OGC Earth Observation Applications Pilot

European Union Satellite Centre Engineering Report

Publication Date: 2020-10-22

Approval Date: 2020-09-23

Submission Date: 2020-08-14

Reference number of this document: OGC 20-038

Reference URL for this document: http://www.opengis.net/doc/PER/EOAppsPilot-SatCen

Category: OGC Public Engineering Report

Editor: Omar Barrilero, Adrian Luna

Title: OGC Earth Observation Applications Pilot: European Union Satellite Centre Engineering Report

OGC Public Engineering Report

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of Contents

Chapter 1. Subject

This Engineering Report (ER) describes the achievements of the European Union Satellite Centre (SatCen) as an application provider in the OGC Earth Observation Applications Pilot and the lessons learned from the project.

Chapter 2. Executive Summary

The objective of the project was for application developers to adapt an application, deploy the application and execute it on different Earth Observation (EO) data and processing platforms.

The main objective was not the integration itself, but to explore what changes are needed in the application to be integrated in a platform, the points that are working well and which ones need to be improved, what solutions/technologies are useful, and what are the limitations to be able to integrate the same application on different platforms.

SatCen has integrated in three platforms an application that computes the coherence of two Sentinel-1 SLC images. During the process, some modifications were needed in the application in order to facilitate the integration. This process and the findings and discussion are explained in detail in this ER.

The main conclusion is that building a platform-agnostic application is feasible: the SatCen application has been integrated in different platforms using the same docker image and following the approach decided during the project. The approach involved the use of Docker for containerization, the Common Workflow Language (CWL) for describing the application, and common metadata.

Although the Pilot can be considered successful, it has also brought to light some points that need to be improved, which are described in detail throughout this document.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Omar Barrilero	SatCen	Editor
Adrian Luna	SatCen	Editor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- **Coherence**

  The coherence refers to the amplitude of the complex correlation coefficient between two SAR images. It is a measure of the similarity of two Synthetic Aperture Radar (SAR) images.

- **Container**

  A standardized unit of software (Docker [https://www.docker.com/resources/what-container]).

- **SNAP Graph**

  A set of operators that define a processing chain in SNAP. SNAP [https://step.esa.int/main/toolboxes/snap/], the Sentinel Application Platform, is common architecture for all Sentinel Toolboxes.

3.1. Abbreviated terms

- ADES Application Deployment and Execution Service
- AI Artificial Intelligence
- AOI Area Of Interest
- AP Application Package
- AWS Amazon Web Service
- BPEL Business Process Execution Language
- CFP Call For Participation
- CWL Common Workflow Language
- DEM Digital Elevation Model
- DWG Domain Working Group
- EMS Execution Management Service
- EO Earth Observation
- EP Exploitation Platform
- ER Engineering Report
- ESA European Space Agency
- FUSE Filesystem in Userspace
- GCP Google Cloud Platform
- GDAL Geospatial Data Abstraction Library
- GUI Graphical User Interface
- JSON JavaScript Object Notation

- MEP Mission Exploitation Platform
- OGC Open Geospatial Consortium
- OWC OWS Context
- REST REpresentational State Transfer
- S1 Sentinel-1
- S2 Sentinel-2
- SAR Synthetic Aperture Radar
- SLC Single Look Complex
- SNAP SeNtinel Application Platform
- TEP Thematic Exploitation Platform
- TIE Technology Integration Experiments
- TOI Time Of Interest
- UI User Interface
- URI Uniform Resource Identifier
- URL Uniform Resource Locator
- VM Virtual Machine
- WKT Well-Known Text
- WCS Web Coverage Service
- WFS Web Feature Service
- WPS Web Processing Service
- WPST Web Processing Service Transactional
- XML eXtensible Markup Language

Chapter 4. Overview

Section 5 introduces the application developed by SatCen to be integrated into the Earth Observation data and processing platforms.

Section 6 presents the lessons learned during the project as an application developer. All the findings and discussions are explained in detail.

Section 7 provides the conclusions and recommendations for future work.

Chapter 5. EO Application Description

5.1. Overview

The application developed by SatCen and integrated into the different platforms aims to identify changes between two Sentinel-1 (SAR) images acquired at different times but acquired with very similar acquisition conditions (sometimes also called interferometric conditions), by computing the "Amplitude" or calibrated backscatter and the "Coherence" between the two images. The coherence refers to the amplitude of the complex correlation coefficient between two SAR images.

The analysis of the coherence is very useful for change detection. Weather and illumination independence make SAR acquisitions particularly interesting for change detection and monitoring scenarios, especially in certain areas of the world or periods of time where weather conditions prevent the use of optical imagery. Additionally, the coherence maps have been found to be of great utility in detecting changes that occur on the ground and that are not detectable by using only the amplitude or backscatter signal.

5.2. Inputs

The inputs of the application are two Sentinel-1 SLC images that should be acquired in the same conditions (same relative orbit). Also, it is possible to define the area of interest in order to compute the coherence only over the desired area. Some other ancillary data is needed, but it is not directly selected by the user: the application automatically will look for some orbit and DEM (Digital Elevation Model) files to download them.

5.3. Processing

The application is written in Python and it uses SNAP [https://step.esa.int/main/toolboxes/snap/] and GDAL [https://gdal.org] internally. The main process for the coherence calculation is carried out in SNAP, but some functions of GDAL are also used.

The main steps in the processing are:

1. Read parameters and check them
2. Split inputs and run SNAP graphs
3. Mosaic the intermediate outputs
4. Generate metadata

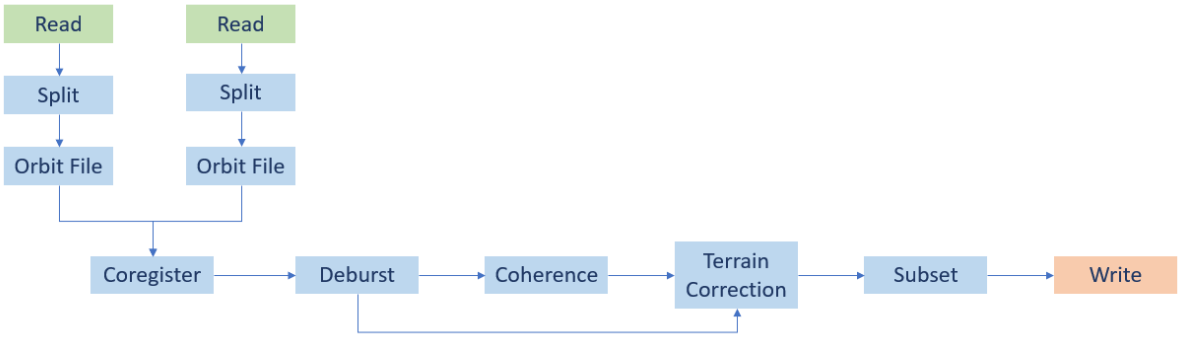

Figure 1. Processing chain

5.4. Outputs

The output is a GeoTIFF image with three bands:

- Image 1 backscatter
- Image 2 backscatter
- Coherence

The output can be the input for other algorithms (e.g. thresholding and algorithms based on Artificial Intelligence, ...) for automatic change detection but is also useful for a visual analysis. A usual Red-Green-Blue (RGB) color composite (R:Image1 backscatter, G:Image2 backscatter, B:Coherence) facilitates the interpretation of the results, since it allows the interpreter to assign the different colors of the pixels to different conceptual categories, for example:

- White: High backscatter in both images and high coherence (e.g. buildings, vehicles). It can be interpreted as "No Change".
- Yellow: High backscatter in both images but low coherence (e.g. changes on buildings, parking lots, containers). It can be interpreted as "Change" or "Activity".
- Red: High backscatter in first image, low backscatter in second image, low coherence (e.g. building destruction, ship in the first image but not in the second). It can be interpreted as "Change", usually a built-up element present on date1 but not on date2.
- Green: High backscatter in the second image, low backscatter in first image, low coherence (e.g. building destruction, ship in the second image but not in the first). It can be interpreted as "Change", usually a built-up element present on date2 but not on date1.
- Blue: Low backscatter in both images but high coherence (e.g. roads, airport).

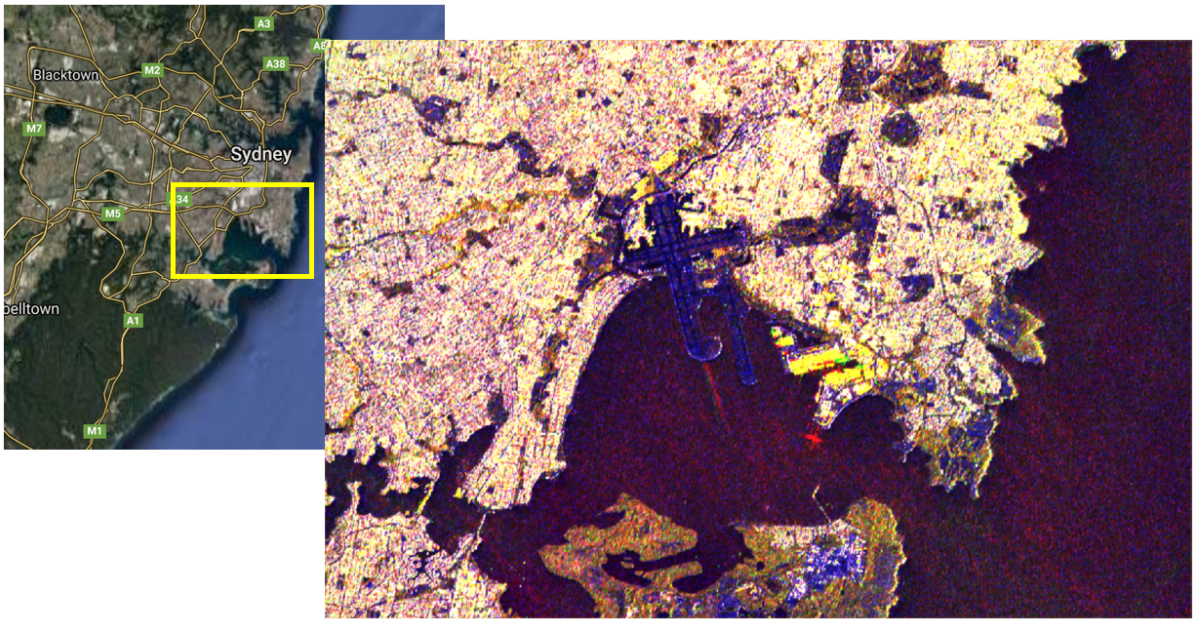

Figure 2. Example of RGB output

5.5. Status

SatCen runs locally the application using Docker but wanted to explore the advantages of moving the application to the data, and the feasibility of creating a single package deployable to many different external platforms.

The main advantages of using a platform are:

- Reduced cost for managing and maintaining IT systems, infrastructure and services.
- More flexible since there is no need for long procurement processes.
- More scalable.
- Pay only for the resources used at every moment.
- Ready-to-use services like security, networking, automatic updates, replication, disaster recovery
- Move the application to the data instead of copying and transferring the products.
- Developers can focus on the application and forget about other issues.

Chapter 6. Findings and discussion during the application integration in the platforms

The steps that the applications providers should follow in order to integrate the application in the platforms are the following:

1. Write/Adapt the EO Application taking into account the parts of it that are specific to the local platform and not relevant when moving to a Platform: data discovery, data access, output data post processing and output data cataloguing.
2. Containerize the EO Application, so that it packs all the dependencies and ensures the application will run on a predictable and isolated environment regardless of in which Platform it is finally running.
3. Register the Docker Image in a registry that is accessible to the Platform. This can include the needs for respecting security rules and access constraints.
4. Describe the EO Application: The process description provides all the information needed by the platform to interact with the application.
5. Deploy the EO Application to the platform.
6. Test the EO Application in the platform.

6.1. Inputs

From the data processing perspective, the inputs required for the EO Application are two Sentinel-1 SLC images acquired with the same geometry. Sometimes, from the user perspective, it is more convenient to use as inputs the date and the area of interest. With this information it is possible to discover the Sentinel-1 SLC scenes in the catalogue, to identify automatically which are the pairs of images that satisfy the user needs and to download them or access them from the platform data archive. In fact, this was the approach followed by the application when working locally: The input from the user are the AOI and the date and the application itself performed the search for the most suitable S1 images; and download them.

One of the first discussions during the project was about the possible approaches for the final EO Application. Different alternatives were considered:

- Manage the data discovery and retrieval directly from within the EO application (keep same approach as locally): With this approach, the application would not take advantage of one of the main reasons for moving to a Platform: "bring the application to the data". In fact, this approach would not use at all the platform catalogue, since it will use a data discovery and retrieval mechanism independent from the Platform (e.g. use the ESA Scihub [https://scihub.copernicus.eu] catalogue with hardcoded or parameterized credentials in the EO Application).
- Adapt the application to each of the platforms specificities with regards to data discovery and access, in order to adapt it to the available datasets and capabilities: This approach would require specific analysis and development for each platform, since data offer and other detail differ between Platforms with regards to data formats, protocols, etc. With this approach, the behavior of the application would be similar to the local deployment, but replacing the ESA

Scihub catalogue by a specific catalogue per platform for supporting discovery and implementing different retrieval methods to support all possibilities.

- Keep the implementation of the data discovery within the application but rely on the platform for the data retrieval. Some complications were reported by different application providers when working with multiple catalogues. A clear example is the difficulty to univocally identify the same product in different catalogues. A typical decomposition of an application consists of the separation into two steps: a first step for data discovery and a second step for data processing (once the platform has made the necessary data available for the application). However, for example, when working with a Sentinel-1 collection, the identifier of each product is different in the different catalogues. Therefore, the application cannot rely on the identifiers to request data at the various platforms.
- Perform an interactive selection of the products in the platform catalogue to launch the application: The application would use as inputs directly the Sentinel-1 products discovered by the user in the platform's catalogue. This approach would guarantee the interactive scenario whereby a user can select the individual inputs he wants to use, while also building the necessary building blocks for future processing steps towards an automated execution following the decomposition explained in the previous point.

The last option was chosen but it is important to highlight this item as one of the most important topics to be further analyzed in the future.

On the other hand, during the integration phase, it was needed to adapt the source code of the application for each specific platform: some preprocessing steps were needed because the inputs were provided in different ways. With the evolution carried on within the project in terms of standardization on the platforms, these adaptations are no longer needed, since the files are now mounted directly in the working directory of the docker automatically. This is indeed a very relevant evolution obtained during the project, since the different platforms and application providers converging into a solution that satisfies everyone is essential in the way to define a standard in the future.

However, it is important to note that the convergence imposes several constraints that might have an impact on the applications, for example in terms of protocol (direct file access, which was the initial expectation of SatCen's application), and file format (since SatCen's application is able to read from a variety of file formats, this had no impact in the developed application). Generally speaking, inputs from the same collection can have different formats. As an example, Sentinel-1 data is usually offered in zip format or SAFE format. Currently, it is not clear if the application should support any possible format or if it should be possible to define the expected format. Then, it would be up to the Platform to provide the correct format, a process that could introduce transformation issues. Alternatively, the Platform could inform the EO Application about its data offerings, protocols, and file formats.

Finally, another point to mention is the evolution of the capabilities of certain Platforms, that were able to rapidly move from versions imposing some constraints to the supported inputs to versions providing a more flexible approach. As an example, the Spacebel platform, during the first test imposed the limitation to only pass the application inputs and parameters as environment variables, while at the end it was more flexible and easier for the application providers, supporting the standard inputs from the CWL definition.

6.1.1. Input constraints

A major topic under discussion within the project regarding the inputs is how to express the necessary constraints that the application imposes to run properly.

Different constraints need to be considered for the application inputs (e.g. acquisition type, cloud coverage, sensor, processing level…), since certain applications are very specific with regards to which inputs they require to run. Additionally, also dependencies between inputs need to be considered. An example of the latter is found in the coherence application, which expects that the two Sentinel-1 products have the same relative orbit and are close in time. Otherwise, the resulting coherence would probably be of little value.

Dependencies between input parameters have not been explored in the pilot in much detail, but initial discussions indicate that the OpenSearch specification seems the perfect standard for expressing these constraints/dependencies. However, it seems complicated to define a "language" allowing applications or application developers to describe those constraints declaratively, i.e. without building an application on its own for that. An alternative solution to be explored in the future would be to define a two-step workflow decomposed with 1) data discovery and 2) data processing, as discussed above.

6.2. Outputs

At the early phases of the project, some constraints were also imposed on the generated outputs. As an example, in one platform, the output name was fixed with an environment variable parameter but with the evolution of the platform the situation improved.

There are open questions, however, about the cataloguing of the outputs. The application needs to generate some metadata in order to provide information about the output to the platform (metadata is discussed later).

This item has a major impact, since the expectation of developers and final users of the applications is to be able to easily discover and access the results of the executions of the applications triggered by them. Also, following the concept of chaining applications, other applications might re-use the outputs. The different platforms provide different tastes on how output results are provided (e.g. via APIs, via Web-UIs).

As a more advanced experiment, a second application whose inputs are the outputs of the coherence application was developed and deployed on Spacebel’s platform. This test was carried out to verify that the data cataloguing is properly done, thus, demonstrating the feasibility of chaining applications. A full end-to-end integration experiment with a set of chained applications was not carried out, though, due to orchestration issues (which application needs to run first?), and lack of an endpoint fully compatible with the CWL specification.

6.3. Packaging

Docker was introduced in Testbed-13 into this EO series of OGC Innovation Program initiatives. By using Docker, the application can be packaged with all its dependencies at the specific versions expected by its developer.

The coherence application was already available in a docker container that was used to run it locally, so this step was not a problem, except for some errors in the work directory in the first version of the docker container generated. It is important to highlight that docker uses the concept of "volumes" to persist data. This aspect requires special considerations when working with large files (as EO imagery). Volumes, though, are provisioned and managed by the Platform and therefore, there is a need to match the docker expectations in terms of size and availability of disk space, and the capabilities of the platform.

Regarding the access to the docker, the coherence application can be uploaded to a public docker repository. If developers need to control access, then a private repository is needed which could be specific for each platform. Further accounting and access control mechanisms are still to be addressed: e.g. how the developer of the application assigns permissions to users to have access to the application.

With respect to security, it is considered a bad practice to execute containers as *root* user. When moving to a platform, this can cause additional challenges when the local version of the Docker was executed as *root*.

6.4. Interface with platforms

An interesting topic is how applications interface with Platforms in order to for example report the status of the execution or communicate errors to the Platform. This topic has not been fully investigated during the Pilot and needs further investigation.

6.4.1. Application Description

Pure containerization of the application in a Docker container is not sufficient to integrate it into any platform successfully. Instead, the application provider must also provide some information to the platform:

- What is the application and what does it do?
- How to run the application?
- What are the inputs?
- What are the outputs?
- Is there any special requirement (e.g. memory, storage, central processing unit)?

The first approach was to use an XML descriptor for the application in Spacebel's platform:

Example XML Descriptor

```
<?xml version="1.0"?>
<wps:ProcessOffering xsi:schemaLocation="http://www.opengis.net/wps/2.0
https://raw.githubusercontent.com/spacebel/common-xml/master/52n-ogc-
schema/src/main/resources/META-INF/xml/wps/t/wps.xsd http://www.opengis.net/ows/2.0
http://schemas.opengis.net/ows/2.0/owsAll.xsd" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:eoc="http://www.opengis.net/wps/2.0/profile/tb13/eoc" xmlns:owc=
"http://www.opengis.net/owc/1.0" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:wps
```

```
="http://www.opengis.net/wps/2.0" outputTransmission="value reference"
jobControlOptions="async-execute dismiss">
    <wps:Process>
        <ows:Title>S1-SLC-Processing Chain</ows:Title>
        <ows:Abstract>S1-SLC-Processing Chain.</ows:Abstract>
        <ows:Identifier>S1-SLC-CCD</ows:Identifier>

        <ows:AdditionalParameters>
            <ows:AdditionalParameter>
                <ows:Name>ImageReference</ows:Name>
                <ows:Value>aluncob/ogc_pilot:0.1</ows:Value>
            </ows:AdditionalParameter>
        </ows:AdditionalParameters>

        <wps:Input minOccurs="0">
            <ows:Title>Master Scene</ows:Title>
            <ows:Abstract>Master Sentinel1 SLC IW scene.</ows:Abstract>
            <ows:Identifier>MASTER_PATH</ows:Identifier>
            <ows:AdditionalParameters>
                <ows:AdditionalParameter>
                    <ows:Name>EnvironmentVariable</ows:Name>
                    <ows:Value>MASTER_PATH</ows:Value>
                </ows:AdditionalParameter>
            </ows:AdditionalParameters>
            <wps:ComplexData>
                <wps:Format default="true" mimeType="application/gep"/>
            </wps:ComplexData>
        </wps:Input>

        <wps:Input minOccurs="0">
            <ows:Title>Slave Scene</ows:Title>
            <ows:Abstract>Slave Sentinel1 SLC IW scene.</ows:Abstract>
            <ows:Identifier>SLAVE_PATH</ows:Identifier>
            <ows:AdditionalParameters>
                <ows:AdditionalParameter>
                    <ows:Name>EnvironmentVariable</ows:Name>
                    <ows:Value>SLAVE_PATH</ows:Value>
                </ows:AdditionalParameter>
            </ows:AdditionalParameters>
            <wps:ComplexData>
                <wps:Format default="true" mimeType="application/gep"/>
            </wps:ComplexData>
        </wps:Input>

        <wps:Output>
            <ows:Title>Output path to the results</ows:Title>
            <ows:Abstract>Output Path</ows:Abstract>
            <ows:Identifier>OUTPUT_PATH</ows:Identifier>
            <ows:AdditionalParameters>
                <ows:AdditionalParameter>
                    <ows:Name>EnvironmentVariable</ows:Name>
```

```
                    <ows:Value>OUTPUT_PATH</ows:Value>
                </ows:AdditionalParameter>
            </ows:AdditionalParameters>
            <wps:ComplexData>
                <wps:Format default="true" mimeType="application/gep"/>
            </wps:ComplexData>
        </wps:Output>

  </wps:Process>
</wps:ProcessOffering>
```

But finally, it was decided to change to CWL as it was going to be supported by all platforms of the Pilot.

The use of CWL is very powerful but also has some limitations for application providers. For example, there are some points that cannot be described in the CWL as input constraints.

Issues arise if platforms support only parts of the CWL specification. Given that not all Pilot platforms use a CWL backend engine, application providers need to know what features are or are not supported (e.g. hardware requirements are supported in some platforms but not all). It is possible to test applications using tools such as for example *cwl-runner*, but this does not guarantee successful execution on the platform.

The base of the CWL file specifies the following: the reference to the Docker Image containing the application (in *hints/DockerRequirement*), the command to be executed (*baseCommand*) and the application identifier, label and description (*id*, *label*, *doc*). Also, the inputs and outputs of the application are described in the *inputs* and *outputs* sections.

Example CWL file

```
cwlVersion: v1.0
class: CommandLineTool
baseCommand: s1_coherence_cd
hints:
  DockerRequirement:
    dockerPull: obarrilero/s1coherence:1.0  # Reference to the Docker Image containing
the application
id: satcen-coherence-spb
label: Satcen S1 Coherence Process (SPB version)
doc: The application computes the coherence of two Sentinel-1 IW products. The output
is a RGB (master backscatter, slave backscatter, coherence) in GeoTiff format.
inputs:         # Definition of the inputs and parameters required by the application
  input_files:
    inputBinding:
      position: 1
      prefix: --input_files
    type:
      items:
      - File
      - Directory
      type: array
  aoi_wkt:
    inputBinding:
      position: 2
      prefix: --aoi_wkt
    type: string?
outputs:         # Definition of the outputs generated by the application
  output:
    outputBinding:
      glob: '*.tif'
    type: File
  metadata:
    outputBinding:
      glob: 'metadata.xml'
    type: File
```

Some small variations had to be made for every platform since one does not support Directory type as input or requires the class workflow to be included. These modifications are quite easy to do and are not big changes, but it would be very useful to homogenize the approaches to have a single CWL file that works on every platform.

6.4.2. Metadata

The metadata is needed to give information to the platform about the outputs in order to be catalogued, given that the platform does not know what the application is doing.

The creation of this metadata could be possible in an automated way if some properties are included in the process description and some tool such as GDAL could be used to discover geo

attributes (considering the output can be opened by GDAL). This approach seems problematic and could not be applicable always, so it was decided to generate the metadata in the application.

A template was provided by the platform providers:

Metadata template

```
<?xml version="1.0" encoding="utf-8"?>
<eop:EarthObservation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xmlns:xlink="http://www.w3.org/1999/xlink"
                      xmlns:gml="http://www.opengis.net/gml/3.2"
                      xmlns:eop="http://www.opengis.net/eop/2.1"
                      xmlns:ssp="http://www.opengis.net/ssp/2.1"
                      xmlns:ows="http://www.opengis.net/ows/2.0"
                      xmlns:swe="http://www.opengis.net/swe/1.0"
                      xmlns:om="http://www.opengis.net/om/2.0"
                      xsi:schemaLocation="http://www.opengis.net/ssp/2.1
../xsd/ssp.xsd">
    <om:phenomenonTime>
        <gml:TimePeriod>
            <gml:beginPosition>$startTime</gml:beginPosition>
            <gml:endPosition>$endTime</gml:endPosition>
        </gml:TimePeriod>
    </om:phenomenonTime>
    <om:resultTime>
        <gml:TimeInstant>
            <gml:timePosition>$generationTime</gml:timePosition>
        </gml:TimeInstant>
    </om:resultTime>
    <om:procedure>
        <eop:processorName>s1_coherence_cd</eop:processorName>
    </om:procedure>
    <om:observedProperty xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
    <om:featureOfInterest>
        <eop:Footprint>
            <eop:multiExtentOf>
                <gml:MultiSurface srsName="EPSG:$epsg">
                    <gml:surfaceMembers>
                        <gml:Polygon srsName="EPSG:$epsg">
                            <gml:exterior>
                                <gml:LinearRing>
                                    <gml:posList>$footprint</gml:posList>
                                </gml:LinearRing>
                            </gml:exterior>
                        </gml:Polygon>
                    </gml:surfaceMembers>
                </gml:MultiSurface>
            </eop:multiExtentOf>
            <gml:locationName></gml:locationName>
        </eop:Footprint>
```

```
    </om:featureOfInterest>
    <eop:metaDataProperty>
        <eop:EarthObservationMetaData>
            <eop:identifier>$identifier</eop:identifier>
            <eop:creationDate>$generationTime</eop:creationDate>
            <eop:parentIdentifier>None</eop:parentIdentifier>
            <eop:acquisitionType>NOMINAL</eop:acquisitionType>
            <eop:productType>s1_coherence_cd</eop:productType>
            <eop:status>ARCHIVED</eop:status>
            <eop:processing>
                <eop:ProcessingInformation>
                    <eop:processingCenter></eop:processingCenter>
                    <eop:processingDate>$generationTime</eop:processingDate>
                    <eop:method>s1_coherence_cd</eop:method>
                    <eop:processorName>s1_coherence_cd</eop:processorName>
                    <eop:processorVersion>$version</eop:processorVersion>
                    <eop:nativeProductFormat>S1</eop:nativeProductFormat>
                    <eop:processingMode>NOMINAL</eop:processingMode>
                </eop:ProcessingInformation>
            </eop:processing>
        </eop:EarthObservationMetaData>
    </eop:metaDataProperty>
</eop:EarthObservation>
```

The application shall generate the corresponding metadata by using this template.

It is also supported by some of the platforms to provide a key-value text file.

6.4.3. Deployment

The deployment of the coherence application has been different for every platform:

- Spacebel: By using the web graphical interface by uploading the CWL file.
- CRIM: By using the REST API in three steps:
  - sign in
  - deploy
  - set visibility
- Terradue: By providing the file to the platform providers manually.

All three approaches are valid and can be done easily. When the deployment is carried out by the platform provider, it is easier for the application developers, but deployments or updates of the application depend on the availability of the platform support team.

6.4.4. Hardware requirements

During first executions it was noticed, in the log files, that some memory errors were caused by applications with high memory demands that were not matched by the platforms by default.

The application needs a minimum amount of memory but also can be customized in order to be able to work with less or more memory depending on availability.

During the integration phase, some limitations for providing this information to the platforms have been found. Although the CWL language allows to provide information about the required hardware, not all the platforms support these features.

```
requirements:
  ResourceRequirement:
    ramMin: 24000
```

Ideally, the platform could also provide more memory so that the application can take advantage of all available memory. This is currently not available in the platforms.

6.4.5. Execution

Once the application is deployed, the execution is very easy in all platforms via web interfaces or REST calls.

One limitation that has been experienced is the selection of input data because not all data was available on all platforms. This is a limitation during the pilot, but when the platforms are fully operational this shouldn't be a problem since the application developers will deploy their applications only where the needed datasets are available.

6.4.6. Communication

During the course of the pilot, some limitations have been noticed related to the communication between user-application-platform:

- logs: the level of logs is sometimes not sufficiently detailed
- progress: there is no standard way to communicate the progress of the application to the platform
- available resources: the application does know the maximal available resources and thus cannot optimize its performance

6.5. Technology Integration Experiments (TIEs)

The integration experiments consisted of the manual execution of the application using two Sentinel-1 products as input.

6.5.1. Spacebel

The experiment was carried out using the graphical web interface that is available for the platform. For the selection of the input, it is possible to upload files or to use the tools for selecting products in the platform. There are some limitations:

- Data availability on the platform: there are not a lot of products available, but since this was a

demo, it is not a problem.

- The queries for looking for products does not support useful search criteria for coherence application, like the relative orbit.

Once the products and the AOI are selected, the execution works properly and the results are as expected. It is very easy to download both outputs and log files, which makes it easy to analyze any issues.

6.5.2. Terradue

The deployment was carried out by Terradue, since in this case, it is a process managed by the platform operator. Terradue provided access to the GUI in order to be able to run the application. The execution completed successfully, but an issue in the platform does not allow it to publish the results, so the output cannot be checked.

6.5.3. CRIM

The deployment of the application, through the API, was successful. The process for the execution seems to work properly but no output result was produced because of a limitation with the inputs. The platform does not contain the Sentinel-1 products so a URL can be used to provide them to it. The SciHub URL was provided but the platform does not support the "$" character which is present. No more tests with other inputs could be carried out.

Chapter 7. Conclusions/Recommendations

7.1. Building platform-agnostic applications is feasible

It has been demonstrated during the project that it is possible to create a single docker image containing a simple application in order to be integrated into different platforms with no change in the docker whatsoever.

SatCen's application has been integrated in Spacebel, Terradue and CRIM platforms. At the beginning, in order to start interacting with the platforms, different Docker-based containerization approaches were used, but during the project the different approaches started to converge and finally the main objective of using the same containerization approach in all the platforms was achieved.

The collaboration with partners, multilateral and bilateral meetings and the will to reach agreement allowed the participants to make significant progress during the pilot.

7.2. EO App Pilot outcomes

The Pilot is perceived as very useful for advancing and validating certain aspects in real-life scenarios.

The use of docker for packaging the applications with all its dependencies has been demonstrated to be very useful. The status of this technology, its widespread use and the numerous resources and examples makes Docker the recommended solution.

The Docker containing the application is not enough to integrate the application into the platform. What is also needed is a description of the application that tells the platform how to execute it, what are the inputs, the outputs, and additional parameters (e.g. hardware requirements or other specifics of the application).

With respect to provisioning application descriptions to the platform, the use of CWL has been recommended and implemented by all the partners. Though, some differences in the level of CWL support caused some integration issues. CWL is very powerful and although not all features are currently implemented in the platforms, it is perceived as a suitable, powerful, and extensible solution. It has substituted the *WPS Process Description*, which is a lot more verbose. Though the *WPS Process Description* has some other advantages, CWL is considered a simpler solution for application developers. Further investigations shall evaluate the differences in more detail regarding definition of hardware requirements and input constraints in particular.

Still, it is necessary to review CWL in contrast to other workflow orchestration engines that exist either platform independent or for specific platforms. Examples include Airflow, Argo, or Pachyderm. The analysis should include compatibility issues and limitations of the various orchestration engines in particular with respect to support cloud environments such as e.g. Kubernetes.

The platform (ADES) shall manage both discovery and retrieval of input data as discussed previously in this report. A deeper analysis is needed in order to address current interoperability

challenges:

- OpenSearch capabilities and metadata differ: the product ID could be different depending on the catalogue.
- Access protocols (http://, ftp://, file://, s3://, etc.)
- Product formats (it could be different depending on platform)
- Expressing input constraints

SatCen opted for an interactive selection of the inputs when running on the various Pilot platforms. Ideally, this approach should be replaced by a two-step process (first discover the data and finally run the algorithm).

Taking a look at the expected advantages of deploying applications on a remote platform compared to in-house deployment and execution, the following aspects shall be named:

- More flexible: This point seems to be achieved. Currently, the application can be deployed in different platforms without major changes. Only some minor changes in the application descriptors are needed.
- More scalable: When deployed on a platform, it is possible to scale up easily and multiple processes can be executed simultaneously.
- Platform integration issues: If it is necessary to process some applications on different platforms and then compile all outputs to be used as input for another application, cross-platform compatibility issues may arise, including data cataloguing and discovery of results
- Ready-to-use-services: Moving to a cloud-platform allows developers to focus on the actual EO application while benefiting from the additional offering of additional services from the Platform. However, the analysis of those services was out of the scope for the pilot and could be analyzed in a future activity. In this context, the following aspects shall be explored:
  - Authentication
  - Accounting
  - User management
  - Processing models
- Process close to the data: The application has been executed successfully with the data available directly on the platforms with data selection in an interactive mode. It is not easy to implement the discovery of the data in a platform-agnostic manner.
- The EO Application developer focuses only on the application: Moving the application to the Platform allows the EO Application developers to focus on the implementation and integration of the processing algorithm, rather than devoting time and effort in putting in place an infrastructure and all its building blocks. During the Pilot, there has been good communication and availability from the platform providers to solve any issues that might arise during the integration. Therefore, the expectation is considered successfully achieved, since we could mainly focus on the algorithm and its integration.

7.3. Recommendations and Future work

A critical review of the achievement agreements has to be done in order to be sure that the proposed solutions, besides meeting the needs of application developers and platforms providers, are the best taking into account current technologies.

There are some points that have not been discussed in detail during the pilot but are interesting for the application developers. More tests and discussions are needed for:

- Communication user-application-platform: for progress reporting and provisioning of platform resources to the application and vice versa
- Chaining applications: both, inside a platform and from different platforms
- Cross-Platform applications and compatibility (including scaling-out workflows)
- Data discovery: implementation in a platform-agnostic manner

Appendix A: Revision History

Table 1. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
July 31, 2020	Omar Barrilero and Adrian Luna	1.0	all	initial version
August 5, 2020	I. Simonis	1.1	all	full review
October 15, 2020	G. Hobona	1.2	all	final staff review