Eiffel

GEOSS APPLICATIONS FOR CLIMATE CHANGE

REVEALING THE ROLE OF GEOSS AS THE DEFAULT DIGITAL PORTAL FOR BUILDING CLIMATE CHANGE ADAPTATION & MITIGATION APPLICATIONS

D3.5 Report of new features of the Sentinel Hub App

Version 1.0
Date 28-Nov-2022

Editor	Roxanne S. Lorilla, Charalampos Kontoes (NOA)
Authors	Thanassis Drivas, Anastasis Katsos (NOA), Alexia Tsouni (EDGE)
Reviewers	Dimitris Bliziotis (ICCS), Maria Dekavalla (ICCS)
Dissemination Level	Public

Call	H2020-LC-CLA-2020-2
Topic	LC-CLA-19-2020
Type of Action	Research and Innovation Action
Start Date	01 June 2021
Duration	36 months
Project Information	https://cordis.europa.eu/project/id/101003518

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003518

The Members of the EIFFEL Consortium:

ID	Organisation	Short Name	Country
1	INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS	ICCS	GREECE
2	ETHNIKO ASTEROSKOPEIO ATHINON	NOA	GREECE
3	PRODEVELOP SL	PRO	SPAIN
4	UNIVERSITAT POLITECNICA DE VALENCIA	UPV	SPAIN
5	DRAXIS ENVIRONMENTAL SA	DRAXIS	GREECE
6	STICHTING IHE DELFT INSTITUTE FOR WATER EDUCATION	IHE	NETHERLANDS
7	OPEN UNIVERSITEIT NEDERLAND	OUNL	NETHERLANDS
8	NOORD-BRABANT PROVINCIE	NOORD-BRABANT	NETHERLANDS
9	LIBRA MLI LTD	LIBRA	UNITED KINGDOM
10	DIABALKANIKO KENTRO PERIBALLONTOS	IBEC	GREECE
11	AUTORIDAD PORTUARIA DE BALEARES	BPA	SPAIN
12	UNIVERSIDAD AUTONOMA DE BARCELONA	UAB	SPAIN
13	PERIFEREIA ATTIKIS	ATTICA	GREECE
14	NATIONAL PAYING AGENCY	NPA	LITHUANIA
15	SCHWEIZERISCHES FORSCHUNGSINSTITUT FUER HOCHGEBIRGSKLIMA UND MEDIZIN IN DAVOS	PMOD WRC	SWITZERLAND
16	EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS	ECMWF	UNITED KINGDOM
17	SUOMEN YMPARISTOKESKUS	SYKE	FINLAND
18	RISA SICHERHEITSANALYSEN GMBH	RISA	GERMANY
19	EDGE IN EARTH OBSERVATION SCIENCES MONOPROSOPI IKE	EDGE	GREECE

Disclaimer

Funded by the European Union

Document History

Version	Date	Editor	Comments
0.1	20-Sep-2022	Thanassis Drivas	Table of contents
0.2	20-Oct-2022	Thanassis Drivas	Introduction and Motivation
0.3	08-Nov-2022	Thanassis Drivas, Anastasis Katsos, Alexia Tsouni	Deliverable preparation
0.4	15-Nov-2022	Roxanne S. Lorilla	Internal review
0.5	17-11-2022	Roxanne S. Lorilla	Final editing of First Draft
0.6	22-11-2022	Maria Dekavalla, Dimitris Bliziotis	Review
0.7	25-11-2022	Thanassis Drivas, Anastasis Katsos, Roxanne S. Lorilla	Revisions and final editing
1.0	28-Nov-2022	Roxanne Lorilla	Submitted version

Funded by the European Union

Executive Summary

This document describes the concept and activities connected with Deliverable 3.5 "Report of new features of the Sentinel Hub App". The main goal is to showcase how to filter and subset the results returned by the Umbrella using other data sources, such as GEOSS-related datasets. For this, the main structure of D3.5 consists of an introduction to the Umbrella Sentinel Access Point, the description of the extended features for filtering and subsetting the results returned by the Umbrella based on the condition of external data sources, as well as some examples and scenarios for the use of the extended application. The developed tool will facilitate Pilot studies in decreasing the volume of data needed to be acquired and processed, minimising the storage capacity and computational complexity of the Climate Change (CC) related applications. Lastly, at the end of the document, limitations and future steps to improve the tool are reported, which mainly focus in the availability of accessible external APIs to be linked/blended with Umbrella, and in the possible integration of more external datasets into the developed tool.

Funded by the European Union

Table of Contents

Funded by the European Union

List of Figures

List of Tables

List of Acronyms and Abbreviations

Acronym	Meaning
API	Application Programming Interface
CC	Climate Change
DHuS	Data Hub Software
GEOSS	Global Earth Observation System of Systems

Funded by the European Union

1 Introduction

Researchers and Data Scientists often struggle to unlock the power of Earth Observation (EO) Data. Thus, there is the demand not only for downloading and storing the data, but also to transform them in a more efficient format in order to be given directly as input to their pipelines. Analysis-Ready-Data (ARD) aim at realizing the full potential of EO data by overcoming the obstacles set by the Big data challenges and therefore allowing access to large spatio-temporal data in an analysis ready form. To that direction, we provide a series of tools that download and pre-process the raw data provided by the data hubs.

This document describes the concept, activities, methodologies and definitions connected with Deliverable 3.5 on the extended features of the “Umbrella Sentinel Access Point”. The main structure of D3.5 consists of i) an introduction to the Umbrella Sentinel Access Point, including its use for searching, downloading, and pre-processing Sentinel 1, 2, 3 and 5p, ii) the description of the extended features for filtering and subsetting the results returned by the Umbrella based on the condition of external data sources, and iii) examples and scenarios for the use of the extended tool. This tool can be used by the project’s Pilots to minimise the storage capacity required to download Sentinel data, also decreasing the computational complexity of the Climate Change (CC) related applications.

1.1 Context

1.1.1 Objectives

The Report of new features of the Sentinel Hub App (D3.5) contributes to 1 of 5 EIFFEL project objectives. The main input consisted of the functional and non-functional requirements defined in the D2.2 which has been the detailed outline of specifications for the CC-related applications per Pilot study and the GEOSS tools implementations (Figure 1).

The Grant Agreement[1] lists five project objectives (O1-O5), of which two are indirectly linked to the findings published in this report due to their relevance for T3.5:

- (**O3**) EIFFEL will contribute to GEO’s new infrastructural feature, known as the GEO Knowledge Hub[2] (GKH); GKH is a digital repository providing access to knowledge needed to build GEOSS-driven applications.

- (**O5**) EIFFEL will develop, using co-creation, a set of CC adaptation and mitigation applications in different and quite diverse GEO SBAs, to demonstrate the project innovations in the following Pilot studies: PILOT1/P1-Water/Land Management, PILOT2/P2-Sustainable Agriculture, PILOT3/P3-Transport Infrastructure, PILOT4/P4-Sustainable Urban Development, PILOT5/P5-Disaster Resilience.

The main goal of D3.5 is to showcase how to filter and subset the results returned by the Umbrella using other data sources, such as GEOSS-related datasets. The developed tool will

[1] Grant Agreement No 101003518, Part B, p. 5

[2] Officially introduced as a concept during the GEO Ministerial Summit in Canberra, Australia, 4-9/11/2019

Funded by the European Union

facilitate Pilot studies, but not limited to, in decreasing the volume of data needed to be acquired and processed. Furthermore, it is designed to offer users the ability to minimise the storage capacity required to download Sentinel data, increasing the time and storage efficiency of the CC applications that will be using Sentinel data.

Figure 1. EIFFEL Pilot studies in the European Union.

1.1.2 Work plan

This report, **Deliverable 3.5**, corresponds to **T3.5** “Blend Sentinel Hub App with GEOSS portal datasets (M3-M18)”, and is part of **WP3** “Augmenting GEOSS data exploration”.

Through an elaborated process of engaging stakeholders organised around the five pilot applications, EIFFEL, through T2.2 and D2.2, has used participatory means to determine the project personas, scenarios, and user requirements. The latter consisted of the main input to the horizontal technical developments of relevant Work Packages (WP), namely WP3 and WP4. This document reports the workflow related to WP3 and specifically D3.5 on the new features of the Umbrella Sentinel Access Point.

1.1.3 Milestones

D3.5 is linked to MS4:

- Operational services for ARD Sentinel raw data and products ready for the pilots

Funded by the European Union

1.1.4 Deliverables

D3.5. was based on D2.2 on the Report on EIFFEL specifications and is strongly correlated to the actions and deliverables related to the CC applications WP5, WP6 and WP7.

1.2 Intended Readership and Document Structure

By compiling the information presented in D2.2, it is expected that all Pilots can make use of the products/tools related to D3.5. However, the way the scripts and application were created was to enable their use by users (not only those involved with EIFFEL) that are interested in or in need of filtering/subsetting, downloading, and pre-processing Sentinel satellite images in less time and efficiently.

The document is organised as follows:

- **Section 1** provides the context and the intended readership for this report.
- **Section 2** describes the Sentinel Umbrella application, including its architecture, technologies and main advantages.
- **Section 3** presents the extended tools of the Sentinel Umbrella application, the available parameters to filter the results returned by the Umbrella, the back-end processes of the tool, and the final architecture.
- **Section 4** presents some examples of the use of the extended application.
- **Section 5** describes the limitations, as well as the future steps to improve of the developed tool.

Funded by the European Union

2 The Umbrella Sentinel Access Point

2.1 Overview of the application

The first tool provided by task 3.5 is linked to MS4 and refers to the Sentinel data service provision to EIFFEL applications. As searching for Sentinel data is often a complicated process (due to the different available missions, the different hubs that host the data, and the different performances of the hubs), rather than searching on several hubs to acquire the needed datasets, the EIFFEL Pilot applications will be given access to a dedicated API through Jupiter notebooks to download Sentinel data. This will be based on the Umbrella Sentinel Access Point developed by NOA (http://umbrella.beyond-eocenter.eu/), which provides uniform access to Sentinel 1, 2, 3 and 5p metadata via connecting at the back end (such as Open Access Hub, PEPS, and ONDA Dias) to a number of available Sentinel access points. Umbrella is not connected only to DHuS-based platforms but is platform-independent. The solution provides a single data access point, which is already successfully deployed and accessible. This unique access point provides users with all the required Sentinel metadata related to their requests via an API, allowing the implementation of a transparent single data access point. Ultimately, this application gives the potential for:

1. linking federated Copernicus Sentinels Hubs to a single data hub instead of searching for the most appropriate for the user's needs;
2. accessing all Sentinel mission data and better performance on downloading products, as products are chosen from the most appropriate data hub based on integrity, speed and availability tests. Note, users occasionally visit more than one hub to discover and select the required products due to several reasons.

Most data hubs, but not all, are using DHuS, which allows users to access the data via their own scripts or user applications. API Query is based on the OpenSearch protocol, allowing the publishing of search results in a format suitable for syndication and aggregation. Existing umbrella applications provide the capacity for users to not search on several hubs but on one. Due to the vast amount of Sentinel data, every hub has different data offerings, rolling policies, and data availability (e.g., maintenance or providing not all Sentinel missions). Based on the EIFFEL applications' requirements, the best solution of the two developed applications will be selected and enhanced by providing not only the best product(s) to download but also pre-processing products based on the raw Sentinel data. Therefore, the EIFFEL Pilot applications will have access to a dedicated API to download the needed pre-processed data/indices, making the creation of the application faster and easier.

The application is hosted in NOA's infrastructure, providing the needed resources (hardware, software, bandwidth, reliability etc.). In this way, the Sentinel Umbrella application tackles problems related to the existing data hubs; lack of global coverage, availability of all Sentinel missions, data rolling policy, maintenance, and performance issues. Bringing these data sources together has the advantages of accessing a single hub instead of looking across several Sentinel Hubs to find the appropriate products, accessing all Sentinel

Funded by the European Union

missions data without geographic restrictions and providing better performance as the Umbrella serves the most efficient data source to download from on each request. The main advantages of linking these sources are listed below:

- access to a single hub instead of looking across several Sentinel Hubs to find the appropriate products;
- access to all Sentinel mission data;
- no geographic restrictions;
- better performance/download variability by exploiting Hub diversity;
- no delay due to maintenance of a hub.

2.2 Architecture

The architecture of the Umbrella application aims at designing a single point of access for Sentinel metadata by searching and collecting them from all the available hubs, either these hubs are based on the DHuS software or not. The application is based on three main processes that take place in order to eliminate the aforementioned limitations (Figure 2):

1. searching hubs for new metadata;
2. scoring hubs based on performance;
3. deleting unavailable metadata.

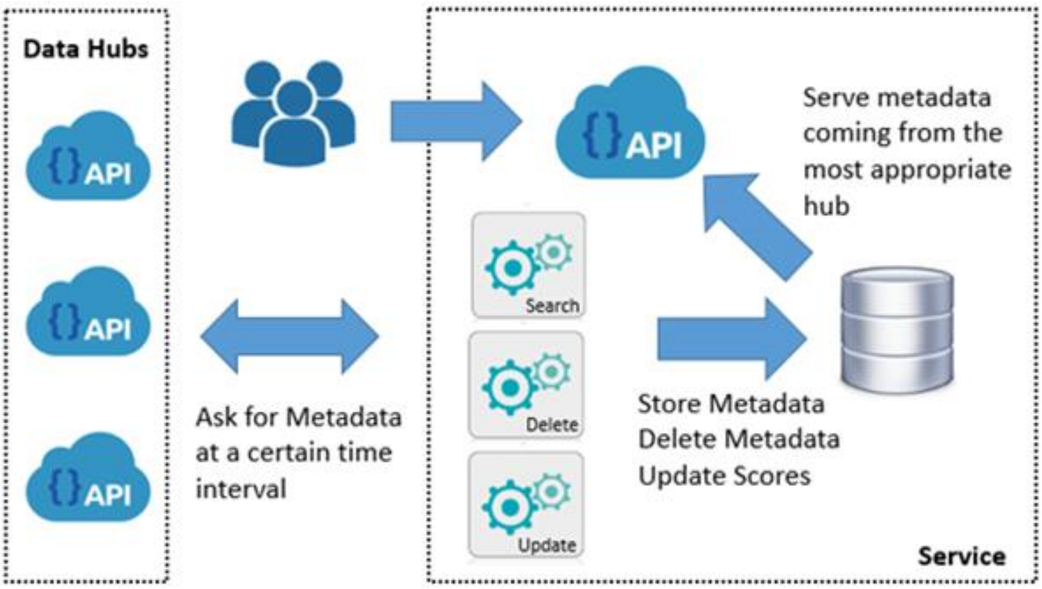

Figure 2. The architecture of the Umbrella Sentinel Access Point.

The searching process is executed multiple times per day, in particular every fifteen minutes in order to harvest new metadata ingested in the hubs. Taking into consideration that Sentinel data are made available to the hubs usually two to twelve hours post sensing, it is important to search for newly ingested products every fifteen minutes. This frequent

Funded by the European Union

update of metadata is crucial for near-real time applications that are dependent on the immediate acquisition of the required data. Furthermore, the scoring process is set to run every ten minutes allowing the frequent check of availability and download speed for each connected hub in order to rank the hubs. Having a short time scoring frequency ensures that the last measured download speed is indeed representative of the current download speed. Consequently, when a user asks for a product, they get metadata from the most appropriate - at this time - hub. In addition, a part of metadata stored in the hubs is deleted due to several reasons, including the respective hub's archive rolling policy. As users wait for getting active download links for the requested products, the Umbrella application needs to clear all the records already stored locally but that do not longer exist at the hubs.

2.3 Technologies

The development of the Umbrella application is based on the Django[3] framework for Python. Django is considered to be a very mature choice with a large community supporting it. The selected framework provides a high-level abstraction of the common web while it comes fully equipped with libraries related to user authentication, site maps, etc. These functionalities aim at simplifying the building of several processes. One of the main advantages of Django is that it also provides the GeoDjango module, which allows geospatial functionalities. Finally, Django REST framework was selected for the Umbrella API. Representational State Transfer (REST) is an architectural model that is used to design distributed software architectures based on network communication. Responses of the selected API type have to be in eXtensible Markup Language (XML), JavaScript Object Notation (JSON), Yet Another Markup Language (YAML), or any other format depending on what the users request. Moreover, it has to be stateless, meaning that requests can be made independently of one another, and each request contains all of the required data to complete itself successfully. The Django REST framework is powerful and flexible for building Web APIs, allowing filtering and easy data serialisation.

Moving to the database layer, PostgreSQL has been selected as the relational database management system, which has proven very reliable. PostgreSQL comes without licensing costs and has the potential to scale up easily. In addition, GeoDjango is fully compatible with PostgreSQL, and also supports JSON type. The PostGIS extension adds extra types (geometry, geography, raster and others) to the PostgreSQL database. Functions, operators, and index enhancements that apply to these spatial types can also be added. These extra functions, operators, index bindings, and types empower the core PostgreSQL DBMS, upgrading it into a fast and robust spatial database management system.

Concerning the interaction among the databases, Django Models API simplifies the interaction with them. Most common databases are programmed with some form of Structured Query Language (SQL), however, each database implements SQL in its own way. Django Models API provides an Object-relational Mapping (ORM) to the underlying database.

[3] https://www.djangoproject.com/

Funded by the European Union

ORM is a powerful programming technique that simplifies tasks related to data and relational databases.

2.4 Use case scenario

The following scenario illustrates how the Umbrella app can be exploited. Initially, users have to write a simple script to make requests to the API. There are four endpoints, one for each Sentinel mission. The request included a series of essential parameters such as the bounding box in the form of xmin, ymin, xmax, ymax and the time range. Additional parameters may be used as for example the cloud coverage. An example of a request in the Sentinel-2 endpoint is the following:

```
url='http://umbrella.beyond-eocenter.eu/api/products/sentinel2?in_bbox=
20.8,38.41,23.82,40&sensing_date__gte=2022-10-01&sensing_date__lte=2022-10-31'
```

The response of the request above is depicted below in JSON format and it contains products from more than one data sources (e.g. PEPS and Open Access Hub) due to the reason that some products may have been deleted from a data hub.


```
"count": 58,
"next": null,
"previous": null,
"results": [
    {
        "id": 66242494,
        "uuid": "faa4b9e4-9a01-4604-b30d-03197569d1f3",
        "identifier": "S2A_MSIL1C_20221027T091101_N0400_R050_T34SFH_20221027T111019",
        "filename": "S2A_MSIL1C_20221027T091101_N0400_R050_T34SFH_20221027T111019.SAFE",
        "url_download": "https://scihub.copernicus.eu/dhus/odata/v1/Products('faa4b9e4-9a01-4604-b30d-03197569d1f3')/$value",
        "url_checksum": "https://scihub.copernicus.eu/dhus/odata/v1/Products('faa4b9e4-9a01-4604-b30d-03197569d1f3')/Checksum/Value/$value",
        "instrument": "MSI",
        "product_type": "S2MSI1C",
        "cloud_coverage": "0.115995",
        "sensing_date": "2022-10-27T09:11:01.024000Z",
        "ingestion_date": "2022-10-27T14:41:42.966000Z",
        "orbit_number": "38371",
        "relative_orbit_number": "50",
        "pass_direction": "DESCENDING",
        "wkt_footprint": "MULTIPOLYGON (((23.384099971340483 37.83530211029287, 23.416732203212913 38.82399234658653, 22.7851380532572 38.83364389959216, 22.76899286345851 38.77948124886581,
        22.726465618968952 38.63206656345893, 22.684468775530163 38.48452780647443, 22.64045119686436 38.33765652118512, 22.597820591759287 38.190398012709245, 22.556443732821386 38.04273389595777,
        22.51420568547393 37.89525254865437, 22.50064319345844 37.848513019658874, 23.384099971340483 37.83530211029287)))",
        "size": "461.59 MB",
        "source": "https://scihub.copernicus.eu",
        "tile_name": "34SFH",
      ▸ "geom_footprint": { … }, // 2 items
        "quicklook": "https://scihub.copernicus.eu/dhus/odata/v1/Products('faa4b9e4-9a01-4604-b30d-03197569d1f3')/Products('Quicklook')/$value"
    },
    {
        "id": 66241956,
        "uuid": "1f8df178-2aee-5a60-9474-317577eef04a",
        "identifier": "S2A_MSIL1C_20221027T091101_N0400_R050_T34SFJ_20221027T111019",
        "filename": "S2A_MSIL1C_20221027T091101_N0400_R050_T34SFJ_20221027T111019.SAFE.SAFE",
        "url_download": "https://peps.cnes.fr/resto/collections/S2ST/1f8df178-2aee-5a60-9474-317577eef04a/download",
        "url_checksum": "-",
        "instrument": "MSI",
        "product_type": "S2MSI1C",
```


Funded by the European Union

3 The extended Umbrella Sentinel Access Point

3.1 Motivation

Earth Observation monitoring aims at mapping, analysing and understanding the status of a series of activities that take place on Earth. This procedure is based mainly on satellite Imagery. These types of images are considered to be a major source of EO data and have already been utilised for the development of important applications. However, building on recent trends in the context of big EO data analytics, additional data sources, such as meteorological data, have been used alongside satellite data. This multi-source data approach is also relevant to the EIFFEL Project. Specifically, the pilots of the project will be able to blend Sentinel Data with meteorological data. For example, Pilot 1 demands the fusion of temperature, precipitation and Sentinel-2 data. In that direction, a diverse set of observations can be used for constructing a more accurate dataset and at the same time to contribute to an improved data analysis.

The Umbrella Sentinel Access Point has already laid the foundations of a smart and efficient searching of Sentinel metadata. However, it only deals with satellite images' metadata. Taking into consideration all the above, the need to extend the functionalities of Umbrella towards the fusion of meteorological and Sentinel data is manifold. The updated application proposed in this document connects to additional APIs that provide meteorological data. In this way, it not only accepts users' requests for Sentinel products but also enables users to adding meteorological parameters to the request so as to receive metadata for Sentinel images based on these new parameters. The main advantages of linking these sources together are reported below:

1. The number of products that are returned as a response to the users' query is decreased if all images do not fulfil the conditions of the different meteorological parameters.
2. A one-stop-shop for multi-source data, as the Umbrella API can provide except from the filtered list of Sentinel metadata, the actual meteorological data for the requested bounding box.

3.2 Data sources

On top of the Sentinel missions the extended Umbrella Sentinel Access Point includes meteorological data coming from multiple sources. These sources either allow on-the-fly requests for retrieving the stored data or provide downloadable files that contain the targeted information. The second category of data sources acts as an alternative solution for blending satellite metadata and meteorological data, if there is no availability of free of cost APIs or/and no provision of actual data from the endpoints. Both solutions serve as a working proof of concept to showcase what can be done by extending the umbrella with other apis of that sort. In the future more apis and geoservers will be connected. Focusing on the current

Funded by the European Union

data sources that Umbrella connects to, Meteostat[4] is considered as one one of the largest vendors of open weather and climate data as it provides open long-term time series of thousands of weather stations. On the other hand, local Geoserver has been populated with weather and climate data coming from several data sources, traced via the Global Earth Observation System of Systems (GEOSS[5]) catalogue and which does not provide the raw values of the variables on-the-fly, but only in downloadable file formats. GEOSS acts as an ecosystem under which users have the potential to discover and access heterogeneous collections of Earth observations globally. GEOSS single access point serves users the links to the repositories of the requested data. Nonetheless, these links often are inactive or lead to the homepage of the data source, forcing users to start over the searching process.

3.2.1. MeteoStat API

MeteoStat API[6] offers a unique and transparent access to weather and climate data, that are harvested from several data sources and ingested into MeteoStat's databases. Data can be retrieved from the users either via a simple HTTP request or by downloading full data dumps. In contrast to most other weather-related applications and APIs, Meteostat focuses on historical weather and climate data that was actually measured on-site by weather stations around the globe. You can either retrieve raw observations of individual weather stations which are free of interpolation or use our comfortable point data to query data by geographic location. Meteostat uses weather and climate data provided by the following interfaces and organisations:

- Deutscher Wetterdienst[7]
- NOAA - National Weather Service[8]
- NOAA - Global Historical Climatology Network[9]
- NOAA - Integrated Surface Database[10]
- Government of Canada - Open Data[11]
- MET Norway[12] -
- European Data Portal[13]
- Offene Daten Österreich[14]

[4] https://meteostat.net/en/
[5] https://www.geoportal.org/
[6] https://rapidapi.com/meteostat/api/meteostat
[7] https://www.dwd.de/DE/Home/home_node.html
[8] https://www.weather.gov/
[9] https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-daily
[10] https://www.ncei.noaa.gov/products/land-based-station/integrated-surface-database
[11] https://open.canada.ca/en/open-data
[12] https://www.met.no/en/free-meteorological-data/Open-source-code
[13] https://data.europa.eu/en
[14] https://www.data.gv.at/

Funded by the European Union

3.2.2. Local Data on GeoServer

GeoServer is an open-source server for sharing geospatial data. It implements industry-standard OGC protocols such as Web Feature Service (WFS), Web Map Service (WMS) and Web Coverage Service (WCS)[15]. Umbrella API blending uses WCS[16], which provides multidimensional coverage data accessible via the web. GeoServer[17] is a modular application with additional functionality added through extensions. It uses two extensions: the GRIB[18] extension, which allows you to use GRIB files, and the CoverageJSON[19] extension, which allows you to use the CoverageJSON format. CoverageJSON is a format for encoding geotemporal coverage data like grids and time series. For example, it can be used as output format for a WCS2.0 getCoverage requesting a TimeSeries on a specific coordinate.

The data used for the local GeoServer implementation is "ERA5-Land hourly Data from 1950 to present". ERA5-Land[20] is a reanalysis dataset that provides a consistent overview of the evolution of land variables over decades. The temporal and spatial resolution of ERA5 land makes this dataset very useful for all kinds of surface applications, such as flood and drought forecasting. The temporal and spatial resolution of this dataset, the period covered in time, as well as the fixed grid used for the data distribution at any period enables users to access and use more accurate information on land states. The ERA5-Land dataset covers the period of January 1950 to 2-3 months before the present. Considering the launch of the first Sentinel on 3 April 2014 [2] and its first public image on 4 October 2014 [3] data from the 1st of October 2014 up until the latest available month will be used. The spatial resolution of the data is 0.1° x 0.1° which translates to a native resolution of 9km and the temporal resolution of the data is 1 hour[21].

For the purposes of this deliverable the variable 'Total precipitation' has been downloaded for Europe (CDS API Coordinates: North: 71, West: -10, South: 34, East: 35) for the years 2020, 2021 and the first 3 months of 2022. The files are stored on a local NAS in the GRIB file type. Each month has a size of about 200MB. No pre-processing was needed to be done on the data.

3.3 Extension 1 - One-the-fly connection to external sources

The parameters within the REST API can be both satellite-based and non-satellite based. Presuming that the query contains parameters related to other data sources, there are two scenarios that are taken into consideration:

---

[15] https://geoserver.org/

[16] https://www.ogc.org/standards/wcs

[17] https://geoserver.org/about/

[18] https://docs.geoserver.org/stable/en/user/extensions/grib/grib.html

[19] https://docs.geoserver.org/stable/en/user/community/cov-json/index.html

[20]https://www.geoportal.org/?m:activeLayerTileId=osm&targetId=71360e53-ad4f-4afc-8dab-aebc6c4c8632&f:phrase=%20ERA5-Land%20hourly%20data%20from%201950%20to%20present&f:dataSource=dab

[21] https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-land

Funded by the European Union

1. If the data source's API provides data values in specific formats (e.g. json), the application makes an on-the-fly request, retrieves the data and makes usage of this data in order to meet the requirements set by the user's request.
2. Otherwise, the application performs a query to the PostgreSQL/PostGIS database in order to access the relevant data that were harvested via GEOSS portal, downloaded in an appropriate file format, and stored in the GeoServer.

The workflow for the update API which implements the aforementioned scenarios is depicted in Figure 3.

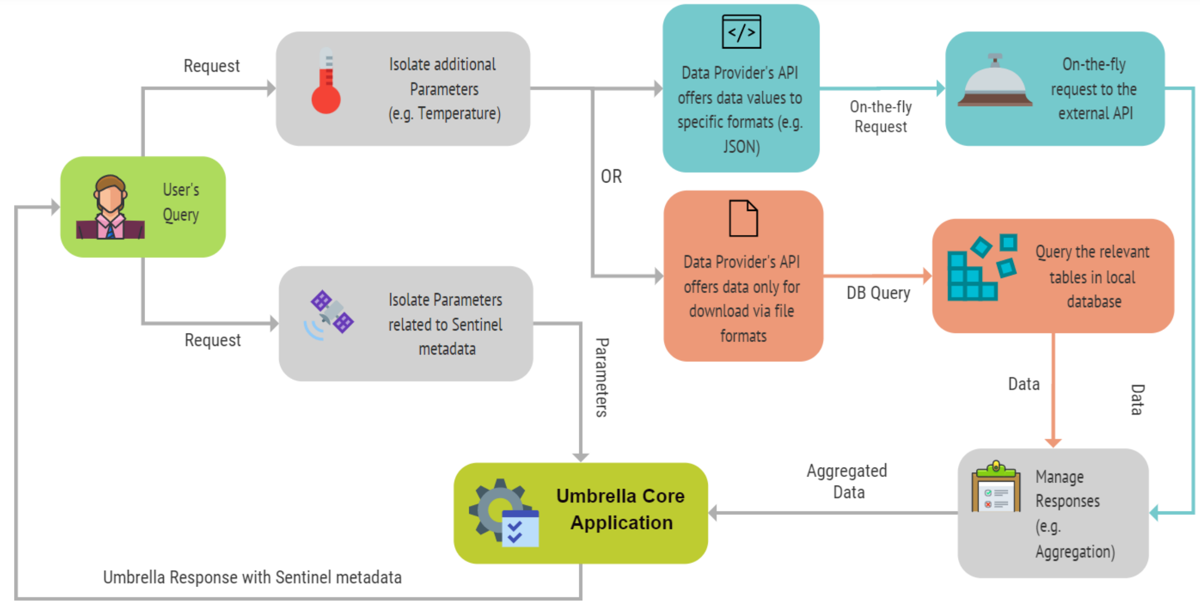

Figure 3. The workflow for blending Umbrella Sentinel Access Point Application with external datasets.

3.3.1 Additional Parameters

The Umbrella Sentinel Access Point module is in charge for providing a series of metadata for Sentinel products based on users' requests. These requests include parameters that are directly related to Sentinel data such as maximum percentage of cloud coverage. Now, users are able to use the following parameters to filter the results:

Table 1. Available parameters that can be used by users to filter the Umbrella's results.

Parameter	Possible Values	Description
variable	precipitation, max_tempature, min_temperature, min_wind_speed, max_wind_speed	This parameter showcases the desired meteorological observation

Funded by the European Union

threshold	float numbers with no limitation	These are the limits above or below the variable's value. For example, if min is chosen, then the app filters and keeps only the variable's values greater or equal than min.
stations	all, one, majority	This parameter indicates the number of stations that satisfy the request variable's threshold. As the requested bounding box may include more than one station, it is essential for the user to ask the number of the stations that meet the requirements. Thus, the all value demands all of the stations to satisfy the criteria, whereas the one value needs at least one. Finally, the majority value counts the number of the stations that meet the requirement and if this number is more than half of them, then the metadata of this time instance is included in the application's response.

3.3.2 Back-end processes

This section describes the Umbrella compatible back-end characteristics and processes currently available. The presented functionalities for connecting to MeteoStat API are implemented under the same framework of Umbrella's first version and developed as Django views. The main processes are described below.

Request's Parsing

Users can construct requests via HTTP using the list of available parameters mentioned in the Umbrella's website[22]. In that direction, the views' functions of the Umbrella Application have been updated to parse also the aforementioned meteorological parameters. Currently, the available parameters include both Sentinel metadata as they are listed in the Umbrella's website and the climate-related parameters. Finally, the existence or not of a meteorological parameter is checked in order to determine the trigger of the corresponding function.

On-the-Fly Request

As the application has analysed the user's request, the next step includes the request to the external API based on the given variable (e.g. wind speed). As the MeteoStat API exploits values that have been recorded via stations, the selected stations have to be determined during the request. In this way, the Umbrella App constructs a request to the MeteoStat that includes the given bounding box and collects data from all the stations located within it. A simplified code is shown below:

```
stations = Stations()
all_stations = stations.bounds((ymax,xmin),(ymin,xmax)).fetch()
data = Daily(all_stations, start, end)
data = data.fetch()
index = data.index
```

As the subset of stations related to the specific area has been selected, we collect the **hourly** data from these stations. Taking into consideration possible differences of the

[22] http://umbrella.beyond-eocenter.eu/

Funded by the European Union

requested variable during the day, we only keep the values that come with the **closest datetime** to the satellite product's sensing datetime. Each hour is represented by a Pandas DataFrame row which provides the weather data recorded at that time. Figure 4 depicts the available columns of the DataFrame.

In case there are no stations inside the given area of interest, the application asks for the **interpolated values on each point inside the area of interest**. We are currently working on alternative solutions such as to determine a **buffer zone outside the requested bounding box** and request the closest station inside this area or the option not to filter Sentinel products if there is not any station found.

Column	Description	Type
station	The Meteostat ID of the weather station (only if query refers to multiple stations)	String
time	The datetime of the observation	Datetime64
temp	The air temperature in °C	Float64
dwpt	The dew point in °C	Float64
rhum	The relative humidity in percent (%)	Float64
prcp	The one hour precipitation total in *mm*	Float64
snow	The snow depth in *mm*	Float64
wdir	The average wind direction in degrees (°)	Float64
wspd	The average wind speed in *km/h*	Float64
wpgt	The peak wind gust in *km/h*	Float64
pres	The average sea-level air pressure in *hPa*	Float64
tsun	The one hour sunshine total in minutes (*m*)	Float64
coco	The weather condition code	Float64

Figure 4. An example of available columns in the DataFrame of the nearest station.

Data Filtering

The response of the MeteoStat API in the pandas dataframe format comes with all the possible variables. Thus, it needs a filtering to only keep the one that meets certain requirements defined by the user during the construction of the request. Data Filtering eliminates values above (or below) this threshold. The next step includes the on-off requirement of identifying the acceptable values to **at least one, the majority or all of the stations found in the area of interest.** A series of scenarios of this functionality is given to the following table:

Funded by the European Union

Table 2. Example scenarios for data filtering.

# Scenario	Temperature			Operator	Acceptable Date for given operator and temperature below 20?
	Station 1	Station 2	Station 3		
1	17	22	23	At least one	Yes
2	19	19	20	Majority	Yes
3	22	19.3	21	Majority	No
4	27	18	18.1	All	No
5	18	19	19.2	All	Yes

Finally, the dedicated function returns a list, which stores the dates that meet the aforementioned criteria.

Response

Having calculated all the accepted dates based on the meteorological observations, an **intersection** with the Sentinels' products dates retrieved from the internal database takes place. The metadata related to the common items are included in the response.

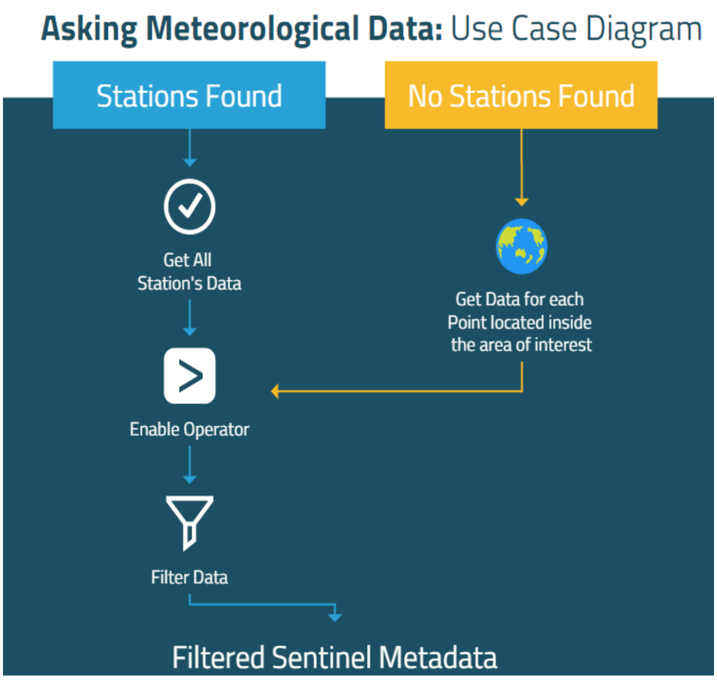

Figure 5. Process for filtering Sentinel metadata.

Funded by
the European
Union

3.4 Extension 2 - Internal storage of required data

The second way of blending Sentinel data with mereological data is through the use of locally stored data served through a local GeoServer. The use of locally stored data complements the aforementioned external API solution in a great way since it can hide some of its potential weaknesses and make the Umbrella API more robust. One of the upsides of a local setup is the guaranteed availability of data. In case an outage in an external API is noticed, the time to fix the issue is uncertain, since we are not the ones in control, and could vary from minutes to days. In case an outage is noticed in the local GeoServer, troubleshooting and fixing the issue would be very fast. By using both methods, a possible unavailability of the Umbrella API would be highly unlikely. Another upside of the local setup is the data selection and variety. If a user requests specific meteorological data to be implemented which are not available on an online API, the data could be downloaded locally and be tailored to the user's needs. The downsides of the local setup are the local storage and system requirements needed. Accumulating a lot of data for a long period of time can significantly increase the total volume occupied. Also, the use of a local GeoServer assumes the availability of system specifications (e.g. CPU, RAM) that will handle user traffic without any issues.

3.4.1 Architecture

The architecture is based on three steps. Firstly, the data is requested and downloaded from the CDS API. Then, the data is inserted into the GeoServer and finally the Umbrella API requests the data from the GeoServer to filter the products. An SQLite database is used to store the metadata of the downloaded products and records whether all the steps have been completed.

3.4.1.1 Database

For every product to be downloaded, a record is made with the attributes "year" (the year the product corresponds to), "month" (the month the product corresponds to), "filename" (the name of the product to be downloaded), "downloaded" (whether the product has been downloaded), "indexed" (whether the product has been inserted in the GeoServer). By default, the attributes "downloaded" and "indexed" are set to 0. When the product is downloaded the value of "downloaded" is set to 1 and when it is inserted in the GeoServer the value of "indexed" is set to 1. It should be noted that every product corresponds to a single month of data.

Funded by the European Union

year	month	filename	downloaded	indexed
Filter	Filter	Filter	Filter	Filter
2021	1	total_precipitation_2021_1.grib	1	1
2021	2	total_precipitation_2021_2.grib	1	1
2021	3	total_precipitation_2021_3.grib	1	1
2021	4	total_precipitation_2021_4.grib	1	1
2021	5	total_precipitation_2021_5.grib	1	1
2021	6	total_precipitation_2021_6.grib	1	1
2021	7	total_precipitation_2021_7.grib	1	1
2021	8	total_precipitation_2021_8.grib	1	1
2021	9	total_precipitation_2021_9.grib	1	1
2021	10	total_precipitation_2021_10.grib	1	1
2021	11	total_precipitation_2021_11.grib	1	1
2021	12	total_precipitation_2021_12.grib	1	1

Figure 6. Database columns and example records.

3.4.1.2 CDS API

The Climate Data Store Application Program Interface[23] is a service providing programmatic access to CDS data. Once the CDS API client is installed, it can be used to request data from the datasets listed in the CDS catalogue. The data are downloaded by making requests to the CDS API. Requests are being constructed using the “cdsapi” python library. For every product (a month of data), the request on the cdsapi library is:

```
c.retrieve('reanalysis-era5-land',
        {
                'variable': variable,
                'year': year,
                'month': month,
                'day': day,
                'time': hours,
                'area': area_bbox,
                'format': 'grib'
        },
                file_path
        )
```

Where ‘variable’ is the variable of the data to be downloaded, ‘year’ the year to be downloaded, ‘month’ the month to be downloaded, ‘day’ the days to be downloaded, ‘time’

[23] https://cds.climate.copernicus.eu/api-how-to

Funded by the European Union

the hours to be downloaded, 'area' the bounding box of the requested area, 'format' the filetype format of the requested file and 'file_path' the path to the file. In our case, the year, month and file_path are changed for each product. The rest of the variables are set as:

```
variable = 'total_precipitation'
day = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18',
    '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31']
hours = ['00:00', '01:00', '02:00', '03:00', '04:00', '05:00', '06:00', '07:00', '08:00', '09:00', '10:00', '11:00',
    '12:00', '13:00', '14:00', '15:00', '16:00', '17:00', '18:00', '19:00', '20:00', '21:00', '22:00', '23:00', ]
area_bbox = [71, -10, 34, 35]
```

The downloaded files use the name convention:

"{Variable}_{Year}_{Month}.grib"

After the download has been completed the database is updated accordingly.

3.4.1.3 GeoServer

To insert a product in the GeoServer there are certain steps that need to be taken. Firstly, in order to group all the datastores and layers that will be created, a workspace is created. A workspace is a grouping of data stores. It is used to group data that is related in some way. We created a workspace named "Umbrella". For every product downloaded, a store is automatically created, using the GeoServer REST API, pointing to the downloaded product. A Store connects to a data source that contains raster or vector data. A data source can be a file or group of files, a table in a database, a single raster file, or a directory (such as a Vector Product Format library). The store[24] construct allows connection parameters to be defined once instead of for each dataset in a source. For example, a datastore (Figure 7) named "total_precipitation_2020_1" belongs to the Umbrella workspace and its URL is the file:

"/mnt/data/Umbrella/total_precipitation/2020/total_precipitation_2020_1.grib"

After the datastore has been created, a layer is also created pointing to the created datastore. The term "layer" refers to a raster or vector dataset that represents a collection of geographic features. Vector layers are analogous to "featureTypes" and raster layers are analogous to "coverages". Every layer has a source of data, known as a Store. The layer[25] is

[24] https://docs.geoserver.org/stable/en/user/data/webadmin/stores.html

[25] https://docs.geoserver.org/stable/en/user/data/webadmin/layers.html

Funded by the European Union

associated with the Workspace (Figure 8) in which the Store is defined. After a layer has been created for a product, it can be accessed through the GeoServer services (e.g. WCS).

Stores

Manage the stores providing data to GeoServer

Add new Store Remove selected Stores

<< < 1 2 > >> Results 1 to 25 (out of 26 items)

Data Type	Workspace	Store Name
	Umbrella	total_precipitation_2020_1
	Umbrella	total_precipitation_2020_10
	Umbrella	total_precipitation_2020_11
	Umbrella	total_precipitation_2020_12
	Umbrella	total_precipitation_2020_2
	Umbrella	total_precipitation_2020_3
	Umbrella	total_precipitation_2020_4
	Umbrella	total_precipitation_2020_5
	Umbrella	total_precipitation_2020_6
	Umbrella	total_precipitation_2020_7
	Umbrella	total_precipitation_2020_8
	Umbrella	total_precipitation_2020_9

Figure 7. Example Stores on the GeoServer.

Layers

Manage the layers being published by GeoServer

Add a new layer Remove selected layers

<< < 1 2 > >> Results 1 to 25 (out of 26 items)

Type	Title	Name	Store
	total_precipitation_2020_1	Umbrella:total_precipitation_2020_1	total_precipitation_2020_1
	total_precipitation_2020_10	Umbrella:total_precipitation_2020_10	total_precipitation_2020_10
	total_precipitation_2020_11	Umbrella:total_precipitation_2020_11	total_precipitation_2020_11
	total_precipitation_2020_12	Umbrella:total_precipitation_2020_12	total_precipitation_2020_12
	total_precipitation_2020_2	Umbrella:total_precipitation_2020_2	total_precipitation_2020_2
	total_precipitation_2020_3	Umbrella:total_precipitation_2020_3	total_precipitation_2020_3
	total_precipitation_2020_4	Umbrella:total_precipitation_2020_4	total_precipitation_2020_4
	total_precipitation_2020_5	Umbrella:total_precipitation_2020_5	total_precipitation_2020_5
	total_precipitation_2020_6	Umbrella:total_precipitation_2020_6	total_precipitation_2020_6
	total_precipitation_2020_7	Umbrella:total_precipitation_2020_7	total_precipitation_2020_7
	total_precipitation_2020_8	Umbrella:total_precipitation_2020_8	total_precipitation_2020_8
	total_precipitation_2020_9	Umbrella:total_precipitation_2020_9	total_precipitation_2020_9

Figure 8. Example Layers on the GeoServer.

3.4.1.4 Umbrella API Blending

In order to blend the Sentinel data with the meteorological data stored on the GeoServer, the following steps are being made:

1) **Retrieve the variable values from the Umbrella API query**. These variables are "variable" (e.g. precipitation), "threshold" (any integer), "in_bbox" (a bounding box based on Latitude and Longitude), "sensing_date__gte" and "sensing_date__lte".

Funded by the European Union

2) **For every Sentinel image that is returned from the Umbrella API, the "sensing_date" of the image is being used**, along with the bounding box that the user has provided, to build a query for the GeoServer. For instance, the Umbrella API query:

```
"/api/products/sentinel1?variable={variable}&threshold={threshold}&in_bbox={Long1},{Lat1},{Long2},{Lat2}&sensing_date__gte={Date1}&sensing_date__lte={Date2}"
```

will return a list of X products. Then for every product, a query to the GeoServer is made:

```
"/geoserver/ows?service=WCS&version=2.0.1&request=GetCoverage&CoverageId={variable}_{SD_Year}_{SD_Month}&subset=Long({Long1},{Long2})&subset=Lat({Lat1},{Lat2})&subset=time("{SD_Rounded}")&format=application/prs.coverage+json"
```

where SD_Year is the year of the sensing_date of the image, SD_Month is the month of the sensing_date of the image and SD_Rounded is the sensing_date of the image rounded to the hour (e.g. 2022-01-04T01:25:59.1234Z becomes 2022-01-04T01:00:00.0000Z). Note the use of the *getCoverage* request along with the "*application/prs.coverage+json*" format which, as per the specification, is the format to be specified to get back a cov-json output.

3) **The response of the GeoServer query consists of a list of values which correspond to the pixel values of the CDS data contained in the selected bounding box for the sensing_date of the image**. The mean of all the pixels is then calculated. In this step, other calculations are taking place as well. For example, the precipitation unit is converted from metres to millimetres.

4) **If the mean value is above the threshold defined by the user, the image will be contained in the results returned to the user**. If the value is below the threshold set, the image is skipped.

3.5 Provided tools

The Umbrella Sentinel Access Point shares only the metadata for Sentinel images and therefore users have to download the actual data. As the Umbrella is able to provide a list of multiple products, it becomes apparent that there must be an automated way to download these products from the corresponding data sources. The downloaded products come in a raw format. This format is not utilised by most of the users as there must be a series of pre-processing steps in order to generate intermediate products such as the SAR coherence or the NDVI index in an efficient format such as the Cloud Optimised GeoTIFF. In that direction, we have developed tools that can automate the above mentioned processes.

Funded by the European Union

3.5.1 Downloading tool

The download of a low number of satellite images can take place instantly via the data source interface, without the need of an automated process based on a script. However, when it comes to the download of a time-series of images, the automation of the download process is essential. Thus, we have developed a modular tool, which makes requests to the Umbrella, saves the metadata of the requested products and downloads the latter from the sources that the Umbrella proposes. The tool's steps for harvesting and downloading the asked (meta)data are presented below:

1. Set the parameters of the request e.g. bounding box, mission, time range (Table 3);
2. Construct the request based on the parameters and make the request;
3. Check the response status code;
4. If the response is successful:
   a. iterate over each page of the response and save the metadata into a dictionary;
   b. Save the dictionary into a csv file;
   c. Open the csv and make a request for each product using credentials for the corresponding data source.

Table 3. Available parameters for Sentinel 1, 2, 3 and 5p.

Available parameters	SENTINEL 1	SENTINEL 2	SENTINEL 3	SENTINEL 5P
identifier	✓	✓	✓	✓
relative_orbit_number	✓	✓	✓	✓
polarization	✓	-	-	-
cloud_coverage	-	✓	-	-
tile_name	-	✓	-	-
filename	✓	✓	✓	✓
sensing_date	✓	✓	✓	✓
product_type	✓	✓	✓	✓
wkt_footprint	✓	✓	✓	✓
in_bbox	✓	✓	✓	✓

Funded by the European Union

3.5.2 Pre-processing tool

3.5.2.1 Sentinel-1 GRD Pre-processing

The Sentinel-1 mission comprises of the Sentinel-1A satellite and its twin Sentinel-1B provides Synthetic aperture radar (SAR) data operating in the C-band of the electromagnetic spectrum. Currently, Sentinel-1B does not generate data due to an anomaly occurred in the early morning (UTC) of 23 December 2021. The high frequency of revisit of the Sentinel-1 mission (6 days) and its global coverage along with its active sensing and ability to penetrate clouds make it ideal for all-weather Earth observation. The main acquisition mode of Sentinel-1 for land is the Interferometric Wide (IW) swath mode. Sentinel-1 satellites have a single transmitter chain and can provide dual-polarisation (VV+VH or HH+HV) or single-polarisation (HH or VV) radar data in the IW mode depending upon the geographic location. The Sentinel-1 Level-1 Ground Range Detected (GRD) product is generated from the Single Look Complex (SLC) product by multi-looking and projecting the flange range to the ground range. GRD provides the radar backscatter amplitude and intensity information, while the phase information is not retained. The most of the users make usage of the GRD products with a standard set of corrections applied on them (Filipponi, 2019; Wagner et al., 2021).

The first tool aims at generating VV and VH backscatter coefficients. The whole process is based on ESA SNAP software and specifically in snappy libraries, which acts as an APIbetween Python and SNAP. The steps for delivering the backscatters include the following:

1. **Calibrations**. The procedure of converting digital pixel values to radiometrically calibrated SAR backscatter is called calibration.
2. **Speckle Filtering**. Speckle, appearing in SAR images as granular noise, is due to the interference of waves reflected from many elementary scatterers. Speckle filtering is a procedure to increase image quality by reducing speckle.
3. **Range Doppler Terrain Correction**. SAR data are generally sensed with a varying viewing angle greater than 0 degrees, resulting in images with some distortion related to side-looking geometry. Terrain corrections are intended to compensate for these distortions so that the geometric representation of the image will be as close as possible to the real world.
4. **Conversion to dB**. As a last step of the pre-processing workflow, the unitless backscatter coefficient is converted to dB using a logarithmic transformation.

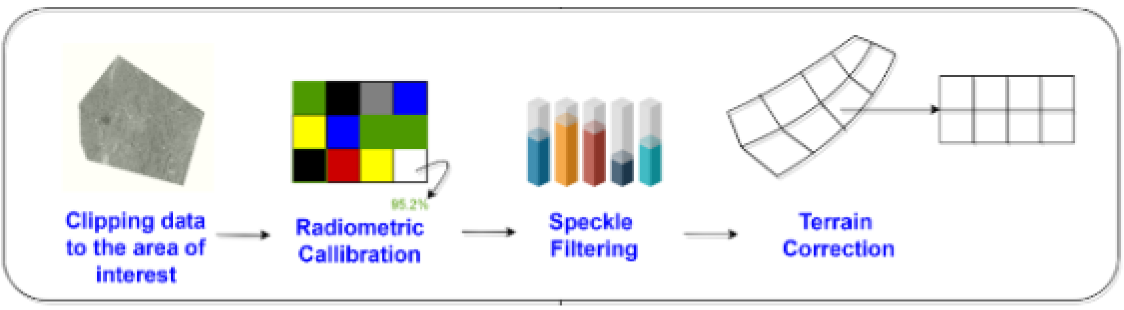

Figure 9. Sentinel-1 processing steps (adapted from Sitokonstantinou et al., 2021).

Funded by the European Union

3.5.2.2 Sentinel-2 preprocessing

The Sentinel-2 mission comprising Sentinel-2A and Sentinel-2B satellites has a combined revisit frequency of 5 days and provides Multispectral imaging (MSI) with 13 optical spectral bands in the visible and near-infrared spectrum at spatial resolutions of 10 m, 20 m and 60 m for each band. The aforementioned bands are stored as jpg-files in the SAFE container The automated pipeline makes usage of the Sentinel-2 Level-2A products, with the latter to have been atmospherically corrected, transforming them from Top-Of-Atmosphere (TOA) Level 1C products to Bottom-Of-Atmosphere (BOA) Level 2A products, using the Sen2Cor software. The Level-2A product contains also, among others, scene classification masks for the detection of dark feature shadow, cloud shadow, vegetation, non-vegetation, water, thin cirrus clouds, medium- and high-probability clouds, snow and ice. The proposed pipeline for pre-processing Sentinel-2 products utilizes Geospatial Data Abstraction Library (GDAL). GDAL allows great flexibility on raster manipulation and high speed on performing raster calculations. The proposed framework applies a series of steps to generate light-weight GeoTIFFs:

1. Resampling of data to 10m spatial resolution,
2. Re-projection to pseudo-mercator projection system.
3. Clipping over an area of interest, if needed
4. Transforming to Cloud Optimized GeoTIFFs (COGs)

Apart from the surface reflectances of Sentinel-2, vegetation indices, e.g., the Normalized Difference Vegetation Index (NDVI), can also be extracted from optical images to enhance certain vegetation characteristics (e.g., water content, physiological stress etc.). Thus, the final step is related to the generation of a series of vegetation indices such as NDVI, NDWI, and SAVI.

4 Application examples and scenarios

The extended Umbrella application enables researchers and service providers to use Sentinel imagery that meet certain meteorological requirements. This combination of satellite and climate data is vital for Earth Systems monitoring, while it opens the door for a more accurate and efficient data analysis, as the Umbrella provides only the most relevant Satellite images, filtering out the irrelevant products. In that direction, we showcase this with 2 example usage scenarios based on the Eiffel pilot needs.

Scenario 1: Water & Land-Use Management of "Aa of Weerijs"

The first scenario utilises the Sentinel-2 mission, while it uses temperature and precipitation in order to decrease the number of Sentinel-2 products above the specified area. Note, the parameters used in this scenario was based on the input provided by D2.3, which reported the pilot requirements. The scenario highlights the difference between the responses returned from the Umbrella by including and not meteorological parameters in the request.

Funded by the European Union

Initially, the scenario imports the required libraries and paths.

Imports and Setup

```
In [1]: import os.path
        import sys
        sys.path.append('/home/userdev/Desktop/eiffel')
        from umbrella_api_utils import search
        import pandas as pd
```

Afterwards, user set the essential parameters such as the bounding box (xmin, ymin, xmax, ymax), the mission, the time range, the meteorological variable and the variable's lower or upper threshold.

Set the parameters

```
In [2]: bbox = [4.68,52.50,5.77,53.58]
        mission =  'sentinel2'
        start_date = '2022-09-01'
        end_date = '2022-10-01'
        meteo_parameter = 'max_temperature'
        threshold = 10
```

As the parameters have been defined, a url is constructed based on them.

Construct the request via a URL

```
In [3]: url = f'http://127.0.0.1:8000/api/products/{mission}?variable={meteo_parameter}&threshold={threshold}&in_bbox={bbox
        url

Out[3]: 'http://127.0.0.1:8000/api/products/sentinel2?variable=max_temperature&threshold=10&in_bbox=4.68,52.5,5.77,53.58&s
        ensing_date__gte=2022-09-01&sensing_date__lte=2022-10-01'
```

Search function exploits the previous url for requesting metadata from the Umbrella API, as a part of useful functions, that perform various utility tasks, ranging from asking metadata, downloading products, writing to files aiming at enhancing the transparency of these operations to the users. As the Umbrella API uses pagination, the search function iterates over the pages to store metadata for the total number of products included in the response. This metadata is cleaned and stored as a series of tuples in the format of (identifier, url).

Search and Save Metadata

```
In [4]: search(url,'aa_products_meteo.csv')
```

Next step is to open the file, store its data via pandas dataframe and filter data so to keep the Level 2A products. The final number of the products meeting the user's requirements are 10.

Funded by the European Union

```
In [5]: products = pd.read_csv('aa_products_meteo.csv')
        products_l2a = products[products['Identifier'].str.contains('MSIL2A')]
        print(f"# Products Found:{products_l2a.shape[0]}")

        # Products Found:10

In [6]: products_l2a.head()
Out[6]:
```

	Identifier	Download Link
8	S2A_MSIL2A_20220927T104821_N0400_R051_T31UEV_2...	https://scihub.copernicus.eu/dhus/odata/v1/Pro...
9	S2A_MSIL2A_20220927T104821_N0400_R051_T31UFU_2...	https://scihub.copernicus.eu/dhus/odata/v1/Pro...
10	S2A_MSIL2A_20220927T104821_N0400_R051_T31UFV_2...	https://scihub.copernicus.eu/dhus/odata/v1/Pro...
11	S2A_MSIL2A_20220929T231601_N0400_R087_T01UBU_2...	https://scihub.copernicus.eu/dhus/odata/v1/Pro...
12	S2A_MSIL2A_20220929T231601_N0400_R087_T01UBV_2...	https://scihub.copernicus.eu/dhus/odata/v1/Pro...

On the contrary, the same procedure without enabling the meteorological factors results in 106 products, showcasing the added value of blending satellite and meteorological data.

Compare the results without using meteo parameters

```
In [7]: url_no_meteo = f'http://127.0.0.1:8000/api/products/{mission}?in_bbox={bbox[0]},{bbox[1]},{bbox[2]},{bbox[3]}&sensi
        url_no_meteo

Out[7]: 'http://127.0.0.1:8000/api/products/sentinel2?in_bbox=4.68,52.5,5.77,53.58&sensing_date__gte=2022-09-01&sensing_da
        te__lte=2022-10-01'

In [8]: search(url_no_meteo,'aa_products_no_meteo.csv')

In [9]: products = pd.read_csv('aa_products_no_meteo.csv')
        products_l2a = products[products['Identifier'].str.contains('MSIL2A')]
        print(f"# Products Found:{products_l2a.shape[0]}")

        # Products Found:106
```

In addition, the following code makes an additional request to the Umbrella having as parameter the precipitation. Umbrella forwards the request for the precipitation values to the locally installed GeoServer instead of the MeteoStat. The whole process is transparent to the user.

```
In [4]: url_geo = 'http://127.0.0.1:8000/api/products/sentinel1?variable=precipitation&threshold=0.5&in_bbox=4.68,52.5,5.77,

In [5]: search(url_geo,'list_geo.csv')
        ...

In [6]: products_geo = pd.read_csv('list_geo.csv')
        products_geo.shape

Out[6]: (11, 2)
```

Scenario 2: Disaster Resilience - Drought, forest fire and pest risk assessment of Finland

The second scenario is related to disaster resilience and specifically to the risk assessment for droughts, forest fires and pests. Therefore, it exploits Sentinel-2 and Sentinel-3 data, along with temperature, precipitation and wind speed. The following notebook, in similar fashion to the previous one, utilises Umbrella application for retrieving Sentinel-3 metadata above Finland based on the maximum temperature variable.

Funded by the European Union

Set the parameters

In [3]:
```
bbox = [25,65,31,70]
mission =  'sentinel3'
start_date = '2022-09-20'
end_date = '2022-10-01'
meteo_parameter = 'max_temperature'
threshold = 4
product_type = 'SR_1_SRA___'
```

Construct the request via a URL

In [4]:
```
url = f'http://127.0.0.1:8000/api/products/{mission}?variable={meteo_parameter}&threshold={threshold}&in_bbox={bbox[
url
```

Out[4]:
```
'http://127.0.0.1:8000/api/products/sentinel3?variable=max_temperature&threshold=4&in_bbox=25,65,31,70&sensing_date
__gte=2022-09-20&sensing_date__lte=2022-10-01&product_type=SR_1_SRA___'
```

Search and Save Metadata

In [5]:
```
search(url,'list1.csv')
```

In [6]:
```
products = pd.read_csv('list1.csv')
products.shape
```

Out[6]:
```
(870, 2)
```

Compare the results without using meteo parameters

In [7]:
```
url_no_meteo  = f'http://127.0.0.1:8000/api/products/{mission}?in_bbox={bbox[0]},{bbox[1]},{bbox[2]},{bbox[3]}&sensi
```

In [8]:
```
url_no_meteo
```

Out[8]:
```
'http://127.0.0.1:8000/api/products/sentinel3?in_bbox=25,65,31,70&sensing_date__gte=2022-09-20&sensing_date__lte=20
22-10-01&product_type=SR_1_SRA___'
```

In [9]:
```
search(url_no_meteo,'list2.csv')
```

In [10]:
```
products = pd.read_csv('list2.csv')
products.shape
```

Out[10]:
```
(1369, 2)
```


Funded by the European Union

5 Limitations and Future Work

The work presented in this deliverable is based on the fundamental assumption that users request data for a limited area of interest, e.g. not an entire continent. Specifically, if the response includes more than 10,000 products, then it automatically prompts users to enter a smaller area of interest. Despite the optimization of the corresponding tables in the database, by adding indexes to specific fields, a time window of some seconds is still required for the Umbrella application to provide a response of a very large area of interest. Another limitation whether data sources provide raw data via an API for free. A series of meteorological APIs do exist, allowing the harvest of historical climate data. However, in most cases the data are not freely and openly available. In addition, other data sources provide free historic data, however, not in an appropriate file formats, meaning that they cannot be accessed on-the-fly.

Future work will focus on providing the coordinates, either as raw data or as a shapefile of the meteorological stations inside the area of interest. This update can be beneficial for the users as it reveals the proximity to certain areas and regions. Furthermore, the climatic data could also be also included in the response, thus, users would be provided with the required Sentinel data, the point areas of the stations to which the filtering conditions were based, as well as the raw climatic data of each station. Lastly, note, depending on the needs and requirements of the Pilots applications in the following months, the Umbrella API may integrate more CC-related data sources.

Funded by the European Union

6 References

Filipponi, F. Sentinel-1 GRD Preprocessing Workflow. Proceedings 2019, 18, 11. https://doi.org/10.3390/ECRS-3-06201

Wagner, W.; Bauer-Marschallinger, B.; Navacchi, C.; Reuß, F.; Cao, S.; Reimer, C.; Schramm, M.; Briese, C. A Sentinel-1 Backscatter Datacube for Global Land Monitoring Applications. Remote Sens. 2021, 13, 4622. https://doi.org/10.3390/rs13224622

Sitokonstantinou, V.; Koukos, A.; Drivas, T.; Kontoes, C.; Papoutsis, I.; Karathanassi, V. A Scalable Machine Learning Pipeline for Paddy Rice Classification Using Multi-Temporal Sentinel Data. Remote Sens. 2021, 13, 1769. https://doi.org/10.3390/rs13091769

Funded by the European Union