status layer Tree Cover Density (TCD) has been created in frame of the tender "EEA/IDM/R0/18/009 - High Resolution land cover characteristics for the 2018 reference year" as part of the EEA Copernicus Land Monitoring Service. The TCD raster product provides information on the proportional crown coverage per pixel at 10m spatial resolution and ranges from 0% (all non-tree covered areas) to 100%, whereby Tree Cover Density is defined as the "vertical projection of tree crowns to a horizontal earth's surface" (see Figure 16 and Figure 17 as examples).

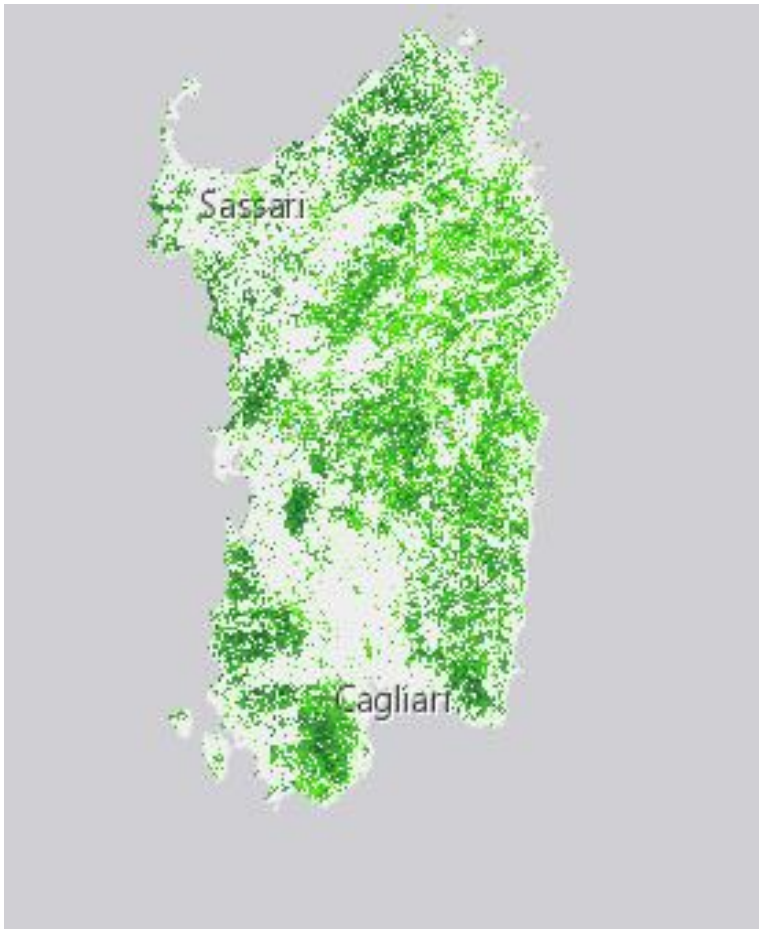



**Figure 16 - Example of the map view of Tree Cover Density of Sardinia**

The selected time window ranges from 01-03-2018 to 31-10-2018. Geometric accuracy (positioning scale) is less than one pixel (10m) according to ortho-rectified satellite image base (Sentinel-2 Level-2A) delivered by the European Space Agency. The accuracy provided is measured at 90.17% overall with a 95% confidence level applied.

Tree Cover Density is an important factor in determining and predicting the behavior of a forest fire. It is consumed by the Fire Spread Model that it utilizes this information, among others, to provide estimations on when and where the fire will be in several future time intervals.

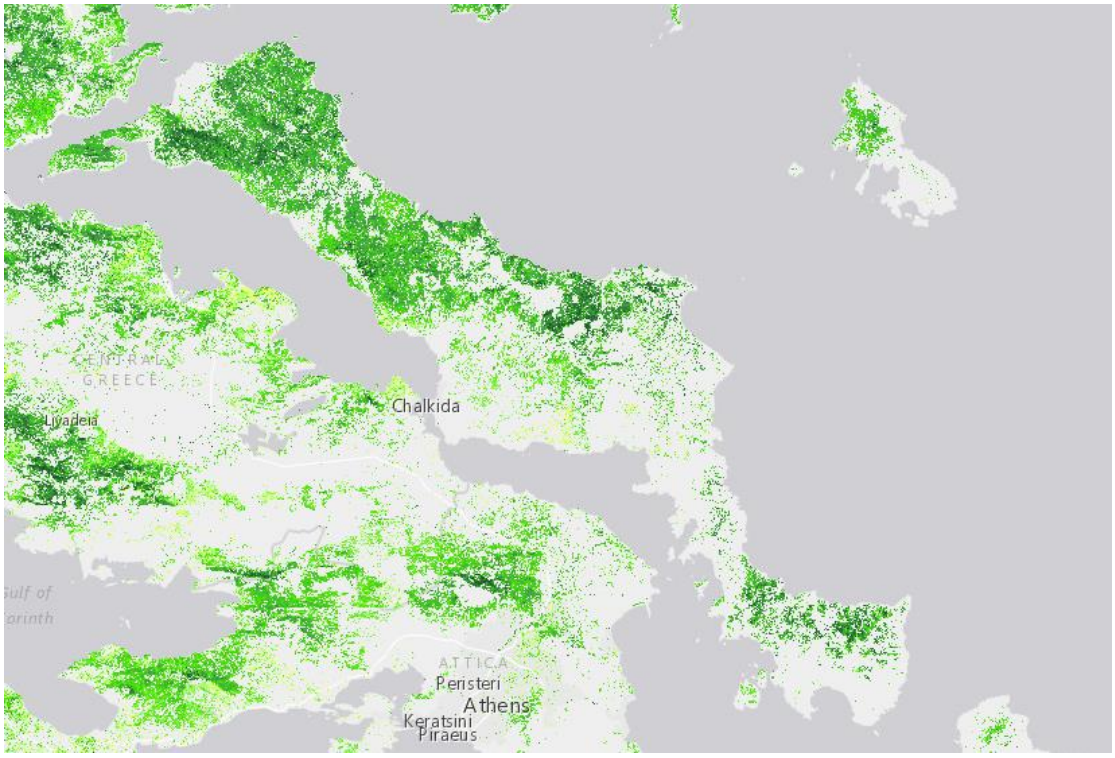



**Figure 17. Example of the map view of Tree Cover Density of Euboea Island, Greece**

### 2.1.9. EUMETSAT Earth Satellite Products for Nowcasting

Different satellite products available within the EUMETSAT platform [15] have been investigated to be used for the identification and assessment of forest fires for nowcasting application (mainly in the range from 0 to 3 hours).

Specifically, two steps have been implemented in parallel:

- satellite station installation at CMCC in Caserta to receive operational data (for research purposes with a latency of less than three hours)
- identification of satellite data of interest for the SILVANUS project, from EUMETSAT data center [16]

As a first step, the satellite station has been installed at CMCC in Caserta in order to receive and manage satellite data from EUMETSAT Meteosat Second Generation (MSG) Satellites that provide images of the full Earth and data for weather forecasts. The MSG system is established under cooperation between EUMETSAT and the ESA, to ensure the continuity of meteorological observations from geostationary orbit, following on from Meteosat First Generation.

There are several ways to access data via the EUMETSAT platform. One of the ways is to download historical data from the EUMETSAT data center that provides a long-term archive of data and generated products from EUMETSAT. In addition, it is possible to access in near-real-time mode through the service called EUMETCAST service. Currently, for research purposes CMCC has a license for EUMETSAT non-essential Meteosat Data with a latency of less than three hours.

Specifically, the system installed at CMCC in Caserta includes the reception of High Rate SEVIRI Level 1.5 Image Data satellite data (available in different spectral bands, geolocated and ready for subsequent processing, from which further meteorological products are extracted) and all

meteorological satellite products. Concerning the system architecture, it includes both storage units to archive the data related to requested products and an easy user interface to access the data (this aspect is still being implemented).

**Figure 18 Satellite station installed at CMCC in Caserta.**

The EUMETSAT platform comprises different product types from several EUMETSAT-operated satellites within the Europe domain, including:

- Images and derived meteorological products from geostationary Meteosat satellites since 1981. For example, the fire products, of interest for the project, are available from 1998 to date. In this way there is an archive of about 24 years which is suitable for the training phase.
- And then there are other products developed by specific work groups but always in the context of EUMETSAT service. These work groups are part of the EUMETSAT network of Satellite Application Facilities including differ application sectors from land surface monitoring to nowcasting applications.

Starting from the EUMETSAT data store, the different products related to fire detection have been identified and reported in the following sections.

#### 2.1.9.1. Active Fire Monitoring

The Active Fire Monitoring product [17] is a fire detection product indicating the presence of fire within a pixel at 3 km spatial resolution (Figure 19). The algorithm distinguishes between potential fire and active fire. It is generated for every image (every 15 minutes) and the algorithm is based on threshold values that use the brightness temperatures of the different channels of Seviri sensors available aboard MSG geostationary satellites. There are 96 daily files with a total estimated size of 190 MB, while for a month, the total amount of data is about 5 GB of data. The data format is GRIB [18]. In addition, there is the same active fire monitoring product with the rapid scanning function with a temporal frequency of 5 minutes so for a month we have about 15 GB of data (in this case, there are 288 daily files). This service is available from March 2009, and detects the fire presence through the optical infrared sensor of IR3.9 Seviri channel [19].

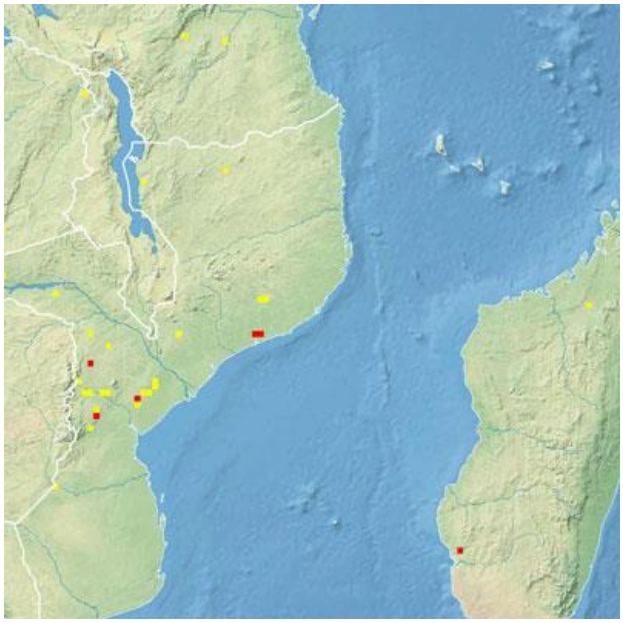


**Figure 19 - Example of image for Active Fire monitoring products. Red indicates a probable fire, while yellow indicates a possible fire.**

The algorithm produces four values for each pixel:

- 0: no fire detection.
- 1: possible fire detection.
- 2: probable fire detection.
- 3: missing.

#### 2.1.9.2. Normalized Difference Vegetation Index

The Normalized Difference Vegetation Index product [20] is derived from the differences in the VIS reflectance. The daily NDVI product estimates the land surface characteristics derived from satellite data. This product is developed using the MSG in a geostationary orbit and using the SEVIRI optical sensor. It is widely used to characterize the density and robustness of the given vegetation cover as well as to identify vegetation stress and drought. Note that no NDVI retrievals will be conducted in cloudy or nighttime conditions. The data is in Hierarchical Data Format (HDF) while the typical file size (for a day) is 9 MB. There are 31 files per month with a total estimated size of 279 MB.

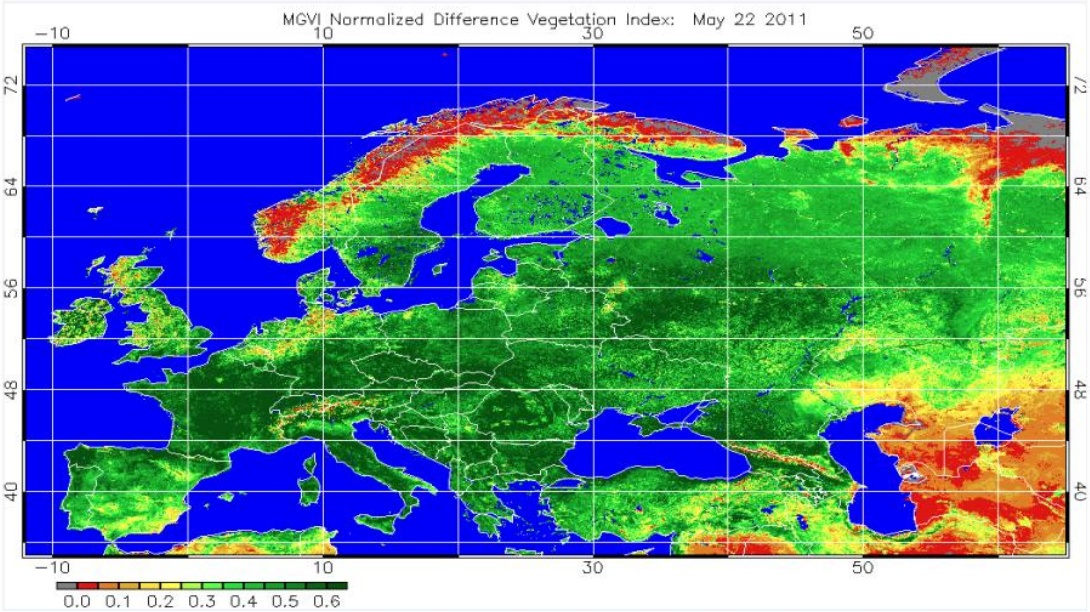



**Figure 20 - Example of image for NDVI product**

As shown in Figure 19 NDVI values greater than 0.1 generally denote increasing degrees in the greenness and intensity of vegetation. Values between 0 and 0.1 are commonly characteristic of rocks and bare soil, and values less than 0 often indicate clouds or snow.

Active Fire Monitoring and NDVI products (both historical and "real-time") will be integrated in the CMCC Data Delivery System and the corresponding Apache NiFi flows will be developed for the ingestion in the SILVANUS platform.

## 2.2. Pre-processing methods & tools

### 2.2.1. Metadata Extraction

In this section we detail the mechanism by which metadata is extracted for data sources ingested through the Apache NiFi pipelines.

#### 2.2.1.1. Custom Metadata Extraction & Pre-processing

Through the implementation of custom ingestion and pre-processing scripts for specific data source providers, we leverage the opportunity to extract and generate metadata at ingestion time. For the purposes of the illustration this manual metadata extracted is implemented for Copernicus Land Services DEM and OSM Feature data.

**Table 6 - Sample of metadata extracted for Digital Elevation Model, after conversion to SILVANUS JSON Format**


```
"descriptor": {
      "obj-class": "DEM",
      "format": {
         "type": "tiff",
         "resolution": "100",
         "output": "raster"
      },
      "access": "data/output/tiff/1674574038.9845824_gargano_dem.tiff",
      "dataset-type": "dem",
      "created": "1674574406.7829435"
   },
   "temporal": {
      "unix": 1674668953.5385747,
```

```
    "datetime": "..."
  },
  "spatial": {
    "bbox": "POLYGON ((...))",
    "coordinates": [
      {"lat": 41.918373450834906,"lon": 16.02958311708167},
      {"lat": 41.88325228808233,"lon": 16.02958311708167},
      {"lat": 41.88325228808233,"lon": 16.08722436464913},
      {"lat": 41.918373450834906,"lon": 16.08722436464913}
    ],
    "pilot": "gargano"
  },
  "tags": {
       "geometry": {
         "type": "Polygon",
         "coordinates": [[...]]
    "tiff_metadata": {
      "driver": "GTiff",
      "dtype": "float32",
      "nodata": -3.4028234663852886e+38,
      "width": 4309,
      "height": 2694,
      "count": 1,
      "crs": 'WGS84'
      "transform": [
        25.0,
        ....
        1.0
     }
```


In Table 6, an example of metadata extracted for a DEM file is reported, where we have identified the initial attributes to be used in the Metadata Indexing component (i.e., the mechanism by which data objects are queried and retrieved). As shown above, we split the common metadata attributes into three top level categories.

1. *Descriptor*: contains the primary object attributes including a top-level dataset identifier, sub-class identifier, format and finally access mechanism.
2. *Spatial*: denotes spatial reference information, e.g., bounding box, pilot string identifier, CRS.
3. *Temporal*: denotes the temporal reference, e.g., date, date range.

A key consideration made for the above spatial and temporal references is the chosen format specification, as we address the challenge of data ingestion and usage from multiple user products. In the above we also consider the free form field *tag* that allows for individual data sources to add additional key/value data that is specific to data domain. For example, while the common attributes for Earth Observation and Weather data objects should remain consistent, we offer the tag field to encode any additional dataset features that may be of interest to allow for a more expansive metadata query.

The format of this metadata descriptor should remain consistent across all data source providers, from cloud based EO/ Weather to far-edge based IoT, UAV, and UGV devices. As such this metadata format has been iteratively designed through open collaboration with partners, to address specific requirements for both user products and data source providers.

#### 2.2.1.2. Automated Metadata Extraction & Pre-processing

In order to extract in an automatic way the metadata related to the data objects before being ingested into the SILVANUS platform, a metadata extraction system has been designed and implemented; it is based on the implementation of different parsers according to the data formats of the objects ingested in the SAL.

The first step was to analyze the type of the ingested data into the SILVANUS Platform, to design the parser for each format according to the metadata structure. As reported in Deliverable D8.1 the following data formats related to the ingested data have been identified:

1. Satellite Earth observations data: SAFE [21]
2. Digital Terrain & Elevation Maps: TIFF, XML
3. Topographic Maps: PNG
4. Corine Land Cover data: Raster, Vector
5. Base Layers, Roads & Railways: OSM (XML), PNG
6. Population Density: TIFF, CSV
7. Climate/Weather: NetCDF, GRIB

Excluding data whose metadata is manually managed during the ingestion phase (described in the previous Section), the set of data format for which it is needed the automated metadata extraction are reported in the following table:

**Table 7 – Data formats for the different dataset ingested in the SILVANUS Platform**

| Data Category | Data Type | Data Format |
|---|---|---|
| Climate/Weather | Burned Area | NetCDF |
| | | GRIB |
| | Land surface temperature | NetCDF |
| | Seasonal/Short term Forecast | NetCDF |
| | NDVI | NetCDF |
| Static Reference Data | Population Density | TIFF, CSV |
| | Corine Landcover | RASTER, VECTOR |
| Satellite Earth Observations | Sentinel (Sentinel-2, Sentinel-3) | SAFE |

##### 2.2.1.2.1. Metadata Extractor

The High Level Architecture of the Metadata Extractor is reported in Figure 21; it processes the data ingested in the SILVANUS Platform according to the different data formats and produces a JSON file containing the extracted metadata. The JSON files produced by the system are stored in the Metadata Index component of the SILVANUS Platform.

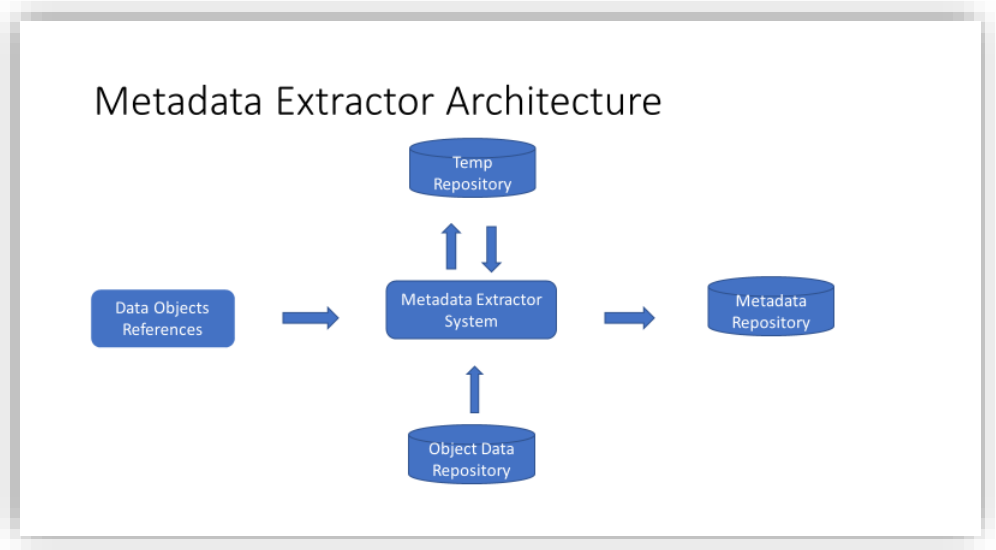



**Figure 21 - High Level Metadata Extractor Architecture**

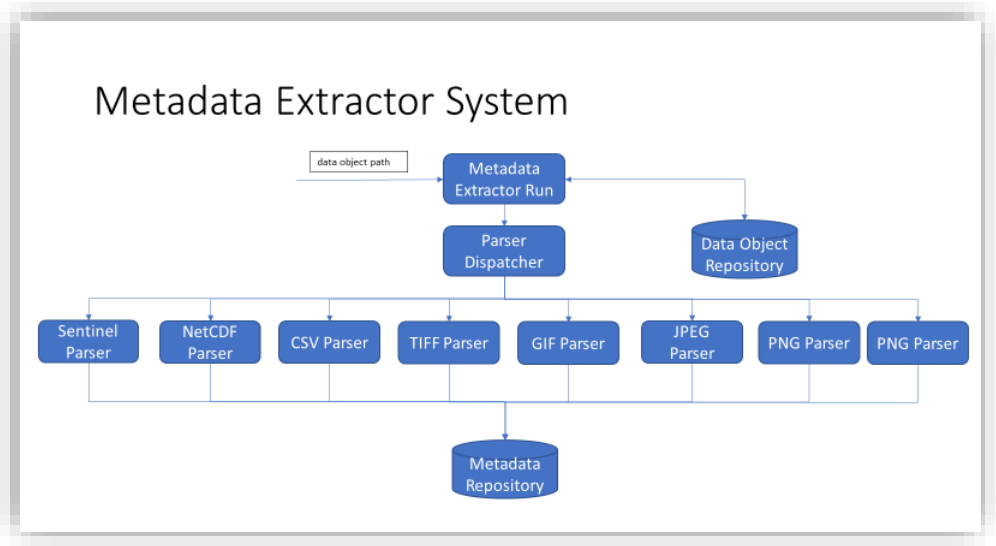



**Figure 22 – Software components of the Metadata Extractor**

The different software components of the Metadata Extractor are presented in Figure 22; in particular the system is composed of the following modules:

- The Metadata Extractor Run takes care of acquiring the necessary input parameters and passing this information to the ParserDispatcher
- The ParserDispatcher according to the data format of the input received, will activate the appropriate parser.
- The selected parser will produce and store the JSON file related to the metadata

In order to implement the parser for the different data formats of the data objects ingested in the SILVANUS Platform, an analysis of the available Python libraries has been carried out.

For each data format a custom parser has been implemented using a specific Python library, as detailed in the following:

- Sentinel SAFE format: Beautiful Soup [22] used for the metadata extraction from the SAFE XML file
- NetCDF: netCDF4 python library [23]
- GRIB: gributils python library [24]
- TIFF: Rasterio python library [25]

In order to integrate the Metadata Extractor within the Data Ingestion Pipeline, the Apache Nifi Dataflow shown in **Error! Reference source not found.** has been developed.

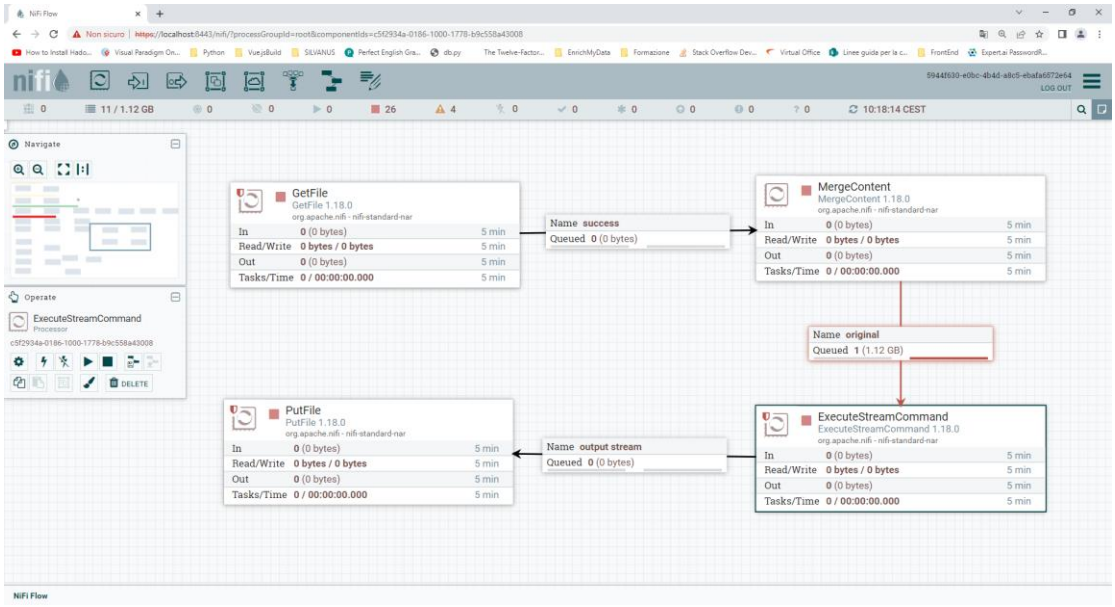



**Figure 23 – Apache NiFi Metadata Extractor Dataflow**

The Dataflow has four different types of processors: a *GetFile* processor, a *MergeContent* processor, an *ExecuteStreamCommand* processor and a *PutFile* processor.

The *GetFile* processor takes care of accessing the files related to the data ingested in the SILVANUS Platform, managing the necessary information (e.g. file name, size of the file, …). In case of an archive file (e.g. .zip for Sentinel data) several FlowFiles are generated containing information necessary for metadata extraction for each file contained in the archive.

The *MergeContent* processor takes care of managing the information in the Flowfiles received from the *GetFile* processor. The *MergeContent* processor analyze the different FlowFiles and create a single FlowFile with the important information that is needed to extract metadata.

The *ExecuteStreamCommand* is responsible for calling the Metadata Extractor system that will create a JSON metadata file according to the information in the FlowFile received from the *MergeContent* processor.

The output provided by the *ExecuteStreamCommand* will be managed by the *PutFile* processor which, specifically, will store the JSON metadata file - related to the processed data object - into the repository set in the processor configuration.

The *ExecuteStreamCommand* processor and parser scripts have been leveraged and integrated directly with the Copernicus Ingestion Pipeline to operate directly on SAFE products at ingestion time within the Data Ingestion Pipeline. The SAFE Metadata Parser is passed a reference to the temporary storage location of the current product after retrieval through the Copernicus API pipeline. At this stage the Parser resumes normal operation and outputs the extracted metadata in JSON format. This JSON object is then merged and attached via SAFE FlowFile attributes as the relevant descriptor and output to the SAL via a REST Interface as described in 2.1.3.

In Figure 24 it is shown an example of the JSON metadata file for a Sentinel product.


```
{
	"descriptor":{
		"obj-class": "EO",
		"format": {
			"type": "SAFE"
		},
		"access": "Product/file/path/manifest.xml",
		"dataset-type": "sentine-2",
		"created": "1674574406.7829435"
	},
	"spatial": {
		"bbox": "POLYGON ( (16.0295831170816712 41.9183734508349062, 16.0295831170816712
41.8832522880823319, 16.0872243646491313 41.8832522880823319, 16.0872243646491313 41.9183734508349062,
	16.0295831170816712 41.9183734508349062))",
		"coordinates": [
			{
				"lat": 41.918373450834909,
				"lon": 16.02958311708167
			},
			{
				"lat": 41.88325228808233,
				"lon": 16.02958311708167
			},
			{
				"lat": 41.88325228808233,
				"lon": 16.08722436464913
			},
			{
				"lat": 41.918373450834909,
				"lon": 16.08722436464913
			}
		],
		"pilot": "gargano"
	},
	"temporal": {
		"datetime": "date",
		"start": "date",
		"end": "date"
	},
	"tag": {
		"platform": "xyz",
		"cloudcoverpercentage": "10"
	}
}
```


**Figure 24 – Example of JSON containing metadata related to a Sentinel Product**

This demonstration and integration mechanism between automated parsers and the Copernicus SAFE pipeline will be implemented across all data sources.

### 2.2.2. TIFF ROI Extraction

Starting from a requirement of the DEM Ingestion pipeline to extract relevant pilot areas from larger DEM Tiles offered by Copernicus, a TIFF ROI Extraction tool has been implemented as a general approach for manipulating TIFF images (DEM output data format).

There are three pre-processing functionalities provided by the custom ingestion script as described in the following:

1. *Extract Region of Interest from TIFF Image*. This step runs a cropping function on the original TIFF image based on a pre-defined pilot bounding area. The extracted region is output to a new TIFF image object. This process is based on a bounding box or polygon coordinate specification with reference to a common Coordinate Reference System (CRS) 'WGS-84'. This functionality allows user products to specify custom regions that can be useful to produce feature inputs to ML-based services and visualization layers.
2. *Preserve / update TIFF object metadata*. As the pre-processing script modifies the original image object provided by Copernicus, we must also consider the metadata encoded within the TIFF file. That is, the image bounds and maximum elevation (encoded as a '1' value in the height map) must be updated in the resulting output image. We also consider the preservation of static metadata / data lineage of metadata which is transferred to the new TIFF image object shown in Figure 25 .
3. *Extract Indexing metadata for the TIFF object*. As the final step, the relevant metadata is extracted and stored in the Metadata Index which will be used to search for and retrieve this data object from the SILVANUS SAL.

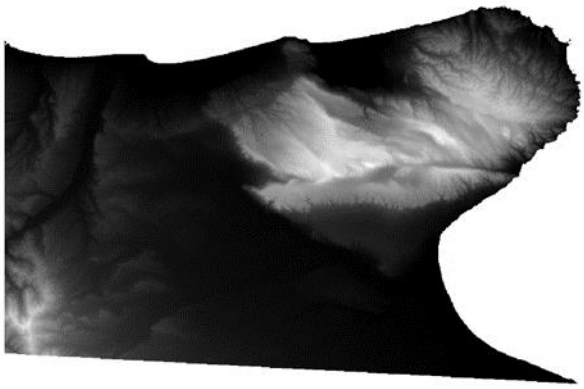


**Figure 25 - Tiff ROI Extracted from DEM Tile at ingestion time**

## 2.3. Post-processing methods & tools

### 2.3.1. Sentinel Derived Indices

Sentinel Derived Indices are mathematical algorithms that are used to extract specific information from satellite or aerial imagery of the earth's surface. These indices provide information about different environmental and land use parameters such as vegetation, land cover, soil moisture, atmospheric conditions, and more.

Some commonly used Sentinel Derived Indices are:

- Normalized Difference Vegetation Index (NDVI) uses red and near-infrared light to measure the amount of photosynthetically active vegetation in a particular area. It is calculated as the difference between the near-infrared and red bands

divided by their sum. NDVI values range from -1 to 1, with higher values indicating more vegetation.

- Normalized Difference Moisture Index (NDMI) – NDMI is an index used to estimate vegetation moisture content. It is calculated using the reflectance values of near-infrared and shortwave-infrared bands and provides information about the water content in the vegetation. NDMI values range from -1 to 1, where negative values indicate dry conditions and positive values indicate moist conditions.
- Normalized Difference Water Index (NDWI) - NDWI is an index used to map and monitor open water bodies and wetland areas. It is calculated using the green and near-infrared bands and provides a quick and simple method for mapping water features. NDWI values range from -1 to 1, with higher values indicating areas with higher amounts of water, while negative values correspond to areas with little or no water.

Sentinel Derived Indices can be calculated from Sentinel-2 satellite image bands [26] downloaded from the Copernicus Open Access Hub. Each band has a specific spatial resolution, and it will be limiting the output resolution for the calculated product index.

**Table 8 - Spectral bands for the Sentinel-2 sensors**

| Sentinel-2 bands | Description | Spatial resolution (m) |
|---|---|---|
| **Band 01** | Coastal aerosol | 60 |
| **Band 02** | Blue | 10 |
| **Band 03** | Green | 10 |
| **Band 04** | Red | 10 |
| **Band 05** | Vegetation red edge | 20 |
| **Band 06** | Vegetation red edge | 20 |
| **Band 07** | Vegetation red edge | 20 |
| **Band 08** | NIR | 10 |
| **Band 08A** | Narrow NIR | 20 |
| **Band 09** | Water vapor | 60 |
| **Band 10** | SWIR – Cirrus | 60 |
| **Band 11** | SWIR | 20 |
| **Band 12** | SWIR | 20 |

The developed application is divided in two parts:

1- A wrapper around SentinelSat to request and download raw Sentinel data.
2- A high-level object oriented to calculate Sentinel Derived Indices.

To get Sentinel satellite images, we use the SentinelSat Python package. SentinelSat is a Python package for searching, downloading, and retrieving Sentinel satellite data from the Copernicus Open Access Hub. It provides a convenient interface to query, download, and pre-

process Sentinel data, and access the metadata of the products. Despite SentinelSat can search and download Sentinel-1, Sentinel-2, Sentinel-3 and Sentinel-5P, for this purpose, we only use Sentinel-2 datasets.

To access the data and information products available on the Copernicus Open Access Hub, users need to create an account first on their website [27]. The registration process requires users to provide basic personal information, such as their name and email address, and agree to the platform's terms of use. Once you have created an account, you can use your username and password to log in to the Copernicus Open Access Hub. This will give you access to the data and information products available on the platform, as well for downloading and processing the data.

With a valid username/password, SentinelSat allows to search for Sentinel data based on various parameters, such as:

- *Footprint*: geographical search of the products whose footprint intersects or is included in a specific geographic type. The geographic type value can be expressed as a Point (lon,lat), a Bounding Box (lon1,lat1:lon2,lat2) or as a Polygon (lon1,lat1:lon2,lat2:lon3,lat3...) in decimal degrees format.
- *Start Date*: start date of the query.
- *End Date*: end date of the query.
- *Cloud coverage*: maximum cloud coverage of Sentinel-2 products for each area covered by a reference band in percent.
- *Product type:* we have limited to 2 products, the S2MSI1C and the S2MSI2A, but it is preferable to always choose S2MSI2A, which is the Level-2A product, that provides an atmospherically corrected Surface Reflectance (SR) images, derived from the associated Level-1C products (S2MSI1C).

Depending on the chosen date range and cloud coverage, the query can return none, one or more results for the chosen local and product type. Each result corresponds to an image (tile) with a 100x100 $km^2$ area in UTM/WGS84 projection. For each one, the chosen indices will be calculated and reprojected to ESPG:4326 (WGS84). At the same time, it will be resampled, changing the cell values due to changes in the raster cell grid (default: 100m). Since the original resolution can be very high (10m), in most cases, the resampling will be a down sampling to a lower resolution.

Most recently published products of Copernicus Open Access Hub are kept “online” according to a sliding window period of at least one month, based on the first publication date of the products in the catalogue. After that, it turns into an “offline” product. The download request of an offline product automatically triggers the request for restoring the product back online from the historical archive. Once restored, the product(s) can be downloaded. The application is transparent to this and can handle without any problems.

Multiple indices are predefined and can be used directly:

- SR|RVI - Simple Ratio | Ratio Vegetation Index: (NIR / Red)
- NDVI - Normalized Difference Vegetation Index: (NIR - Red) / (NIR + Red)
- NDWI - Normalized Difference Water Index: (Green - NIR) / (Green + NIR)
- NDMI - Normalized Difference Moisture Index: (NARROW_NIR - NIR) / (NARROW_NIR + NIR)
- SAVI - Soil Adjusted Vegetation Index: [(NIR - Red) / (NIR + Red + L)] * (1 + L)

- OSAVI - Optimized Soil Adjusted Vegetation Index: (NIR - Red) / (NIR + Red + L)
- EVI - Enhanced Vegetation Index: 2.5 * [(NIR - Red) / (NIR + 6*Red - 7.5*B + 1)]

If necessary, other indices can be added to the code in a very simple way.

The output generated can be at NetCDF (default) and/or GeoTIFF format, each one with a 2D dimension array of float type data values, which corresponds to each of the Sentinel Derived Index chosen to do.

The developed application is in Silvanus GitHub platform [28]

### 2.3.2. OpenStreetMap Features Conversion

Considering that roads and railways are essential elements for the management of spaces where fires can occur, it is important to carry out their survey and geographic location so that they can be used in models for fire prevention and fire spread. Since the format used by those models is NetCDF, it is necessary to convert information provided by OSM to that format.

To accomplish this, a Python application has been implemented in order to perform the following 4 steps (as shown in Figure 26):

1. Get roads/railways features from OSM, in a bounding box area given by (minX, minY, maxX, maxY).
2. Create a grid with Δxy of resolution to the desired area.
3. Clip operation.
4. Convert the clip process output to NetCDF format.

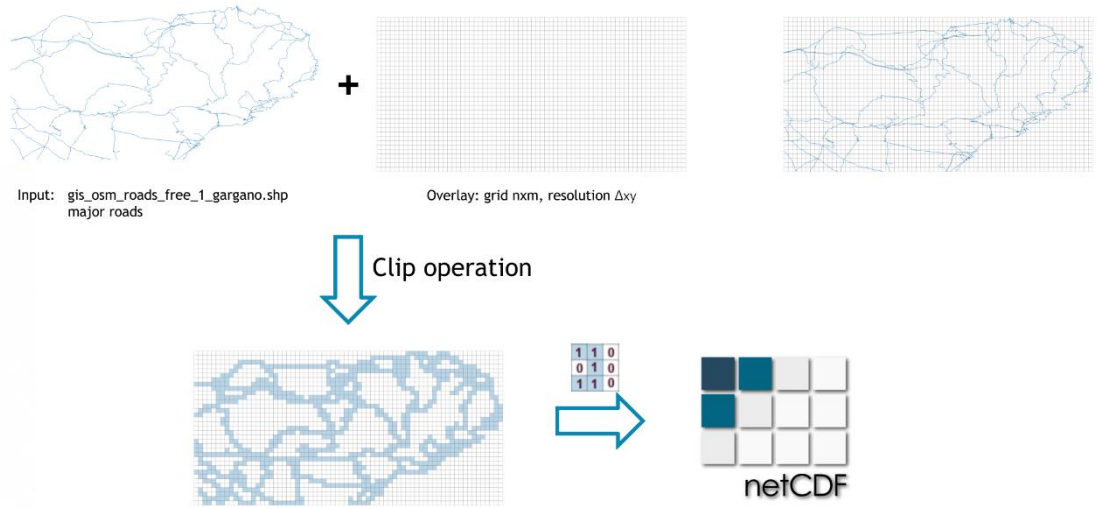



**Figure 26 - OSM feature conversion main process diagram**

OSM features can be directly downloaded from the website for all countries. Some countries are subdivided by regions. This data is created by Geofabrik [29] and updated daily. The following information are provided in shapefiles: buildings, land use, natural, places, point, railways, roads, and waterways. An updated PDF file [30] can be consulted with all the information and description about a general-use mapping of the basic features. This file also has a list of all geometry codes and a brief description of each one.

For our purpose, we have only considered 6 type of pre-configured features (see Table 9).

**Table 9 - OSM features considered**

| Feature Type | Geometry code | | Description |
|---|---|---|---|
| | From | To | |
| **Major roads** | 5111 | 5119 | Motorway; Important roads; Primary roads; Secondary roads; Tertiary roads. |
| **Minor roads** | 5121 | 5129 | Smaller local roads; Roads in residential areas; Streets where pedestrians have priority; Pedestrian only streets; Dedicated roads for bus, usually closed for any mode of transport except public transport. |
| **Link roads** | 5131 | 5139 | Roads that connect from one road to another of the same of lower category. |
| **Small roads** | 5141 | 5149 | Service roads for access to buildings, parking lots, etc.; For agricultural use, in forests, etc. Often gravel roads; Tracks can be assigned a "tracktype" from 1 (asphalt or heavily compacted) to 5 (hardly visible). |
| **No cars roads** | 5151 | 5199 | Paths for horse riding; Paths for cycling; Footpaths; Unspecified paths; Flights of steps on footpaths; Unknown type of road or path |
| **Railways** | 6101 | 6119 | Regular railway tracks; Light railway tracks, often commuter railways; Underground railway tracks; Tram tracks (may be incident with roads); A monorail track; A narrow gauge railway track; A miniature railway track; A funicular, or cable railway usually on a steep incline; A rack railway; An overhead tow-line for skiers; An open chairlift run; A cabin cable car run; An aerialway where the cabins go around in a circle; An aerialway for the transport of goods; Another type of lift. |

The main Python packages used to develop this application are:

- GeoPandas [31]: an open-source project to work with geospatial data. It extends the datatypes used by Pandas [32] to allow spatial operations on geometric types.
- Dask-geopandas [33]: a project merging the geospatial capabilities of GeoPandas and scalability of Dask [34]. Dask provides advanced parallelism and distributed out-of-core computation with a dask dataframe module designed to scale pandas.
- Shapely [35]: a BSD-licensed Python package for manipulation and analysis of planar and geometric objects.
- netCDF4-python: a Python interface to the NetCDF C library.

Although the coordinate reference system for the output data is pre-configured to ESPG:4326 (WGS84), it can be updated using a configuration file.

The output will be a NetCDF file with a multi-dimensional array of data values, along with associated 2D dimensions (lat, lon) and metadata. Each variable represents a OSM extraction feature in integer data type with the value 1 in cells where exist the feature and the value 0 in cells where it doesn't exist any selected feature.

The developed application is in Silvanus GitHub platform [36].

# 3. Weather and Climate Data

In this section we describe the weather and climate models developed in SILVANUS to provide the short term and seasonal forecast for the Fire Weather Index (FWI).

## 3.1. Weather and Climate Models

### 3.1.1. Short-Term Forecast & Fire Weather Index

The numerical Weather Research and Forecasting Model (WRF) [37] in version 4.2.1 has been used to develop short-term weather forecasts. It has been selected among many Numerical Weather Prediction (NWP) models, since WRF has been extensively used in other European projects focused on the Adriatic Sea and in particular on the Apulia Region (e.g., Framesport, https://www.italy-croatia.eu/web/framesport; Stream, https://www.italy-croatia.eu/web/stream).

The development of the WRF began in the second half of the 1990s, thanks to a partnership between different research centers and universities, including the National Center for Atmospheric Research (NCAR) in Boulder, CO. WRF has the characteristic of being completely compressible and not hydrostatic. It represents a flexible atmospheric simulation system, which allows you to operate on very diverse spatial scales. Spatial Integration scheme 6th order centered difference

The main features of the used configuration are as follows:

- Vertical coordinates → *sigma-pressure* with 60 vertical levels.
- Time Integration Scheme → Runge-Kutta (RK) of the third order of accuracy.
- Horizontal discretization → Arakawa C staggered grid.
- Spatial Integration scheme → 6° order centered difference.

| Lat | Lon | (degrees) |
|---|---|---|
| 39.594 | 13.7157 | **min** |
| 43.3795 | 18.8843 | **max** |

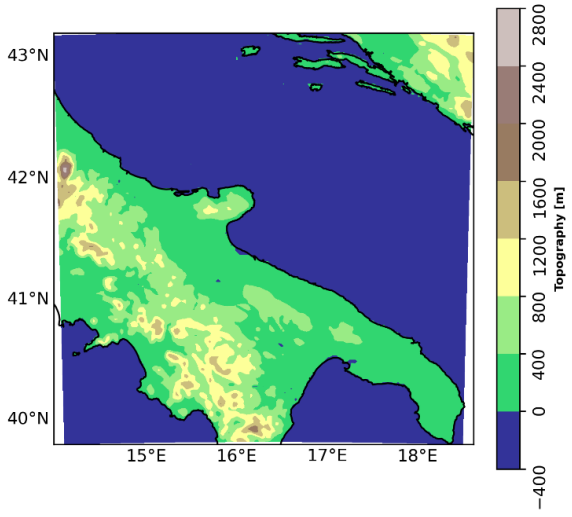



**Figure 27 - Bounding box (top) and simulated domain (bottom)**

WRF has been run over a domain focused on the Apulia Region (Figure 27), using regular Lat-Lon grid in Lambert conformal projection with ~ 2km of horizontal resolution using different forcing data ( e.g. ECMWF, NCEP Data).

Two different run types have been performed, using dissimilar ECMWF forcing over a domain centred on the Apulia Region:

1. Hindcast simulations driven by the Atmospheric Model high resolution analysis (IFS) provided by the ECMWF with spatial resolution of 0.075 degrees (~ 9km) over two periods: 01/04/2019–31/10/2019 and 01/04/2020–31/10/2020.
2. WRF semi-operational simulations (delay~3 days) with 72 hours forecast range initialized by HRES forecast provided by the ECMWF with spatial resolution of 0.075 degrees (~9km). Preliminary tests have been performed, running daily 72h simulation from 01/08/2020 to 31/08/2020 (Figure 28). Each run takes about 1 hour.

Actually, the WRF operational configuration is under development. Tests with the HRES (ECMWF product) and GFS (NCEP product) global forcings are ongoing.

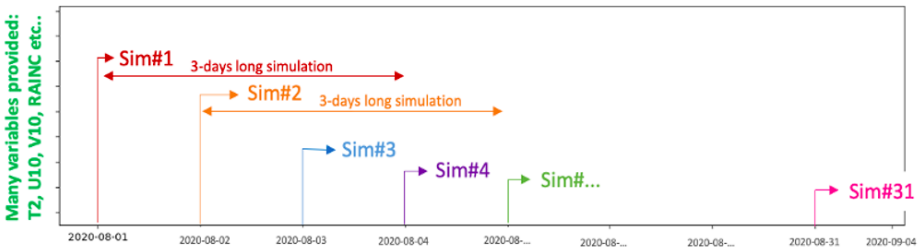



**Figure 28 - WRF operational scheme performed for August 2020**

A preliminary assessment of the WRF performance in reproducing the atmospheric weather variability, on the overmentioned time period, has been conducted, validating WRF (initialized by IFS) with two different datasets:

1. VHR-REA_IT: The Very High-Resolution Dynamical Downscaling from ERA5 at ~2km resolution obtained with the COSMO-CLM atmospheric model [38]. VHR-REA_IT provides hourly data at convection permitting scale (horizontal grid spacing 0.02°, 2.2 km) over the Italian Peninsula (Lon = 5◦ W–20◦ E; Lat = 36◦ N–48◦ N) from 1989 to 2020. This dataset is available on the CMCC Data Delivery System [39].
2. In situ observations provided by the Servizio Agrometeorologico Regionale ARIF Puglia [40]. The location of the network of stations investigated over the Apulia Region is shown in Figure 29. Meteorological measurements at hourly (e.g., 2m temperature, relative humidity) and sub-hourly (e.g., precipitation, wind speed) frequency are available for each weather station.


**Figure 29 - Weather Station Map adapted by mobile stations are shown.**

The models' skills are investigated inspecting the following meteorological variables: hourly 2m temperature (T2), hourly relative humidity (RH), hourly wind speed (W) and daily cumulative precipitation (P). These variables are chosen, since they are used to derive a wide set of wildfire weather indicators.

In order to validate the gridded datasets against the observations, the nearest points in WRF-2km and VHR-REA_IT with respect to the weather stations (Figure 29) have been selected. The optimization method "Nearest neighbour earch" [41] has been used.

Several statistical measures have been computed to measure the performance of WRF-2km and VHR-REA_IT with respect to the in-situ observation, such as the Root Mean Square Error, the Correlation coefficient, and the standard deviation of their variances. The results are shown in the Taylor Diagrams presented in Figure 30 and Figure 31 for 01/04/2019-31/10/2019 and 01/04/2020-31/10/2020 respectively and summarized below.

The Taylor diagram allows the comparison among datasets $S$ with respect to a reference $O$, in terms of Root Mean Square Error ($RMSE$, Equation 1), Correlation Coefficient ($RHO$, Equation 2), and Standard Deviation ($\sigma$, Equation 3).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S_i - O_i)^2} \qquad (Eq.1)$$

$$RHO = \frac{\sum_{i=1}^{N}(S_i - S)(O_i - O)}{\sqrt{\sum_{i=1}^{N}(S_i - \overline{S})^2}\sqrt{\sum_{i=1}^{N}(O_i - \overline{O})^2}} \qquad (Eq.2)$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2} \qquad (Eq.3)$$

Where $N$ is the total number of samples, $X_i$ is the value in the data distribution and $\mu$ is the population mean.

Figure 30 and Figure 31 highlight that for both the analyzed periods, WRF-2km is able to reproduce the relative humidity, 10m wind speed and 2m temperature variables, reaching correlations higher than 0.9 ($pvalue \leq 0.05$ ) with respect to the observed variables, while it tends to overestimate the precipitation. Nevertheless, since the correlation between the WRF-2km daily cumulated precipitation and the observed one is higher than 0.5 ($pvalue \leq 0.05$ ), the WRF-2km configuration is considered suitable for the project purposes.

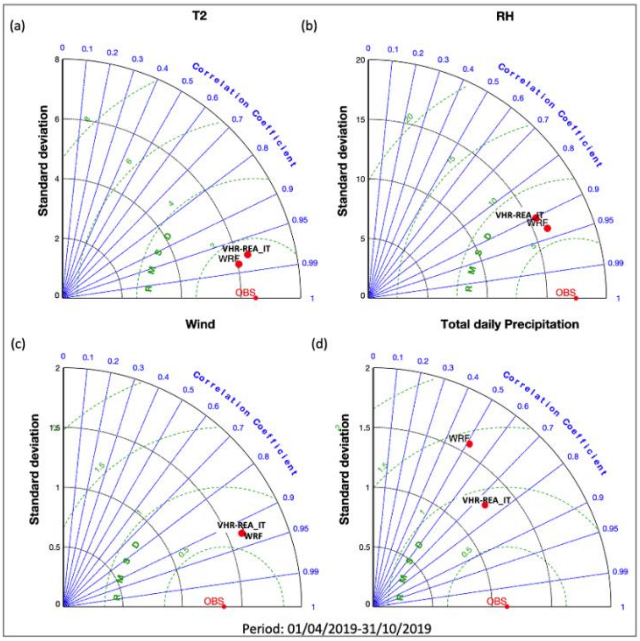



**Figure 30 - Taylor Diagram for (a) 2m-temperaure, (b) relative humidity, (c) 10m wind speed and (d) total daily precipitation over the period 01/04/2019-31/10/2019.**

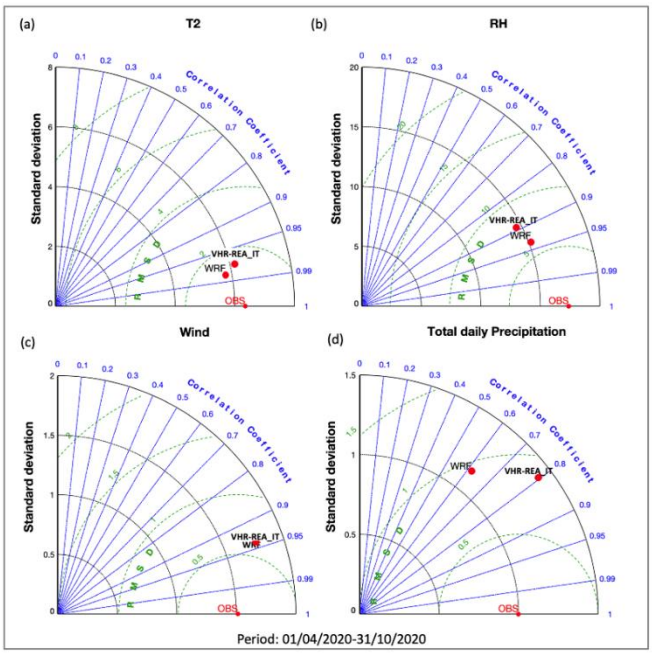



**Figure 31 - As Figure 30 but related to the period: 01/04/2020-31/10/2020**

After assessing good model performance in reproducing the above four mentioned variables, the Fire Weather Index (FWI) is computed for each dataset. The FWI is obtained integrating two intermediate indices: the Initial Spread Index (ISI) and the Buildup Index (BUI), which for their computation need in input the T2, RH, P and W. ISI estimates a spread potential, integrating fuel moisture for fine dead fuels and surface wind speed, while BUI estimates the potential heat release in heavier fuels. This meteorological based index is used worldwide to evaluate fire danger: the higher the FWI is, the more favourable the meteorological conditions to trigger a wildfire are [42].

A comparison between the Probability Density Function (PDF) of the FWI computed with WRF-2km and VHR-REA_IT over the entire Apulia Region, in the two reference periods, is shown in Figure 32

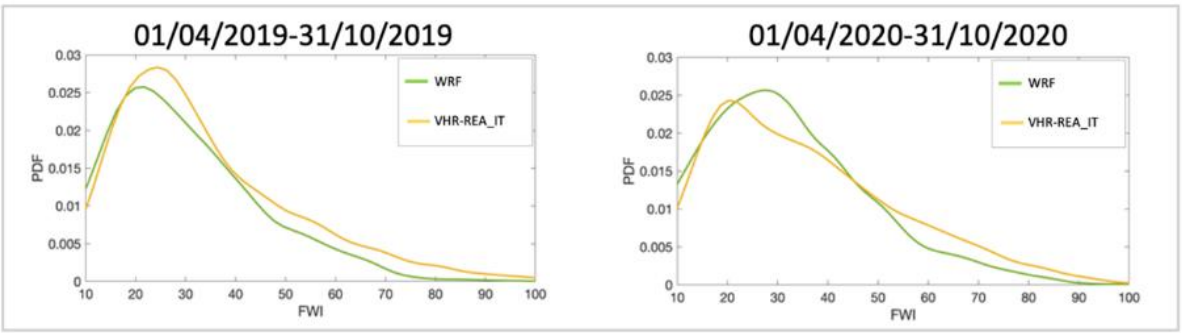



**Figure 32 - The PDFs of the FWI computed with WRF (green line) and VHR-REA_IT (yellow line), over the period: 01/04/2019-31/10/2019 (left) and 01/04/2020-31/10/2020 (right).**

The FWI PDFs in WRF-2km and ERA5-2km are in good agreement with each other in terms of mean value, skewness, kurtosis, and range values for both the analysed time windows.

An example of FWI map, obtained time-averaging a 72h simulation (WRF initialized by HRES Forecast) with lead day 01/08/2020, over the entire simulated domain, is shown in Figure 33.

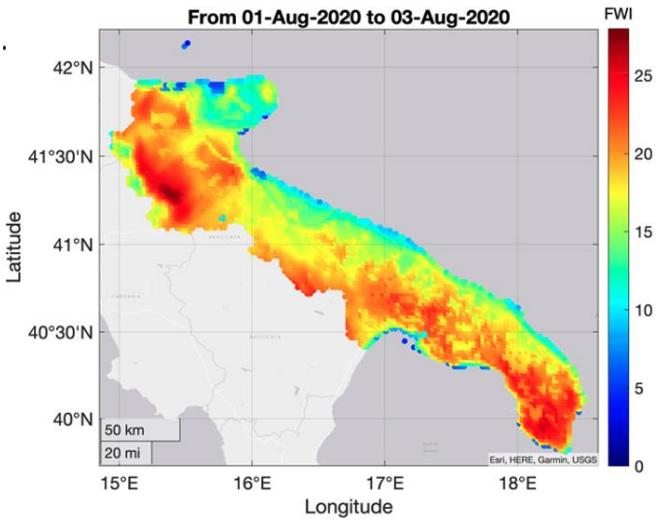



**Figure 33 - FWI time-averaged over 72h simulation (WRF initialized by HRES Forecast) from 01/08/2020 to 03/08/2020.**

### 3.1.2. Seasonal Forecast & Probabilistic Fire Weather Index

To better assess the seasonal forecast products for the development of a service for a probabilistic Fire Weather Index (FWI), we have a proceed with the following methodology:

1. Studying the relationship between the FWI and the large-scale patterns in the atmosphere (Figure 34) [43].
2. Perform a downscaling of the members of the model and a bias correction to increase the resolution of the model and correct the biases of the model with respect to the reference dataset ERA5.
3. Performing a subsampling of the members within the ensemble of the seasonal forecast model to select the members that have a better representation of the FWI and better representation of the large-scale pattern studied in step 1.
4. Computation of the terciles and probabilities of the FWI in seasonal forecast.
5. Computation of the probabilities of fire danger indicators based on the computed FWI.

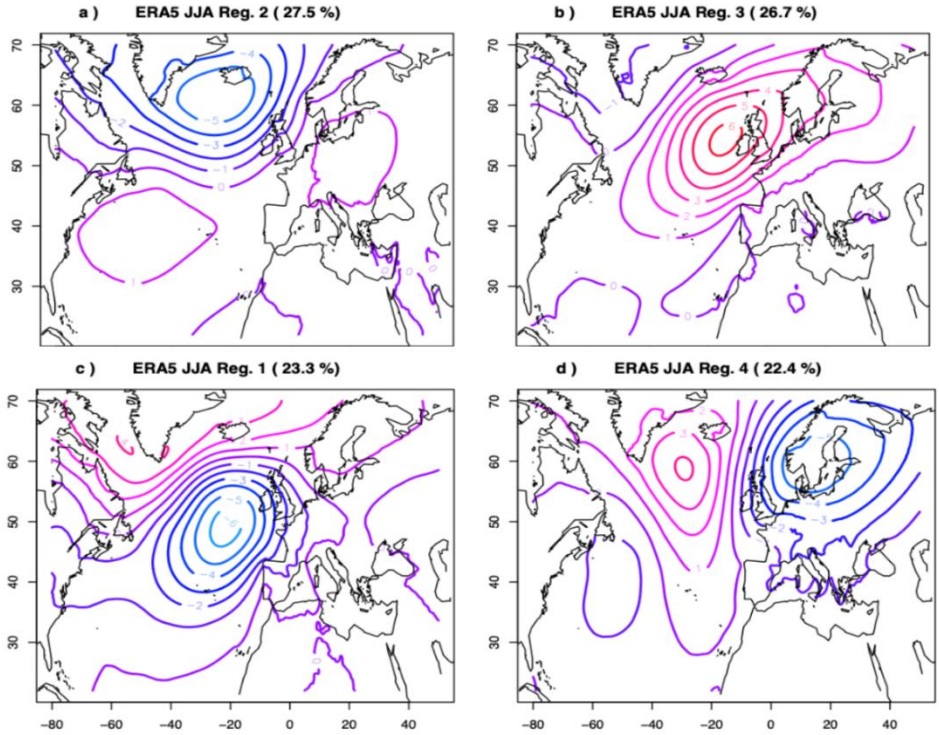



**Figure 34 - Large-scale patterns computed in step 1 of the methodology for the season June-July-August (JJA) during the period of hindcast (1993-2016). The regime a) is the North Atlantic Oscillation in its positive phase (NAO+), b) is the Scandinavian blocking pattern, c) is the Atlantic low pattern and d) is a pattern related to North Atlantic circulation in its negative phase (NAO-). These patterns have been computed daily and used for the subsampling of the members. (Steps 1 and 3 of the methodology)**

The model of seasonal forecast used has been the CMCC SPS3.5 model with 40 members of hindcast (1993-2016) and 0.5x0.5 degrees of resolution and 50 members of forecast (2017-2022) with the same resolution. The resolution has been increased to 0.25x0.25 degrees since we are performing a downscaling (step 2 of the methodology) with ERA5 as a reference dataset.

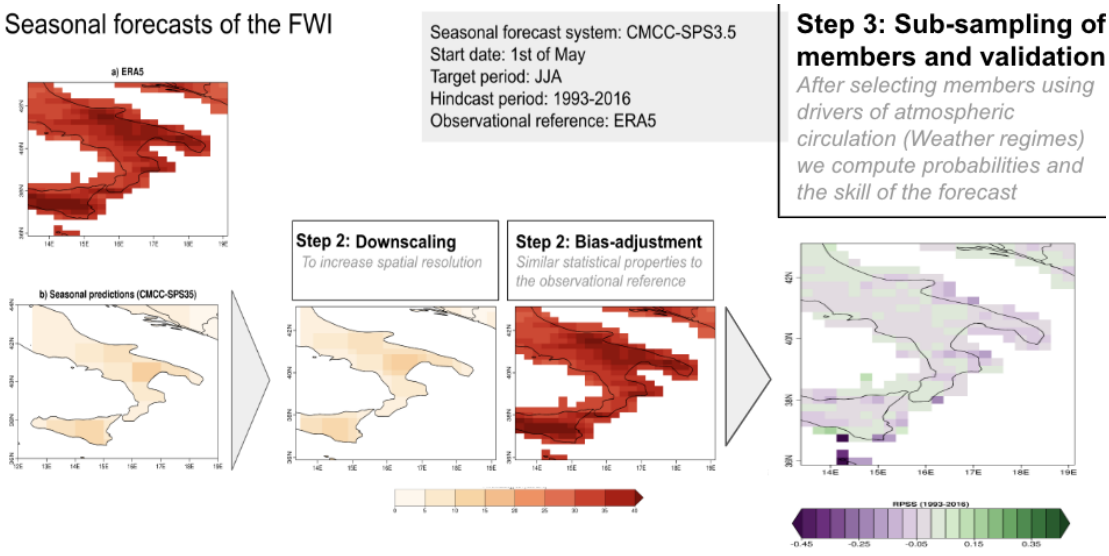



**Figure 35 - Example of steps 2 of Downscaling, bias adjustment and 3 for the subsampling of the members using the CMCC seasonal forecast model for Southern Italy in JJA.**

Once the subsampling of members is performed, we compute the terciles of seasonal forecast and compute the probabilities of the fire danger indicators following the scale of danger of the European Commission and Copernicus [44] and described as follows:

- Very low: <5.2
- Low: 5.2 - 11.2
- Moderate: 11.2 - 21.3
- High: 21.3 - 38.0
- Very high: 38.0 - 50
- Extreme: >=50.0

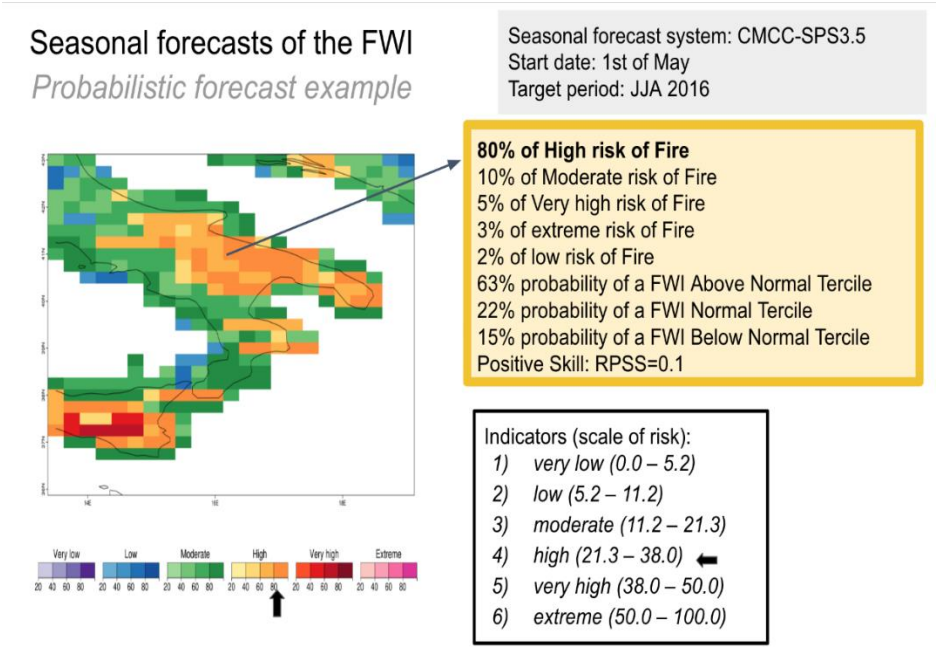



**Figure 36 - Probabilities of Fire danger indicators in southern Italy. Colors show the most likely category within the ensemble. Lighter colors mean lower probabilities and darker higher probabilities. Example of the kind of information available in the prototype for 1 specific summer and further information (yellow box with text) available for each grid point of resolution.**

### 3.2. Ingestion

The short-term FWI and probabilistic FWI, together with the short-term and seasonal forecast data have been ingested in the CMCC Data Delivery System [46]. The Apache NiFi ingestion pipelines will be developed in order to ingest the data in the SILVANUS Platform and to store the data in the SAL.

## 4. In-situ Devices

### 4.1. IoT Devices and Networking

#### 4.1.1. Portuguese Pilot Site

LoRa network [47] is a wireless communication technology designed for the IoT applications. It stands for Long Range and is a low-power, low-data-rate, and long-range wireless protocol and is the best choice to allows devices to communicate wirelessly over long distances with

minimal battery consumption. The technology uses a spread-spectrum modulation technique that allows for long-range communication with minimal interference. It operates in the unlicensed radio frequency bands and can transmit data up to several kilometers in open areas. The technology uses low-power transceivers that can operate on batteries for years, making it suitable for remote, low-maintenance applications.

LoRa network consists of three components: end devices, gateways, and network servers. The end devices are sensors or devices that collect data and transmit information to the gateway. The gateway receives data from the end devices and forwards it to a network server, which can be used to analyze the data, generate alerts and trigger actions based on predefined rules. The network server manages the communication between the gateways and the end devices; it also provides integration with other networks or applications. For example, the system may be configurated to alert when smoke sensors rise above a certain threshold, allowing to take automatic actions such as notifications to farm managers or emergency authorities on a smoke event alert, or sending a message to mobile devices or computers.

At the Portuguese pilot site Quinta da França farm (QF) a LoRa network has been installed and is operating since 2020, covering the entire farm and communicating with several IoT devices available in the field, collecting local data. Also, a local weather station and a private video surveillance system are installed in QF. All these devices and gateways are part of the QF IP and LoRa network, as shown in the schematic Figure 37.

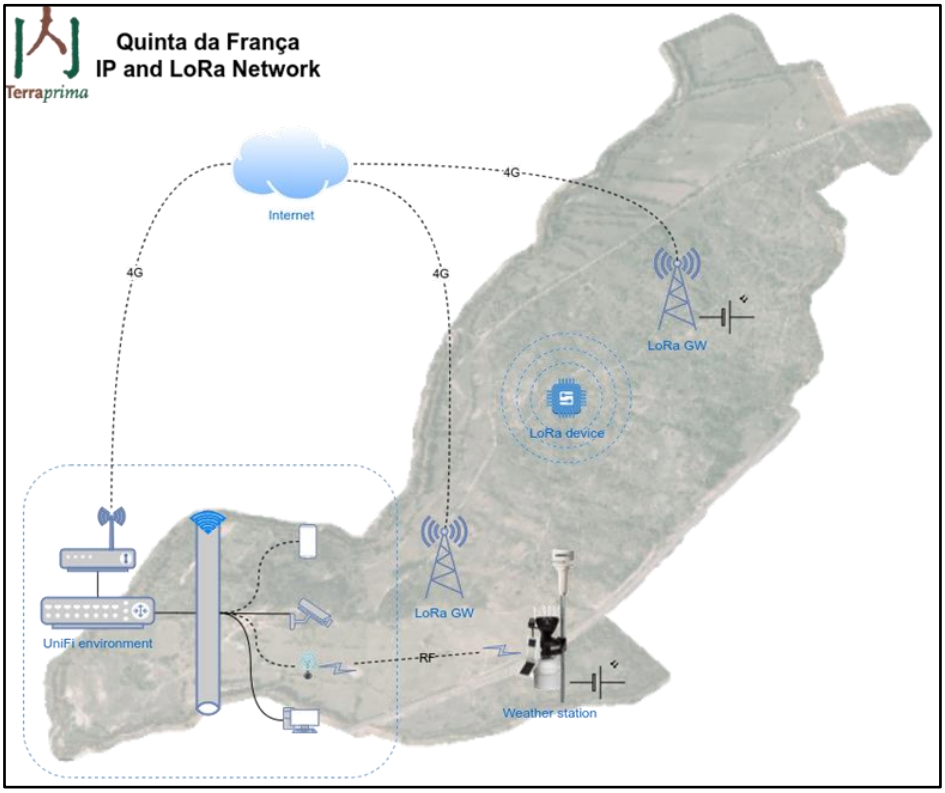



**Figure 37 - Portuguese pilot farm (QF) IP and LoRa network schema.**

Some of the IoTs devices operating in QF farm are soil sensors – able to measure moisture, temperature and electrical conductivity - and GPS animal collars (Figure 38). Those IoT devices are powered by battery or solar panels and have been collecting data in the agroforestry parcels at QF (Figure 39).

**Soil Sensors**

**GPS Animal Collars**

**Figure 38 – Example of IoTs devices at QF farm – Soil sensors (above) and GPS animal collars (below)**

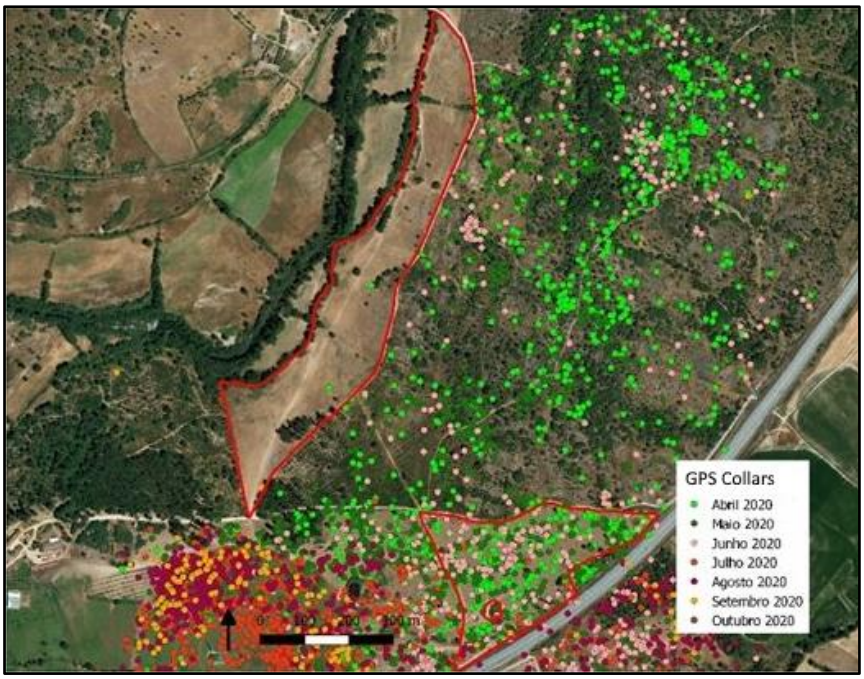



**Figure 39 - Example of data collection from an IoT device at QF farm - GPS animal collars.**

In QF there are other type of IoT devices in an experimental phase to be tested in the field, namely, a particle sensor (configured to be a smoke detector) and thermal cameras.

All these sensors devices are monitoring the local environment and the QF farm activity, contributing to R&D projects, such as SILVANUS, for fire prevention and early warning fire detection. The data collected from the IoTs are sent to the QF´s LoRa gateways and then are stored in a private local storage, and in a dedicated cloud storage that can be accessed via desktop/mobile or application clients (Figure 40).

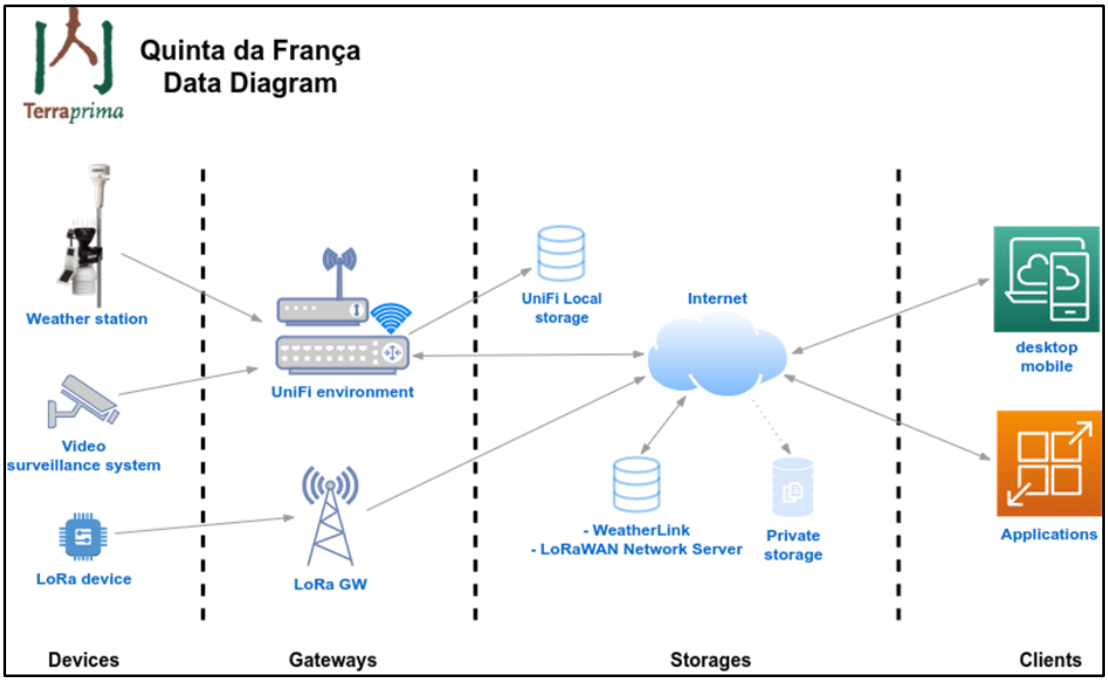



**Figure 40 - QF data flow diagram illustrates network architecture and data flow diagram**

### 4.1.2. IoT Edge Device with Raspberry Pi

The idea behind the development of the IoT Edge Device (IoTED), that will be used in SILVANUS, is to create fairly cheap, easily reproducible devices that will operate on the edge and serve as an early fire detection mechanism. Therefore, such devices should be able to communicate with SILVANUS servers from rural areas (e.g., forests), be power supply independent (as electricity will not be always available) and 'smart' to detect fires as soon as possible. For this reason, an IoTED has been designed to accommodate such objectives and be able to process collected data for the fire detection.

Specifically, the IoTED consists of a waterproof case [47] (with a sturdy strap for its fastening on trees, see Figure 41-b) and encloses widely used sensors and components such as:

- Temperature and humidity sensor (DHT22 [48])
- Smoke detection sensor (MQ-2 [49])
- Raspberry Pi 4 Model B [50]
- RPi RGB camera [51]
- 10,000 mAh power bank
- GSM/LTE adaptor (A7670E_Cat-1_HAT [52])

Sensors and cameras are used for the continuous check of the surrounding area's status, but the captured image will be further processed by the pre-trained Machine Learning (ML) algorithms employed on the Raspberry Pi (RPi). The deployed ML algorithms are described in Section 4.4, and their goal is to detect fire (and in the future smoke) on the edge, thus informing the SILVANUS framework of the new emergency immediately. The informing of the framework will be done through a POST request from the RPi's GSM/LTE adaptor to NiFi (for more details see Sections 4.2 and 4.3). The power bank will be used to power the entire device and allows it to operate on remote areas where we have no access to electricity. The current setting with a power bank of 10,000 mAh allows the device to operate between 8 to 10 hours, depending on the frequency of capturing and transferring data. It is in our future plans to include a solar panel for recharging the battery, thus making the device more power independent.

The device's ability for early fire detection, as well as for processing the acquired data on the edge and having power supply independency, is what makes it a powerful added value to the SILVANUS Platform. Specially, the RPi's ability for processing images on the spot makes the device a powerful supplementary tool to first responders and forest/park monitoring organisations. Upon completion and deployment of the smoke detection ML algorithms, the

device will be an even better ally for fire detection, being smoke an early tale-tell of a fire helping to draw attention for a possible fire event even sooner.


(a)

(b)

**Figure 41 (a) - Example scenario of a forest monitoring with the use of the IoTED and (b) the fastening of the device on a tree's trunk.**

A possible use case scenario of IoTED deployment is shown in Figure 41 (a), where a number of these devices (represented by blue rectangles) have been placed on several trees to adequately cover the surface of the forest. Each device will monitor its surroundings and occasionally (e.g., hourly) send a report with the collected data (i.e., temperature, humidity, captured images) to the SILVANUS Platform to be finally viewed by the end users.

These periodical reports can help with the dynamic monitoring of remote areas, the identification of fire-prone areas and most importantly can create a historical database, that can be used later for the development or for feeding other monitoring/preventive models (e.g., fire probability based on sensor data). Furthermore, the fire detection module (based on ML) will constantly check the captured images (on the edge) for possible fire occurrences

and in case of a positive event a fire warning will be generated in the system (that can be viewed on SILVANUS' User Interface) to alert end users. Upon this fire event, the IoTED will provide more regular updates (e.g., every 10 minutes) on the status of the area, to make its monitoring more efficient keeping track of the fire's progress.

Additionally, because of the multiple devices spread within the area of interest, the direction and speed of fire can be tracked, depending on the time the neighboring devices have triggered a similar alert. This can therefore help with the assessment of a fire's severity, especially in multi-location concurrent events, improving the use and allocation of a station's resources (e.g., firefighters and fire tracks).

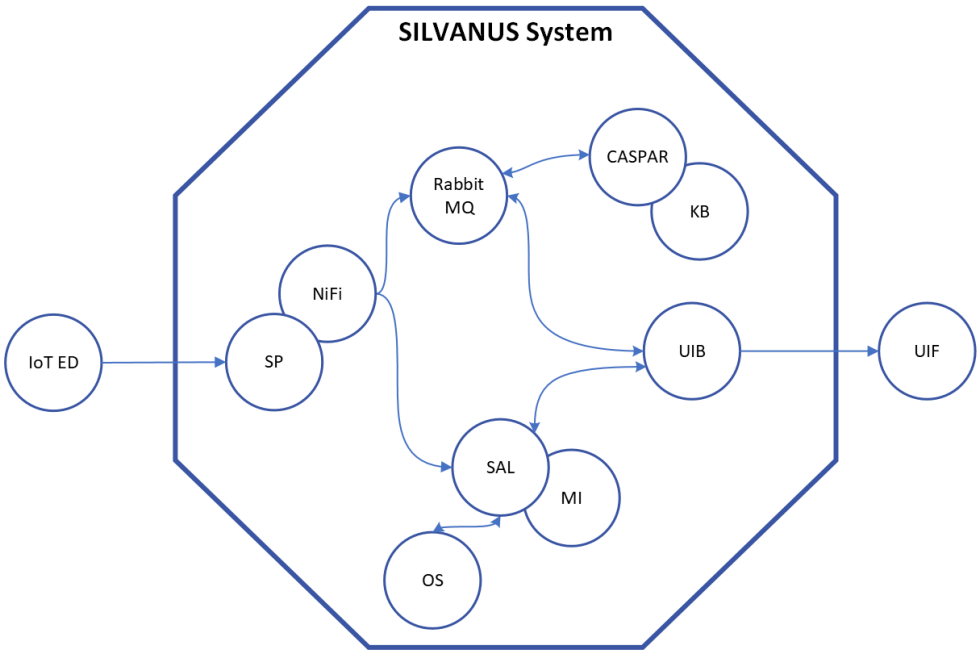
In the following sections, we will present the Apache NiFi data ingestion flow (and message types) from the perspective of the IoTED and the SILVANUS Platform, the data pre-processing done on the edge and finally the different pilot sites where the devices will be tested.

### 4.2. Ingestion Flows

An overview of the IoTED data ingestion flow is shown in Figure 42. The IoTED and the User Interface Frontend (UIF) are not modules of the SILVANUS Platform, but rather external components. The IoTED send the message (data +metadata) to the Security Proxy and NiFi to be validated. Then, upon validation, the data and metadata will be sent to SILVANUS SAL and stored respectuvely to the Object Store and Metadata Index. In parallel, NiFi broadcasts the necessary messages to the message bus (Rabbit MQ), that in turn notifies the SILVANUS components that are subscribed to that topic. More details about the data ingestion pipeline, their storing and the message broker can be found in Section 5.

As shown in Figure 42, the SILVANUS's components that will digest the data provided by the IoTED are

- the UI – for the visualisation of the detected fire events/IoTED status on the map,
- the CASPAR/Knowledge Base – for the extraction of fire related knowledge.



| Acronym | IoT ED | SP | SAL | MI | OS | KB | UIB/F |
|---|---|---|---|---|---|---|---|
| Meaning | IoT Edge Device | Security Proxy | Storage Abstraction Layer | Metadata Index | Object Store | Knowledge Graph | User Interface Backend/ Frontend |

**Figure 42 - Diagram of IoTED ingestion flow in SILVANUS system**

As mentioned in the previous section, the IoTED regularly collects data from the sensors and captures images of its surroundings for the monitoring of the area of interest. Then it sends the collected data to the SILVANUS Platform for storing and sharing with the interested modules, on several occasions such as:

1. Hourly status report of sensor readings and image
2. Possible fire event (occurs per identified incident)

In the first case, the communication with NiFi is done based on a scheduled set of times (i.e., hourly) and its main purpose is to update the system's data, for databasing purposes and/or their processing by other modules. It is important to mention that images will be sent only during sunlight (i.e., JSON image field will be set to an empty string) because the camera is not equipped with night vision or lights, therefore any captured images would be too dark and of no use to the partners.

In the other case though, the communication happens on demand, meaning that whenever the IoTED identifies a possible fire event (see Section 4.4) it sends a warning message to the system, in order to trigger the corresponding modules for the necessary actions to be taken (e.g., informing of firefighters about the new fire through the UI). The fire event messages will keep occurring every 10 minutes and while the fire is still burning (or in range of the IoTED's camera) to keep the end users updated about its progress.

Regarding the data sent by each type of message, they both send a JSON file with the same fields and the metadata JSON file. Their main difference, besides the frequency of their posting, is in the "visual_data" field, because the possible fire events will have a higher "fire_probability" value, which categorises them as a fire event and triggers a different message queue in the system. In this way, the systems will immediately be informed on the new emergency and take the necessary actions.

A list of the fields used in the data JSON file's body, along with their data types and a brief field explanation, can be found in Table 10.

**Table 10 - Description of fields used in IoTED's data JSON file**

| Field | Type | Description | Example value |
|---|---|---|---|
| uuid | String | Unique identifier of the message (created using uuid version 7), that will be used to connect the data file with the metadata file. This field will also be used to match the NiFi responses to the POST request. | "0186e4a2-a4ae-76d5-a08d-b371a130bbea" |

| | | | |
|---|---|---|---|
| sensor_type | List of strings | List with the sensors equipped on the Raspberry Pi | ["camera", "temperature", "humidity", "smoke_gas"] |
| range | JSON object array | The range (distance) that is covered by the Raspberry Pi. The attribute “value” stands for the distance that it covers, and “unit” for the measurement unit of the distance (e.g., meters) | {"value": 5, "unit": "meters"} |
| timestamp | String | Datetime the message was created. It follows the ISO 8601 format (“YYYY-MM-DDTHH:MM:SS.ffffff”) with the letter ‘Z’ at the end to show that it is in UTC timezone. | "2023-03-11T12:10:13.490166Z" |
| location | JSON object array | The location of the RPi, represented as JSON object with attributes “placename” and “geometry” | "location": [{ "placename": "Cyprus", ”geometry": { "type": "Point", "coordinates": [{ “lat": 35.151688, "lon": 33.350244 }]}}] |
| placename | string | The name of the location (country/city) or pilot site name | “Cyprus” |
| geometry | JSON object | A JSON object with attributes “type” and “coordinates” | |
| type | String | The type of geometry to be described. It is always set to “Point” | “Point” |
| coordinates | JSON object array | The coordinates of the RPi’s location, with attributes “lat” and “lon” | "coordinates": [{ “lat": 35.151688, "lon": 33.350244 }] |
| lat | Double | The latitude coordinates | 35.151688 |
| lon | Double | The longtitude coordinates | 33.350244 |
| sensory_data | JSON object | A JSON object with sensor readings | "sensory_data":{ "temperature": {"value": 22.6, "unit": "celsius"}, |

| | | | |
|---|---|---|---|
| | | | "humidity": { "value": 64.3, "unit": "percentage"}, "gas_sensor": { "smoke": 19.57, "liquid_petrolium_gas(lpg)": 4.26, "methane": 11.38, "hydrogen": 8.47, "unit": "parts_per_million(ppm)"}} |
| temperature | JSON object | The temperature captured by the RPi sensor, with attributes “value” and “unit”. | temperature": {"value": 22.6, "unit": "celsius"} |
| humidity | JSON object | The humidity captured by the RPi sensor, with attributes “value” and “unit”. | humidity": { "value": 64.3, "unit": "percentage"} |
| gas_sensor | JSON object | The gas sensor values captured by the RPi, with attributes “smoke”, ”liquid_petrolium_gas(lpg)", “methane” and “hydrogen”, which stand for the different gas the sensor measures, and lastly, the “unit” attribute, which shows the measurement unit of the sensor. | "gas_sensor": { "smoke": 19.57, "liquid_petrolium_gas(lpg)": 4.26, "methane": 11.38, "hydrogen": 8.47, "unit": "parts_per_million(ppm)"} |
| visual_data | JSON object | A JSON object with the ML algorithms results and the image captured by the RPi’s camera. | "visual_data": { "contains_fire": true, "fire_score": 0.8746, "fire_probability": "high",”image": base64_image_encoding(string)} |
| contains_fire | Boolean | Indicates whether the captured image contains fire or not | "contains_fire": true |
| fire_score | Double | The fire detection ML algorithm’s prediction probability for the captured image. Values range between 0 and 1. | "fire_score": 0.8746 |
| fire_probability | String | Characterises the confidence of the predictions of the ML model. For small fire_scores (<0.5) the fire_probability is "low", between 0.5 and 0.7 is "medium" and for larger values, it is "high". | "fire_probability": "high" |

| | | | |
|---|---|---|---|
| image | String | The base64 encoding of the captured image | The example is too large (several thousands of characters) to show |

It is important to mention that the IoT data JSON file needs to be compatible with the CASPAR framework, as it will be used to feed the Knowledge Base for the creation of fire-related queries.

In the next section, the data ingestion flow will be presented from NiFi's perspective.

### 4.3. Apache NiFi Ingestion Pipeline for IoT data

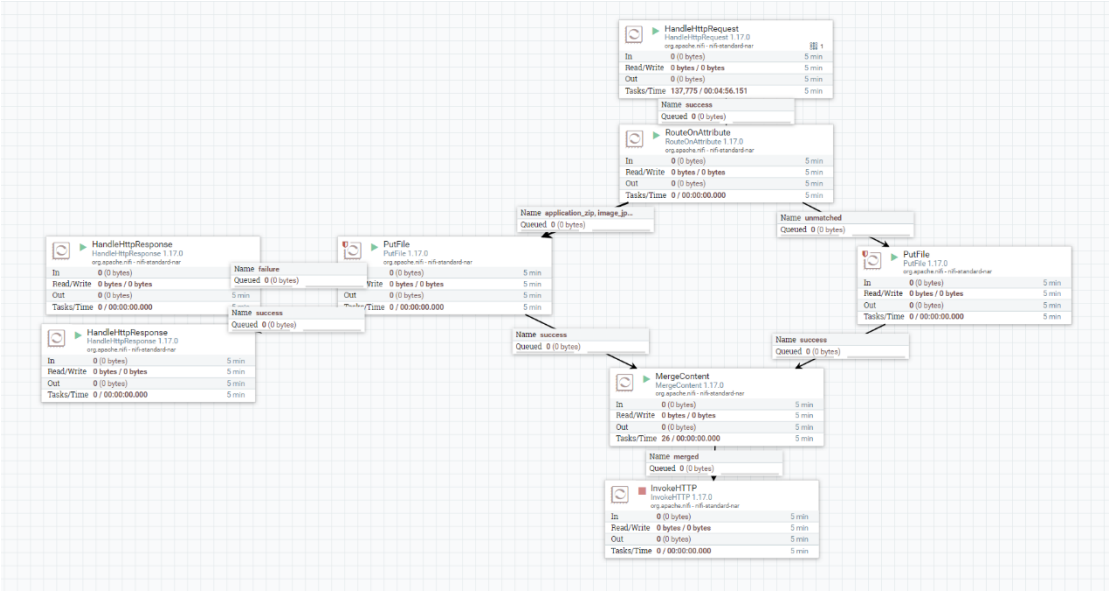



**Figure 43 - Initial Apache Nifi pipeline for IoT data ingestion**

Figure 43 shows the Apache NiFi data flow for ingesting IoT data from far-edge devices. The ingestion mechanism for devices deployed to the far-edge (IoT, UAV, UGV) is different from data ingestion described in the previous sections (i.e. EO, Weather/Climate) since IoT data is pushed from data sources to the SILVANUS platform.

The data ingestion pipeline exposes a HTTP Endpoint that acts as the ingestion interface for IoT data sources. This endpoint accepts HTTP Multi-part/form formatted requests with two data objects attached: the JSON data object and JSON metadata descriptor. The JSON metadata descriptor describes the relevant data object of the request and is leveraged in the pre-processing process and later also in the indexing and for the retrieval from the SILVANUS SAL.

**Table 11 - Building HTTP request using CURL**

```
curl --request POST \
  --url http://10.12.1.2:9004/ \
  --header 'Content-Type: multipart/form-data' \
  --form data=@C:\data-sample.json \
  --form 'metadata=@C:\metadata.json'
```

Regarding the data flow through the data ingestion pipeline, after receiving HTTP message containing the above objects, we initially split the data object and metadata processing.

- **Data Object:** the data object is temporarily stored, assuming a valid request has been received, a response is generated confirming ingestion. At this stage we can integrate any pre/post-processing routines required (e.g., Fire/Smoke detection).
- **Metadata Descriptor:** the metadata descriptor is handled in a separate flow to preserve the original data object metadata. A key consideration is the output of this metadata and attachment to original object within the Merge Content Process. Any additional pre/post-processing routines generate a new data object, and as such should provide a relevant metadata descriptor.

In the final step of this pipeline, we connect with the SILVANUS SAL through a POST HTTP request (see Table 11), transmitting the data object via the message body and metadata as HTTP Headers.

### 4.4. Pre-processing

As previously mentioned, the IoT devices will periodically collect images and sensor data (e.g., temperature) from its surrounding area for storing in SAL. Both data sources can serve as historical databases for monitoring the environmental changes of a site, and also as input for other modules (e.g., Knowledge Graph). Currently, the sensor data are only collected on the device and propagated to NiFi for ingestion, without further processing, but images undergo a pre-processing step. Specifically, captured images are processed by the fire detection algorithm, and in the future by the smoke detection one, to determine if they contain a possible fire event.

**Figure 44 - (left) Example output of the fire binary classification model and (right) an output example of the fire superpixel localisation algorithm.**

In detail, the Raspberry Pi (RPi) is equipped with a lightweight ML model that was pre-trained to detect and localize fires within images (details for the training of the algorithm can be found in Deliverable D4.2 Section 5.2.1.4 AI model training). An example output of the trained fire classification and localization models can be seen in Figure 44. The ML model is responsible for processing all captured images and determining whether they depict a fire or not. In the case of a fire, a special type of message (as described in Section 4.2) is generated (see Figure 45) that is sent to NiFi, which in turn will immediately inform (as described in Section 5.2) the interested modules (e.g., UI) of the new emergency to act accordingly.

```
{
  "uuid":"0186e4a2-a4ae-76d5-a08d-b371a130bbea",
  "sensor_type":[
    "camera",
    "temperature",
    "humidity",
    "smoke_gas"
  ],
  "range":{
    "value":5,
    "unit":"meters"
  },
  "timestamp":"2023-03-11T12:10:13.490166Z",
  "location":[
    {
      "placename":"Cyprus",
      "geometry":{
        "type":"Point",
        "coordinates":[
          {
            "lat":35.151688,
            "lon":33.350244
          }
        ]
      }
    }
  ],
  "sensory_data":{
    "temperature":{
      "value":22.6,
      "unit":"celsius"
    },
    "humidity":{
      "value":64.3,
      "unit":"percentage"
    },
    "gas_sensor":{
      "smoke":19.57,
      "liquid_petrolium_gas(lpg)":4.26,
      "methane":11.38,
      "hydrogen":8.47,
      "unit":"parts_per_million(ppm)"
    }
  },
  "visual_data":{
    "contains_fire":true,
    "fire_score":0.8746,
    "fire_probability":"high",
    "image":"base64 encoding of captured image"
  }
}
```


(a)

(b)

```
{
  "descriptor": {
    "uuid":"0186e4a2-a4ae-76d5-a08d-b371a130bbea",
    "obj-class":"IoT",
    "format": {
      "type":"json"
    },
    "access":"dummy-pilot",
    "dataset-type":"image-sensory",
    "created":"1678720252.864227"
  },
  "spatial": {
    "coordinates": {
      "lat":35.151688,
      "lon":33.350244
    },
    "pilot":"cyprus"
  },
  "temporal": {
    "datetime":"1678720252.864227"
  },
  "tag": {
    "device":"Raspberry Pi",
    "sensors": [
      "camera",
      "temperature",
      "humidity",
      "smoke_gas"
    ],
    "fire_related":true
  }
}
```


(c)

**Figure 45 - (a) Example JSON file of fire event, (b) the corresponding captured image (that will be included in the JSON file in base64 format) and (c) the corresponding canonical metadata JSON file for the event.**

Regarding the other types of images (i.e. non-fire ones), not all of them will be sent to the ingestion pipeline to limit the data traffic in the system. Only a small fraction of the non-fire images will be sent to be stored, along with the corresponding sensor data, as described previously in Section 4.2.

It is worth to mention, that the same ML model as the one deployed in RPi, will be used as for the identification of possible fire events from other sources providing images such as UAVs and UGVs and the corresponding images will be sent to the data ingestion pipeline

### 4.5. Fire Detection Demonstration

SILVANUS' pilot sites demonstrations will take place after the submission of this deliverable, therefore here we will mention in which pilot sites the IoTED will participate, and which components will be tested in each.

Specifically, the IoTED will be used in three (3) official pilot sites in Šapjane in Croatia -, Tepilora Park in Italy –and finally, in Euboea Greece. Currently, only the Croatian pilot site dates are known (18-19$^{th}$ of April) as it is the first pilot that will take place in SILVANUS.

In Šapjane, a controlled fire will be set to test various components developed within SILVANUS and one of them will be the IoTED's fire detection algorithm. Since it will be the only pilot (at the time) that will light a fire, it will be a great chance to test the accuracy of the ML fire detection algorithm and collect data from a real-life case scenario. This data can later be fed to other components, such as the Knowledge Graph, for the extraction of fire related knowledge (e.g., how much does temperature/humidity affect the fire). Simultaneously with the ML algorithm's accuracy, we will test its sensitivity based on the device's distance from the fire, i.e., how the distance affects the algorithm's confidence (fire probability) in detecting a fire. Along with the ML algorithm validation, we will test the communication of the IoTED with the server for storing the collected data. As the SILVANUS Platform is still under development, we will not be able to test the pipeline and view the collected data. For this reason, CTL developed a simple UI (see Figure 46) that shows the latest collected data (i.e., image and sensor readings) to ensure that data are being collected and the device operates as expected during the Croatian pilot site. Finally, and most importantly, we will be able to check the power supply independency of the IoTED and monitor the power consumption in a fully operating scenario. This will help with the selection of the correct size/power of the solar panels that will be used for the recharging of the IoTED's power bank to make the device fully (energy) independent for future pilots.

The Croatian pilot site will also serve to get feedback from actual potential users (i.e., Croatian Firefighting Association) and better prepare for the following pilots, that will happen at a later stage where the whole SILVANUS Platform will be deployed.

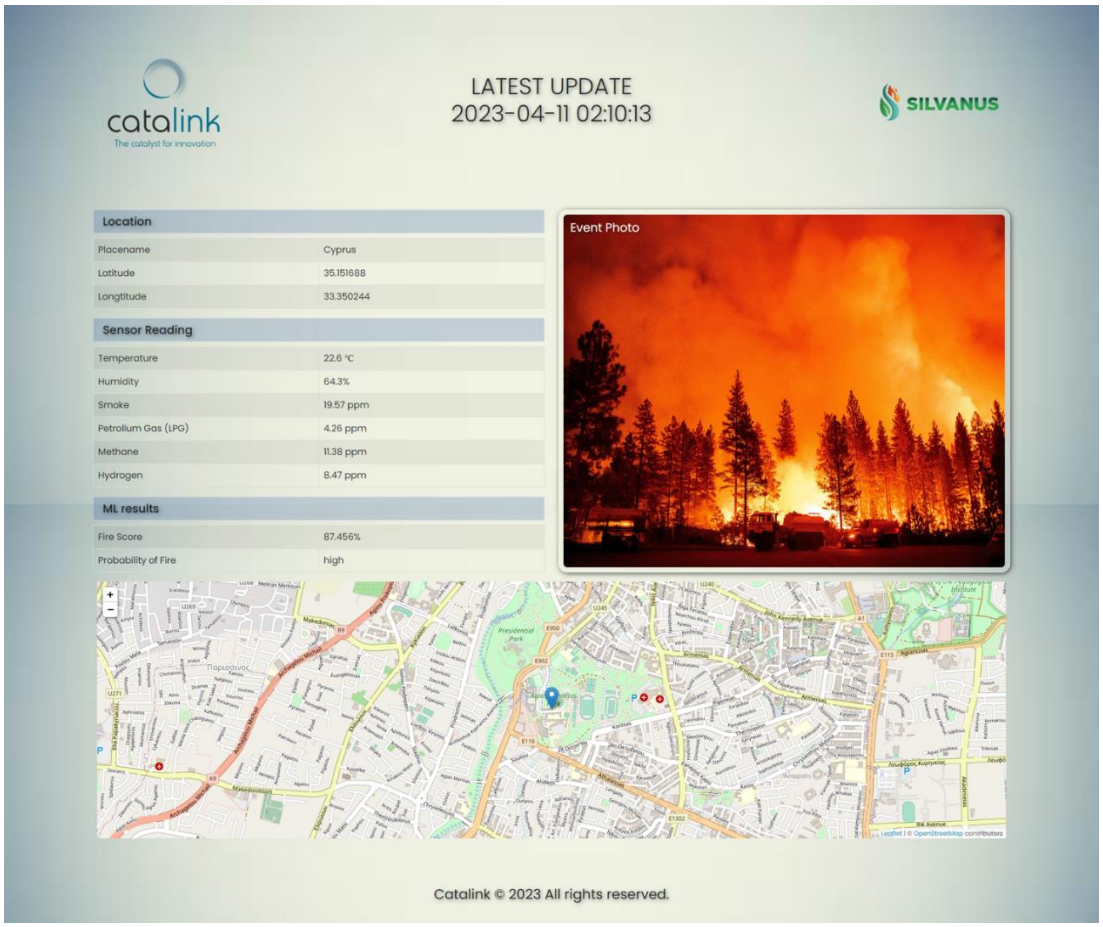



**Figure 46 – UI developed for the purposes of view the collected data by the IoTED.**

The Greek and Italian pilot sites are not finalized yet, but they will probably happen sometime in autumn and summer of 2023, respectively. Both sites will be used collect data, that will be pushed to the SILVANUS framework (through NiFi), as the project will have significantly progressed by then. Furthermore, a version of the smoke detection ML model will be available (trained) by then, and it will be tested with the use of some sort of smoke machine.

An updated description of the pilot sites results/feedback will be provided in future deliverables, such as deliverable D4.5 (M36) "Report on SILVANUS advanced detection capabilities".

# 5. Integration with SILVANUS Platform

This section presents a brief overview of the integration between the Ingestion Pipelines related to the different data sources described in the previous sections and the SILVANUS platform.

## 5.1. Data Ingestion Pipeline

As presented in Figure 1, Apache Nifi acts as the common data ingestion pipeline into the SILVANUS platform for a wide variety of data sources. As defined within the SILVANUS reference architecture (Deliverable D8.1), this component directly integrates with the Big-data framework and data storage solution deployed to the platform.

Figure 1 depicts the flow of data objects through the Data Ingestion Pipeline, as data is ingested and moves through the corresponding Nifi pipeline. The format, size, ingestion frequency and structure of this data can vary depending on the provider. As the specific implementation of each pipeline can vary, however the flow and high-level operations remain consistent. Metadata extraction of data objects at ingestion time is one of these key processes that takes place at this stage, as this supports the data annotation and later retrieval of data objects from the Big-data framework.

The Data Ingestion Pipeline directly communicates via a REST API with the SILVANUS Storage Abstraction Layer (SAL), an abstraction component of the Big-data Framework and underlying object storage solution. Exposed by the SAL is a single abstract POST HTTP endpoint that is responsible for ingesting all data sources from the Data Ingestion Pipeline. The output HTTP message from the Ingestion Pipeline follows a consistent format containing:

- **HTTP Body – Object data**: A single data object encoded as the HTTP message body
- **Headers – Object metadata, Ingestion Pipeline Attributes:** Extracted object metadata is attached to the same HTTP request, following the specification and format described in Section 2.2.1. Metadata generated by the specific ingestion pipeline is also attached, these are the Apache Nifi flowfile attributes generated from the pipeline processors, including the initiating queue, dataset ingestion parameters etc. This allows further enrichment of metadata provided alongside data objects to be leveraged in the SAL.

## 5.2. Message Bus

The ingestion of individual datasets is managed by a series of corresponding data pipelines. As presented in Deliverable D8.1, we implement a communication solution based on asynchronies message queue, the SILVANUS Message Bus. As part of the MVP and

demonstration of the Data Ingestion Pipeline framework, we implement a series of initial sample queues which map directly to a corresponding data pipeline.

As each ingestion queue corresponds to an individual data product, each pipeline implements a corresponding MQTT listener, which awaits new messages published and in turn initiates the ingestion of a data product. Messages are published two an ingestion queue via primary mechanisms.

1. **Static Product Ingestion:** This category defines datasets which have statically defined parameters for ingestion. Parameters for ingestion refer to attributes such as the geospatial area, ingestion frequency, data format or temporal range. For some data sources we need to only consider a statically defined set of parameters (and therefor ingestion messages) which can be automated and ingested based on these requirements. This ingestion mechanism is further detailed in Section 5.3
2. **Custom Product ingestion:** Custom dataset parameters are a requirement of some datasets, specifically the user products that leverage these datasets within AI/ML training / inference and visualization dashboards. An example of this requirement is presented in Section 0.

In order to ingest these custom datasets a user product can publish a message to the relevant message queue, containing the specific ingestion and pre-processing parameters that are required. The current features and format of these ingestion messages is summarized in Table 12. It reflects the current status and pre-processing capabilities within the Data Ingestion Pipeline for demonstration purposes and can be expanded as new data source providers are integrated and additional pre/post-processing techniques become available.

**Table 12 - Status and Pre-processing Capabilities Within the Data Ingestion Pipeline**

| Queue Name | Dataset | Parameters | Output |
|---|---|---|---|
| ingest.dem | Digital Elevation Model | pilot: [*pilot_string]<br>type: [dem, asp, slp] | Tiff |
| ingest.osm | OpenStreetMaps Road / Rail | Pilot: [*pilot_string]<br>type: [road, rail]<br>resolution: [*Int]<br>bbox: [*GeoJSON_coords] | GeoJSON, NetCDF |
| ingest.sentinel-ndvi | Sentinel-2/3 + NDVI | resolution: [*Int]<br>footprint: [*GeoJSON_bbox]<br>cloud: [*Int] | SAFE, Tiff |
| ingest.sentinel | Sentinel 2/3 | | SAFE |
| ingest.lst | Land Surface Temperature | type: - [H, DC, TCI] | NetCDF |
| ingest.ba | Burned Area | version: - [V1, V3] | NetCDF |
| ingest.pop | Population Density | pop_year: - [*YYYY]<br>country: - [*ISO 3166 code] | CSV, Tiff |
| ingest.stf | Short-term Forecast | TBD | NetCDF |

| | | | |
|---|---|---|---|
| ingest.sef | Seasonal Forecast | TBD | NetCDF |
| Ingest.clc | Corine Landcover | TBD | Tiff |

The implemented architecture supports the SILVANUS Platform with two key properties:

- **Loose coupling of individual components** – supporting cases where a connection link between two components may be lost temporarily.
- **Common communication framework** – supporting key functionalities such as request of custom datasets from user products and notification of dataset ingestion status and availability to relevant user products.

### 5.3. Data Pipeline Initiator

The pipeline initiator is a stand-alone component that will publish RabbitMQ messages, to initiate a Nifi flow for the ingestion of data. This application runs continuously inside a Docker container and sends messages according to a configuration file. For each trigger, the application creates a thread that send a RabbitMQ message (queue, body) at specified trigger time.

#### 5.3.1. Config file

This is an extendable file (JSON), loaded by the python application which defines an array of triggers. A trigger must contain 3 values presented in Table 13.

**Table 13 - Trigger Values**

| Key | Type | Description |
|---|---|---|
| queue | string | RMQ queue reference – e.g. “ingest.sentinel” |
| message_body | dictionary | JSON message body, defines ingestion params – e.g. pilot, spatial |
| initiator | dictionary | Defines the initiation parameter. |

The dictionary associated with the key "initiator" must be defined as reported in Table 14:

**Table 14 - Initiator Definition**

| Key | Type | Description |
|---|---|---|
| cron | string | Defines a schedule using the unix-cron string format ( * * * * * ) |

An example of config.json can be found in the file config.example.json. In that configuration, two triggers are defined:

- The first (with queue `queue_name_every_minute`) is triggered every minute.
- The second (with queue `queue_name_every_five_minutes`) is triggered every five minutes.

### 5.4. Cron string format

The unix-cron string format is a set of five fields in a row. Figure 47*shows* the meaning of each field.

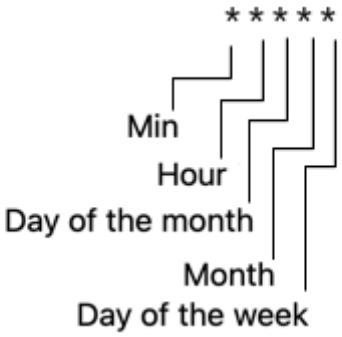



**Figure 47 - The unix-cron string structure.**

The time fields have the following format and possible values, and must follow the format presented in Table 15.

**Table 15 - Time Fields Format and Values**

| Field | Format of valid values |
|---|---|
| Minute | 0-59 |
| Hour | 0-23 |
| Day of the month | 1-31 |
| Month | 1-12 (or JAN to DEC) |
| Day of the week | 0-6 (or SUN to SAT; or 7 for Sunday) |

Each field may have special characters in addition to the previously mentioned values:

- Asterisk (`*`) stands for "every" (every month, every hour, every week...).
- Ranges are two numbers separated with a hyphen (`-`) and the specified range is inclusive.
- Following a range with `/NUMBER` specifies skips of the number value through the range. For example, both `0-23/2` and `*/2` can be used in the `Hour` field to specify execution every two hours.
- A list is a set of numbers (or ranges) separated by commas (`,`). For example, `1,2,5,6` in the `Day of the month` field specifies an execution on the first, second, fifth, and sixth days of the month.

#### 5.4.1. Cron string examples

- `0 * * * *` run always (every minute).
- `0 * * * *` run always when the minutes are 0 (hourly).
- `0 1 * * *` run always at 1 o'clock.

- `* 1 * * *` run each minute when the hour is 1. (1:00, 1:01, ..., 1:59).
- `*/5 * * * *` run every five minutes.
- `45 23 * * 6` run every Saturday at 23:45 (11:45 PM)

### 5.4.2. Docker Usage

- Build the image using *docker build . -t silvanusproject/pipeline-initiator:1.0.0*
- Place the config.json file in the project folder.
- Run the container with the correct parameters set in the environment variables, according to your configuration information:
  ***1.*** *Docker run --name pipeline-initiator-1.0.0 --restart always -v ./config.json:/pipeline-initiator/config.json -e RABBIT_USERNAME=<USERNAME> -e* ***2.*** *RABBIT_PASSWORD=<PASSWORD> -e RABBIT_IP=<IP> silvanusproject/pipeline-initiator:1.0.0*

The environment variables that can be overwritten are described in Table 16:

**Table 16 - Variables that Can be Overwritten**

| Environment | Description | Default |
|---|---|---|
| RABBIT_USERNAME | The username to access RabbitMQ | guest |
| RABBIT_PASSWORD | The password to access RabbitMQ | guest |
| RABBIT_IP | RabbitMQ server IP | 127.0.0.1 |
| RABBIT_PORT | RabbitMQ server Port | 5672 |
| RABBIT_VHOST | RabbitMQ Virtual Host | / |
| RABBIT_EXCHANGE | The destination exchange for messages | - |

## 6. Conclusions

This deliverable presented the implementation of the different SILVANUS components for the ingestion, aggregation and pre-processing of heterogenous data sources: i.e. Satellite Earth Observations, Weather and Climate data, in-situ IoT devices and its integration into the SILVANUS platform in order to provide advanced capabilities to detect fire and to quantify the forest fire danger. Those components are considered the backbone for the SILVANUS application and services and will be fully exploited in the SILVANUS Pilot sites in the incoming months: such demonstrative phase of the SILVANUS systems capacities will be tested extensively to get feedback on the different developed components from the stakeholders and users.

## 7. References

[1] Apache NiFi. https://nifi.apache.org, Accessed on 29th Mar 2023.
[2] EU-DEM version 1.1. https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1, Accessed on 29th Mar 2023.

[3] OpenStreetMap. https://www.openstreetmap.org, Accessed on 29th Mar 2023.
[4] OpenStreetMap Data. https://download.geofabrik.de/europe.html, Accessed on 29th Mar 2023.
[5] Copernicus Programme. https://www.copernicus.eu/en/about-copernicus , Accessed on 29th Mar 2023.
[6] Copernicus Open Access Hub. https://scihub.copernicus.eu, Accessed on 29th Mar 2023.
[7] Copernicus Open Access Hub Advanced Search. https://scihub.copernicus.eu/userguide/AdvancedSearch, Accessed on 29th Mar 2023.
[8] NetCDF Climate and Forecast (CF) Metadata Conventions. https://cfconventions.org/Data/cf-conventions/cf-conventions-1.10/cf-conventions.html, Accessed on 29th Mar 2023.
[9] WorldPop. https://www.worldpop.org, Accessed on 29th Mar 2023.
[10] WorldPop top-down estimation modeling. https://www.worldpop.org/methods/top_down_constrained_vs_unconstrained. Accessed on 30th Mar 2023.
[11] United Nations – Department of Economic and Social Affairs. https://population.un.org/wpp. Accessed on 30th Mar 2023.
[12] Copernicus Global Land Service – Burnt Area. https://land.copernicus.eu/global/products/ba, Accessed on 29th Mar 2023.
[13] Copernicus Global Land Service – Land Surface Temperature. https://land.copernicus.eu/global/products/lst, Accessed on 29th Mar 2023.
[14] Copernicus Land Monitoring Service – CLC 2018. https://land.copernicus.eu/pan-european/corine-land-cover/clc2018, Accessed on 29th Mar 2023.
[15] EUMETSAT platform. https://www.eumetsat.int/what-we-monitor/weather, Accessed on 29th Mar 2023.
[16] EUMETSAT Data Centre. https://www.eumetsat.int/eumetsat-data-centre, Accessed on 29th Mar 2023.
[17] Active Fire Monitoring, https://navigator.eumetsat.int/product/EO:EUM:DAT:MSG:FIR, Accessed on 29th Mar 2023.
[18] NCEP WMO GRIB2 Documentation. https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc, Accessed on 29th Mar 2023.
[19] Active Fire Monitoring: Product Guide. https://www.eumetsat.int/media/38051, Accessed on 29th Mar 2023.
[20] EUMETSAT, "Normalized Difference Vegetation Index: Product Guide", https://www.eumetsat.int/media/38997, Accessed on 29th Mar 2023
[21] "Sentinel 2 Data Formats", https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/data-formats, Accessed on 29th Mar 2023.
[22] BeautifulSoup. https://www.crummy.com/software/BeautifulSoup/bs4/doc/, Accessed on 29th Mar 2023.
[23] Python interface to NetCDF4. https://pypi.org/project/netCDF4, Accessed on 29th Mar 2023.
[24] Gributils. https://pypi.org/project/gributils/, Accessed on 29th Mar 2023.
[25] Rasterio. https://rasterio.readthedocs.io/en/stable, Accessed on 29th Mar 2023.

[26] Sentinel-2 Products Specification Document. https://sentinel.esa.int/documents/247904/685211/sentinel-2-products-specification-document, Accessed on 29th Mar 2023.
[27] Copernicus Open Access Hub. https://scihub.copernicus.eu/dhus/#/self-registration, Accessed on 29th Mar 2023.
[28] SILVANUS GitHub Platform. https://github.com/silvanus-prj/sentinel2_to_ndvi, Accessed on 29th Mar 2023.
[29] Geofabrik. http://download.geofabrik.de, Accessed on 29th Mar 2023.
[30] OpenStreetMap Data in Layered GIS Format. http://download.geofabrik.de/osm-data-in-gis-formats-free.pdf, Accessed on 29th Mar 2023.
[31] GeoPandas. https://geopandas.org/en/stable/, Accessed on 29th Mar 2023.
[32] Pandas. http://pandas.pydata.org/, Accessed on 29th Mar 2023.
[33] Dask GeoPandas. https://dask-geopandas.readthedocs.io/en/stable/index.html, Accessed on 29th Mar 2023.
[34] Dask. https://dask.org/, Accessed on 29th Mar 2023.
[35] Shapely. https://shapely.readthedocs.io/en/stable, Accessed on 29th Mar 2023.
[36] SILVANUS GitHub Platform. OSM to NetCDF. https://github.com/silvanus-prj/osm_to_netcdf, Accessed on 29th Mar 2023.
[37] Skamarock, W. C.; Klemp, J. B.; Dudhia, J.; Gill, D. O.; Liu, Z.; Berner, J.; Huang, X-yu. (2019). A Description of the Advanced Research WRF Model Version 4.1 (No. NCAR/TN-556+STR). doi:10.5065/1dfh-6p97
[38] Raffa, M.; Reder, A.; Marras, G.F.; Mancini, M.; Scipione, G.; Santini, M.; Mercogliano, P. (2021). VHR-REA_IT Dataset: Very High-Resolution Dynamical Downscaling of ERA5 Reanalysis over Italy by COSMO-CLM. Data 2021, 6, 88. https://doi.org/10.3390/data6080088
[39] CMCC Data Delivery System: ERA5 at 2km for Italy. https://doi.org/10.25424/cmcc/era5-2km_italy, Accessed on 29th Mar 2023.
[40] Agro Meteo Puglia. http://www.agrometeopuglia.it/, Accessed on 29th Mar 2023.
[41] Vaidya, P. M. (1989). "*An O(n log n) Algorithm for the All-Nearest-Neighbors Problem*". Discrete and Computational Geometry. 4 (1): 101-115. doi:10.1007/BF02187718
[42] Copernicus Climate Service: Fire Weather Index. https://climate.copernicus.eu/fire-weather-index, Accessed on 29th Mar 2023.
[43] Alvarez-Castro, M. C.; Faranda, D.; Yiou, P. (2018). "*Atmospheric dynamics leading to West European summer hot temperatures since 1851*". Complexity, *2018*, 1-10.
[44] Copernicus Emergency Management Service. (2019). Fire danger indices historical data from the Copernicus Emergency Management Service. In: *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*. https://doi.org/10.24381/cds.0e89c522
[45] CMCC Data Delivery System. https://dds.cmcc.it, Accessed on 29th Mar 2023.
[46] LoRa network. https://lora-alliance.org, Accessed on 29th Mar 2023.
[47] https://github.com/naturebytes/STL-Naturebytes-Enclosures, Accessed on 29th Mar 2023.
[48] https:///www.adafruit.com/product/385, Accessed on 29th Mar 2023.
[49] https://components101.com/sensors/mq2-gas-sensor, Accessed on 29th Mar 2023.
[50] https://www.raspberrypi.com/products/raspberry-pi-4-model-b, Accessed on 29th Mar 2023.

[51] https://www.raspberrypi.com/products/camera-module-v2, Accessed on 29th Mar 2023.
[52] https://www.waveshare.com/wiki/A7670E_Cat-1_HAT, Accessed on 29th Mar 2023.