

OGC Earth Observation Applications
Pilot
Summary Engineering Report

Publication Date: 220-10-26

Approval Date: 2020-09-23

Submission Date: 2020-08-10

Reference number of this document: OGC 20-073

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

Category: OGC Public Engineering Report

Editor: Ingo Simonis

Title: OGC Earth Observation Applications Pilot: Summary Engineering Report

OGC Public Engineering Report

COPYRIGHT

Copyright © 2020 Open Geospatial Consortium. To obtain additional rights of use, visit <http://www.opengeospatial.org/>

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

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

1. Subject	5
2. Executive Summary	6
2.1. Objectives	6
2.2. Document contributor contact points	6
2.3. Foreword	7
3. Introduction	8
3.1. Background	9
3.2. Results and Videos	10
4. Participants	12
5. Results	13
5.1. Open Standards	14
5.2. Application Patterns	14
5.3. Application Package and CWL	15
5.4. ADES and EMS Deployment	15
5.5. High-level Architecture	16
5.6. Data Discovery	16
5.7. SpatioTemporal Asset Catalog (STAC)	16
5.8. Container Format	16
5.9. Quotation and Billing	16
5.10. Workflows	17
5.11. Graphical User Interfaces	17
5.12. Kubernetes Cluster Support	17
6. Future Work	18
6.1. APIs	18
6.2. Application Package Description and Workflow Building	18
6.3. Application Handling	19
6.4. Data	19
6.5. Docker	19
6.6. General Guidelines and Performance	20
Appendix A: References	21
Appendix B: Terms and definitions	22
B.1. Technical terms	22
B.2. Abbreviated terms	22
Appendix C: Revision History	24

Chapter 1. Subject

This Engineering Report (ER) summarizes the main achievements of the OGC Innovation Program initiative [Earth Observation Applications Pilot](https://ogc.org/earth-observation-applications-pilot), conducted between December 2019 and July 2020.

Earth Observation Applications Pilot

Deployable applications for
Earth Observation platforms

ogc.org/earth-observation-applications-pilot



Chapter 2. Executive Summary

This Engineering Report (ER) summarizes the main achievements of the OGC Innovation Program initiative [OGC Earth Observation Applications Pilot](#), conducted between December 2019 and July 2020. The pilot explored an Earth Observation Application software architecture that was developed in OGC Testbeds 13-15. The architecture allows the deployment and execution of externally developed applications on Earth Observation (EO) data and processing platforms. The architecture is essentially based on three major components:

- Execution Management Service (EMS): This component provides a RESTful interface (defined using OpenAPI) to register applications and build workflows from registered applications. The EMS selects the appropriate ADES platform to execute the processes based on the runtime input parameters (close to the data).
- Application Deployment and Execution Service (ADES): This component allows deployment, discovery, and execution of applications or processing of quoting requests.
- Applications are delivered in the form of Docker images along with corresponding metadata called an Application Package (AP). The application package provides all information for deployment and execution of an application.

The pilot demonstrated that the interoperability arrangements developed and documented in OGC Innovation Program initiatives Testbed 13, 14 and 15 provide a solid starting point for maturity tests within operational platforms. Additional arrangements and more detailed definitions have been developed during the pilot that now allow deployment and execution of an application on various platforms with minimal adaptations.

The pilot produced very valuable results. It confirmed the general approach to use Docker for application packaging and HTTP Web APIs or Web Services for application handling and execution. It defined application patterns based on data inputs/outputs, confirmed the role of the Common Workflow Language (CWL) for application description, execution, and workflow building; and recommends the usage of the SpatioTemporal Asset Catalog (STAC) as a data manifest for application inputs and outputs.

2.1. Objectives

The Earth Observations Applications Pilot explored and evaluated the maturity of the Earth Observation Applications architecture. The architecture has been developed in various OGC Innovation Program initiatives over the last three years. This pilot now explored everything in operational environments.

2.2. Document contributor contact points

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

Contacts

Name	Organization	Role
Ingo Simonis	OGC	Editor

2.3. 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. Introduction

This Engineering Report (ER) summarizes the main achievements of the OGC Innovation Program initiative [Earth Observation Applications Pilot](#), conducted between December 2019 and July 2020. The [Call for Participation](#) was published in December 2019. The actual execution phase lasted from January to July 2020.



The growing volume and wide availability of Earth Observation data together with affordable cloud computing resources creates an opportunity for the wide adoption and use of Earth Observation (EO) data in all fields of our society. At the same time, petabyte-sized distributed data storages, long lists of complex data products, and continuously evolving processing chains, tools, and models pose major challenges for software architects. Since it is clear that the classic linear, one-directional model of "data provider", download, "data consumer", "analysis and representation" had served its time in the context of EO data processing, software architects encapsulate application logic into executable units that can be deployed and executed on platforms. Platforms provide all pieces required by application developers "as a service", i.e. Data-as-a-Service, Storage-as-a-Service, Computing-as-a-Service, Infrastructure-as-a-Service etc. Thus, platforms allow the application developers to concentrate on their core business. Platforms reduce the time to market, reduce investment costs, and open access to wide ranges of potential customers.



The Earth Observation Applications architecture is fully aligned with this paradigm shift from "bring the data to the user" (i.e. user downloads data locally) to "bring the user to the data" (i.e. move user exploitation to hosted environments with collocated computing and storage). The pilot explored several operational platforms that provide infrastructure, data, computing and software as a service. These platforms allow scientific and value-adding activities and generate targeted outputs for end-users.

3.1. Background

OGC activities in Testbed-13, Testbed-14, and Testbed-15 initiated the development of an architecture to allow the ad-hoc deployment and execution of applications close to the physical location of the source data with the goal to minimize data transfer between data repositories and application processes (see [References](#)). The various Testbeds produced several draft specifications, which addressed both application description and discovery, APIs for deployment, execution, and result access, billing and quotation models, as well as specifications for service chaining and workflow building.

In summary, the architecture fulfils the following requirements:

- Decouple application developers from platform operators from application consumers
- Allow application developers to make their applications available on any number of platforms with minimal modifications
- Allow application developers to focus on application development by minimizing platform specifics and particularities
- Enable platforms to support any type of EO application that works on platform or external data
- Allow chaining of applications to build complex workflows

The following figure provides a high-level view on the resulting setup. Application developers on the left side develop their application and make it available in the form of a Docker container image. The developers register their application in the platform and thus make it available to all users of that platform. At the same time, application developers can discover other applications and integrate these into workflows, where the output of one application serve as input into the next application.

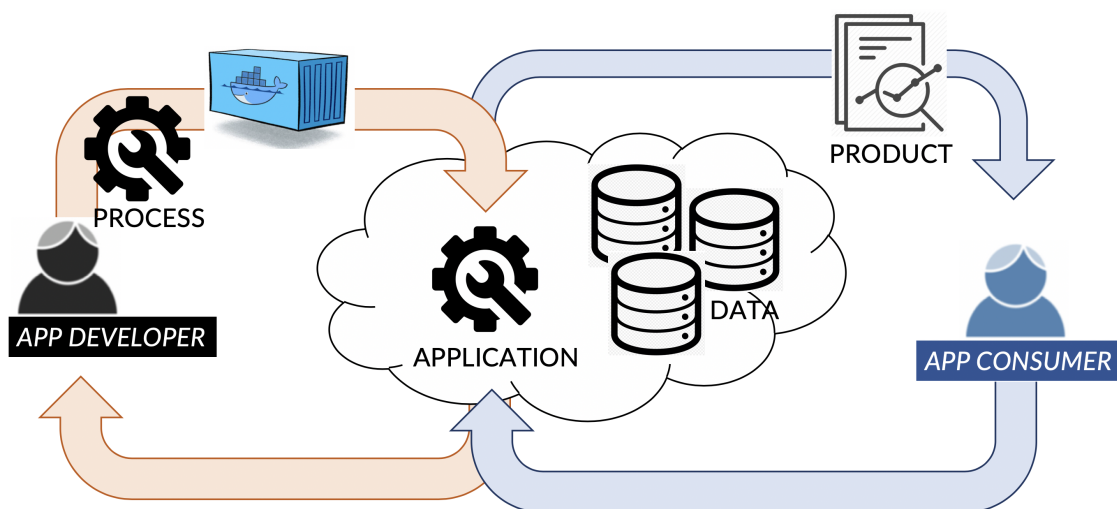


Figure 1. Application developers and application consumers interacting with the cloud platform

Platform providers offer APIs that allow both, application developers as well as application consumers to interact with the platform. All interaction, including application registration, deployment, and execution requests, are supported by a Web Processing Service (WPS) interface or an interface conforming to the [OGC API - Processes](#) candidate standard. To differentiate the various interactions, two logical services have been defined: The Execution Management Service (EMS), and

the Application Development and Execution Service (ADES).

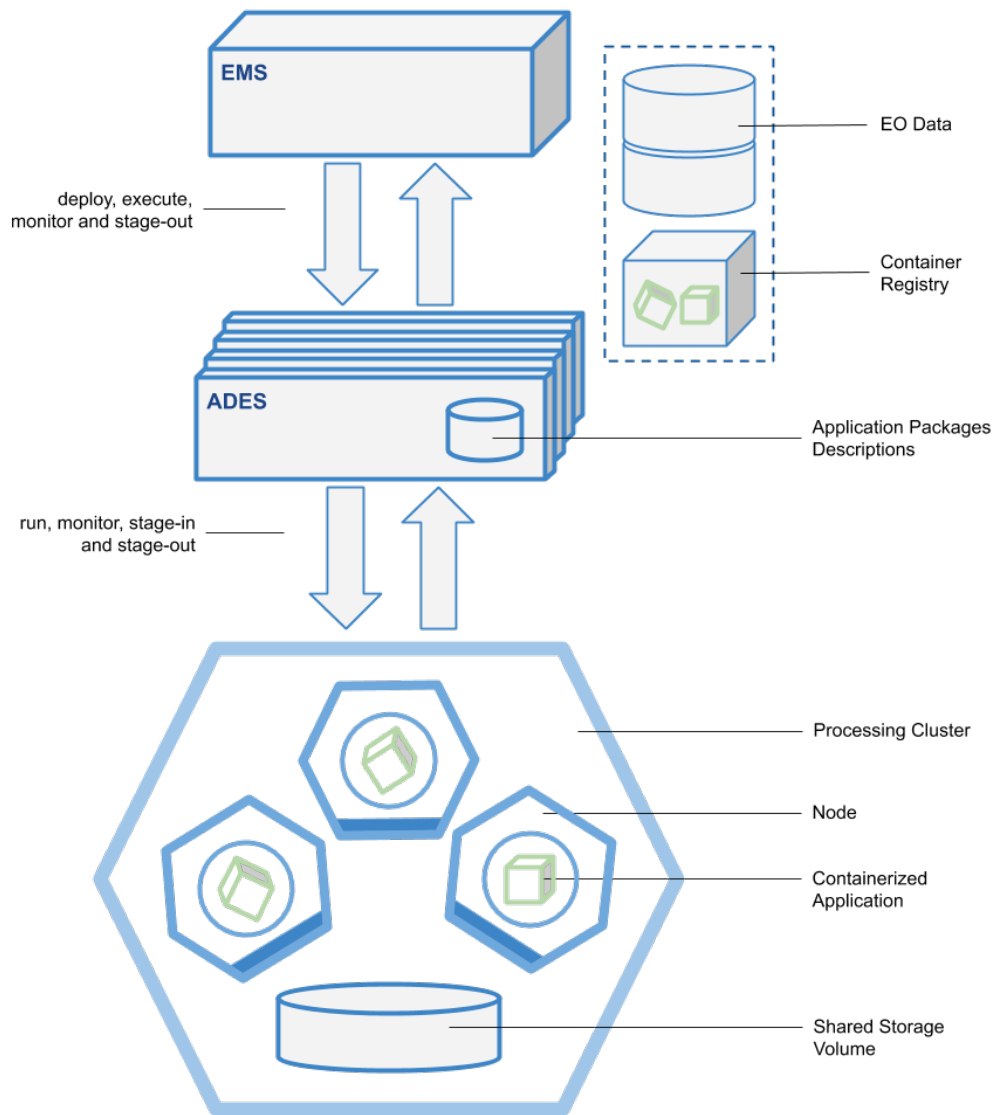


Figure 2. Platform High-level Architecture (source: [Terradue](#))


Together, both APIs provide all functionality necessary to handle the different types of interactions between application developers and the platform (mostly registration and update of applications) and application consumers and the platform (mostly application discovery and execution).

The [Terradue](#) report provides a good introduction into ADES and EMS, as do the reports by [CRIM](#) and [Spacebel](#).

3.2. Results and Videos

All results have been captured in detailed Engineering Reports. Each participant delivered a report that documents all detail about the various applications and platforms, describes interoperability challenges, and recommended solutions.

In addition to the Engineering Reports, all participants produced promotional videos that are available on the OGC Youtube channel:



OGC Earth Observation Applications-to-the-Data Architecture

8 videos • 133 views • Last updated on Aug 26, 2020

OGC opengeospatial

YouTube GB

- OGC EO Apps pilot - Applications
www.CRIMca
4:43
- OGC EO Apps pilot - Platforms
www.CRIMca
4:18
- Pixalytics: EO Applications Architecture for Application Developers - Docker Deployment
opengeospatial
13:48
- The EOxHub platform perspective on OGC EO application packages
opengeospatial
5:58
- Terradue: Earth Observations Applications Architecture
opengeospatial
5:01
- Spacebel Video EO Apps Pilot
opengeospatial
2:08
- EO Applications Architecture for Application Developers - Urban app deployment and execution (DLR)
opengeospatial
6:49
- EU Satellite Center: Application Integration in EO Platforms
opengeospatial
7:27

Figure 3. OGC Youtube Channel: <https://www.youtube.com/playlist?list=PLQsQNjNIDU84GcHzFzUCFGOpAZukuRcTm>

Chapter 4. Participants

The following table lists all organizations that participated in the pilot.

Table 1. List of participants and corresponding Engineering Reports

Organization	Role	Engineering Report
<i>Computer Research Institute of Montreal (CRIM), Canada</i>	Applications + Platform	http://docs.opengeospatial.org/per/20-045.html
<i>European Union Satellite Centre, Spain</i>	Applications	http://docs.opengeospatial.org/per/20-038.html
<i>EOX Consortium: EOX, Austria; DLR, Germany, University of Western Timisoara, Romania; Terrasigna, Romania; Sinergise, Slovenia</i>	Applications + Platform	http://docs.opengeospatial.org/per/20-043.html
<i>Pixalytics Ltd, UK</i>	Applications	http://docs.opengeospatial.org/per/20-037.html
<i>Spacebel s.a., Belgium</i>	Platform	http://docs.opengeospatial.org/per/20-034.html
<i>Terradue Srl</i>	Applications + Platform	http://docs.opengeospatial.org/per/20-042.html
<i>European Space Agency</i>	Sponsor + Applications	-
<i>Natural Resources Canada</i>	Sponsor + Platforms	-
<i>Telespazio VEGA UK Ltd</i>	Sponsor (on behalf of the European Space Agency (ESA))	-
<i>RHEA</i>	Sponsor (on behalf of the European Space Agency (ESA))	-

Chapter 5. Results

The pilot explored the EO Applications Architecture in high detail. The following paragraphs discuss the main findings, as reported throughout the detailed reports as listed in [participants listing](#). Additional insights are provided by ESA/Rhea. Details about the various applications and platforms are provided in the individual reports. Overall, there was strong agreement that the high-level approach with *Application Package*, *Execution Management Service*, and *Application Deployment and Execution Service* is a solid approach to realize an efficient Earth Observation Applications Architecture. Combined with standardized Web APIs and container solutions, the Earth Observation Applications Architecture allows for rapid, cost efficient deployment of the key components and high levels of interoperability across platforms and applications. The stringent usage of open standards reduces adaptation costs for application developers, because only minimal changes need to be applied to applications for deployment on various platforms.

More precisely, the following advantages of the standards-based EO Applications Architecture compared to either in-house or non-standards based approaches have been named repeatedly:

- **Flexibility:** It has been demonstrated that the same application can be deployed in different platforms without major changes. Only some minor changes in the application descriptors had to be applied. The different platforms offered various application deployment schemas: Spacebel offered a web-based user-friendly (GUI) that allowed participants to deploy, run, and monitor their own app autonomously. CRIM provided a command line interface (CLI) to their platform where a more expert user can deploy and interact with the platform. Both EOX and Terradue handled the deployment and execution of applications without offering user interaction interfaces. All three approaches are valid and resulted in different levels of user experiences.
- **Robustness:** The pilot explored two types of interface designs. Some implementations of the OGC Web Processing Service (WPS) followed the Service Oriented Architecture (SOA) design, whereas others implemented the emerging OGC API-Process specifications. It has been demonstrated that the underlying principle does not cause any substantial modifications to the overall design. Thus, the transition path from SOA-based WPS to OGC API based implementations is a rather natural process. It has most consequences on implementation efficiency and cost, which are substantially reduced with OGC API-based approaches.
- **Scalability:** By deploying on a platform, it is possible to make use of the cloud-based advantages offered by the platform: Easy up-scaling and simultaneous execution of multiple processes.
- **Cost-efficiency:** Applications are deployed and executed on a per-use basis, thus do not cause hardware costs when not in use.
- **Ready-to-use-services:** Moving to a cloud-platform allows developers to focus on the actual EO application while benefiting from the additional offerings from the platform. These include, but are not limited to:
 - Authentication
 - Accounting
 - User management
 - Processing models

- **Application Performance:** Processes are deployed and executed physically close to the data: Applications had been executed successfully with data available directly on the platforms and data selection in interactive modes.
- **Focus:** EO application developers can focus exclusively on the application itself: Moving the application to the platform allows the EO Applications 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.

Experiences show that the deployment and execution of applications on remote platforms was not a smooth experience from the start of the pilot, but improved substantially once additional interoperability arrangements had been set. Still, the successful execution of an application sometimes failed due to missing input data (data not offered by platform) or simply stumbled over non-supported URL characters. All these experiences show the necessary level of detail in future standards to guarantee interoperability between application developers and platform providers. Selected results, usually shared across most participants, are documented as follows.

5.1. Open Standards

The pilot was executed during the transition phase from the Service Oriented Architecture period towards the modern Web APIs. The pilot has demonstrated that the transition is in fact very smooth. Though it is essential to use a limited set of open standards to achieve interoperability, the choice of JSON encodings vs. XML payloads and consistent usage of REST principles vs. SOA-remote procedure calls has effects on implementation costs, but less on achievable interoperability.

On a more detailed level, the pilot has proven that the path towards a harmonized set of Web API building blocks rather than a service-oriented concept with disjunct functionality bundled in a concrete service is very promising. It needs to be implemented carefully with keeping both interface design as well as resources representation and serialization in mind. It needs to be avoided that a variety of resource models will have negative effects on the level of interoperability and thus perceived performance from both application developers' and application consumers' perspectives.

It has been demonstrated that open standards are required for all steps of the process. This is in particular true for the platform-internal processes of data provisioning and result handling and their respective discovery. For application developers, these steps have the highest risk for interoperability issues and thus unsuccessful application execution. Platforms shall provide open standards based discovery and data access mechanisms independently of their internal data storage and management system. Only then application developers can deploy their apps on multiple platforms without adapting these crucial parts of their applications from one platform to the next.

5.2. Application Patterns

General agreement was achieved on the definitions of data processing design patterns.

The data driven application fan-in patterns refers to the execution of a data processing function that aggregates several input products. The platform application accesses a list of input products, retrieves and proceeds with the stage-in of the products making them available to the application

execution block.

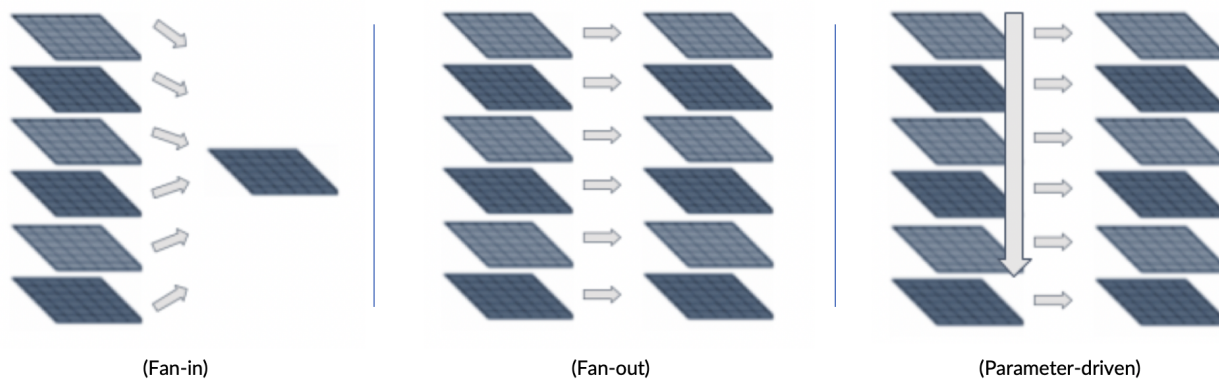


Figure 4. Application data-driven design patterns (source: Terradue)

The data driven application fan-out patterns refers to the execution of a data processing function that processes concurrently several products generating independent output for each input. The platform application loops from a list of input products, retrieves and proceeds with the stage-in of the individual products making them available to the application execution block. The platform can apply different strategies to parallelize the execution of each individual product.

Parameter-driven data flows permit cyclic, systematic retrieval of selected groups of input products between selected the parameter intervals (e.g. start and end dates). In this scenario, the parameter interval acts as a step function, determining how the next batch of products is to be selected.

5.3. Application Package and CWL

The Application Package (AP) contains all information necessary for application platforms to deploy and execute the application correctly. These include information about the application process itself, input and output requirements, dependencies, or hardware requirements. There was common agreement on the items that the AP needs to include, but some discussion about its serialization format. Overall, there was a clear tendency towards the adoption of the Common Workflow Language (CWL) for that purpose. Only Spacebel emphasized the need for [complementary approaches](#). Further work shall identify the essential CWL elements and clarify the use of the CWL specifications [Command Line Tool](#) and [Workflow Description](#). See [Terradue's report](#) for a mapping of CWL elements to the WPS data model and CWL element usage in general. [EOX](#) on the other hand reported a necessary use of heuristics for properly translating CWL types to the corresponding WPS types and vice versa. There was common agreement to add hardware parameters to the Application Package.

Further on, supporting a well-established declarative format like CWL to define application workflows gives platform providers the possibility to leverage a wide ecosystem of existing tools. These can be used by the platforms themselves as well as offered to application developers (e.g. to compose workflows visually) or to application consumers (e.g. to monitor workflow execution).

5.4. ADES and EMS Deployment

CRIM successfully deployed the same core ADES-EMS implementation in three different sites, PBC, EODMS and CRIM and reported only minor adaptation requirements (mostly to deal with different

security environments and settings). The use of an application profile to register external WPS proves to be a rather simple, but useful tool to build a federated cloud.

5.5. High-level Architecture

The separation of roles between EMS and ADES is seen as beneficial, the former being closer to thematic platforms with cross-cutting concerns, the latter located near mission data, or more specifically to the ground segment.

5.6. Data Discovery

Testbed-14 introduced the OpenSearch API for selecting the EO data inputs submitted to the process. Experiences have shown that, despite some inconsistencies among OpenSearch implementations, OpenSearch provides the most promising approach to handle data discovery aspects across platforms. This is particularly important as different platforms use different identifiers for the same product and make the data available in different formats (zipped, unzipped, folder, tar-ball, SAFE etc.).

5.7. SpatioTemporal Asset Catalog (STAC)

The data flow management plays a prominent role in this architecture. Most applications do not bring their own data, but require data being made available by the platform. In addition, applications produce a variety of output data as result. Thus, it is essential to describe both the inputs and outputs of an application in full detail. In the pilot, it turned out that data flow management by using local catalogues encoded in compliance with the SpatioTemporal Asset Catalog (STAC) specification as a data manifest for application inputs and outputs is a very promising approach. STAC, a profile of OGC API-Features and thus based on open standards, blends in very well with the set OGC API building blocks and design patterns.

For a detailed discussion of STAC and its integration with CWL see [Terradue's report](#). [Spacebel](#) argued that STAC would introduce additional, unnecessary conventions that that could be avoided by using OGC data access services such as OGC Coverage (WCS) and Feature (WFS) Services or their OGC API counterparts.

5.8. Container Format

There is common agreement that Docker is the state-of-the-art container format to be used in the EO Applications Architecture. Its maturity, widespread use, and the numerous resources and examples make Docker the recommended solution. Experiences differ in terms of optimal Docker image handling, which led Spacebel to run experiments with server-side production of Docker images; mostly triggered by security concerns (see [Spacebel report](#)).

5.9. Quotation and Billing

[CRIM](#) noted that producing accurate quotations for workflows will be a very difficult challenge, while quoting for applications in simple federations is probably feasible.

5.10. Workflows

Experiments with CWL workflows have been successfully conducted.

5.11. Graphical User Interfaces

As has been shown that the availability of GUIs, ideally paired with directly accessible log files, is an essential ease-of-use cornerstone for application developers.

5.12. Kubernetes Cluster Support

Although Earth Observation platforms are typically installed in a Cloud infrastructure, the implementations in previous OGC Testbeds had limited the execution on a single node machine. This considerably restricts the system scalability and computing possibilities. Kubernetes has become the de facto standard for deploying containerized applications in private and public cloud environments. This pilot experimented with Kubernetes clusters and explored the options to execute atomic tasks in parallel. Experiences have been very promising so far. The automated scalability of the cluster nodes allows to extend the infrastructure by provisioning additional machines when multiple processes are requested or when single requests can be split into parallel tasks. This approach allows to execute a very high number of concurrent jobs and exploits advantages of cloud platforms.

Chapter 6. Future Work

The following general recommendations for future work items have been made. Please consult the individual reports for further details.



6.1. APIs

OGC API-Processes

Align the architecture with the emerging OGC API-Processes, which will complement the WPS standard in the near future.

Standardize ADES and EMS

Conformance classes for ADES API shall be standardized.

6.2. Application Package Description and Workflow Building

CWL and Application Packages

Define subsets of CWL that shall be supported by all platforms to ensure consistent experiences for application developers. Produce an *OGC Best Practice* document to document the use of Common Workflow Language for application packages. The proposed Best Practice document should cater for the various application design patterns discussed in this pilot and provide guidelines for automated generation of CWL (e.g. from Jupyter Notebooks).

In addition to CWL, other approaches shall be reviewed again, such as a [simple command line syntax](#) to describe the base command and corresponding arguments for Docker container execution.

The usage of different CWL versions may lead to issues in the future and should be addressed.

Workflows

More tests are required to better support application workflows running in multiple platforms. In order to build a federated cloud, these tests have to run regularly, in a structured and systemic fashion.

6.3. Application Handling

Logs, Status, Progress, and Error Reports

Currently, platforms provide individual settings for reporting progress, management of logs, and handling of errors. These should be further harmonized.

6.4. Data

Data discovery and access is among the biggest challenges for application developers. All platforms use slightly different approaches. A consistent usage of data access services that are offered by all platforms might facilitate this problem.

OpenSearch and Data Discovery

OpenSearch has been reported as both being very powerful for data discovery, but still not fully consistent across the various platforms. Here, further tests and derived standards shall clarify both OpenSearch usage and provisioning of results.

Metadata

Extend the set of provided metadata to allow platforms to describe their hard- and software capabilities. Explore how metadata can be generated more automatically rather than being based on application developer inputs.

From the application perspective, it shall be investigated how the application package can register the supported input/output formats and file access mechanisms (e.g. direct access S3, http(s), GDAL virtual file systems, etc.), so that platforms can choose the optimal match between platform and application and potentially skip staging in data or outright reject the package if no compatible format is available.

Use of Data Access Services and Datacubes

The use of services for data access, such as the OGC Web Coverage Service, have not been investigated as most platforms provide their data for processing as files staged into the application environment. Thus, it is still open whether application packages should make use of web services for data access or rely on the data provided by each platform.

Semantics, Correlations, and Constraints

Further explore strong semantics and the usage of standard compliant taxonomies for attribute names and values. Explore parameter correlations and constraints can be handled.

STAC

Produce an *OGC Best Practice* document that discusses data flow management using STAC local catalogues as the input and output manifests between the ADES and the hosted application and between ADES and EMS.

6.5. Docker

Explore Docker build pipeline

Explore modified Docker build pipelines to solve concerns with respect to Docker image access

control, security enforcement, and version control. Docker images could be built on the ADES side rather than by the application developer.

Docker Repositories

In most cases, Docker images have been stored in public repositories. The usage of private repositories shall be standardized.

6.6. General Guidelines and Performance

Tutorials

Tutorials should be developed as lots of development follows examples, in support of standards and specifications.

Parallelization

Further work is required to explore the parallelization and other application performance drivers. These include RAM optimization and GPU processing.

Appendix A: References

The following documents are referenced in this document.

- OGC Testbed initiatives that developed parts of the Earth Observation Applications Architecture
 - [OGC Testbed-13: EP Application Package Engineering Report \(17-023\)](#)
 - [OGC Testbed-13: Application Deployment and Execution Service Engineering Report \(17-024\)](#)
 - [OGC Testbed-13: Cloud Engineering Report \(17-035\)](#)
 - [OGC Testbed-14: Application Package Engineering Report \(18-049r1\)](#)
 - [OGC Testbed-14: ADES & EMS Results and Best Practices Engineering Report \(18-050r1\)](#)
 - [OGC Testbed-15: Catalogue and Discovery Engineering Report \(19-020r1\)](#)
- Relevant OGC Standards, Extensions, and Profiles
 - [OGC 06-121r9, OGC® Web Services Common Standard](#)
 - [OGC 14-065r2, OGC Web Processing Service 2.0.2 Interface Standard Corrigendum 2, 2018](#)
 - [OGC 13-026r8, OGC OpenSearch Extension for Earth Observation 1.0, 2016](#)
 - [OGC 10-032r8, OGC OpenSearch Geo and Time Extensions 1.0.0, 2014](#)
 - [OGC 10-157r4, OGC Earth Observation Metadata profile of Observations & Measurements 1.1, 2014](#)
 - [OGC 14-055r2, OGC OWS Context GeoJSON Encoding Standard, 1.0, 2017](#)
 - [OGC 17-069r3, OGC API - Features - Part 1: Core, 1.0, 2019](#)
- External specifications
 - [Commonwl.org: Common Workflow Language Specifications, v1.0.2](#)
 - [Commonwl.org: Common Workflow Language Specifications, v1.1](#)
 - [STAC: SpatioTemporal Asset Catalogs v1.0.0](#)

Appendix B: Terms and definitions

B.1. Technical terms

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9](#) shall apply. In addition, the following terms and definitions apply.

- **Container**

A standardized unit of software ([Docker](#)).

- **OpenAPI Document**

A document (or set of documents) that defines or describes an API. An OpenAPI definition uses and conforms to the OpenAPI Specification [[OpenAPI](#)]

- **OpenSearch**

Draft specification for web search syndication, originating from Amazon's A9 project and given a corresponding interface binding by the OASIS Search Web Services working group.

- **Service interface**

Shared boundary between an automated system or human being and another automated system or human being

- **Workflow**

Automation of a process, in whole or part, during which electronic documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (source ISO 12651-2:2014)

B.2. Abbreviated terms

- ADES Application Deployment and Execution Service
- AOI Area Of Interest
- AP Application Package
- BPEL Business Process Execution Language
- CFP Call For Participation
- CWL Common Workflow Language
- DWG Domain Working Group
- EMS Execution Management Service
- EO Earth Observation
- EP Exploitation Platform
- ER Engineering Report
- ESA European Space Agency

- GUI Graphical User Interface
- JSON JavaScript Object Notation
- MEP Mission Exploitation Platform
- OWC OWS Context
- REST REpresentational State Transfer
- 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

Appendix C: Revision History

Table 2. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
August 10, 2020	I. Simonis	1.0	all	initial release
September 14, 2020	I. Simonis	1.1	all	editorial changes