OGC Testbed-14

Application Package Engineering Report

Table of Contents

1. Summary . . . 4
    1.1. Requirements & Research Motivation . . . 4
    1.2. Prior-After Comparison . . . 5
    1.3. Recommendations for Future Work . . . 5
    1.4. Document contributor contact points . . . 6
    1.5. Foreword . . . 6
2. References . . . 7
3. Terms and definitions . . . 8
    3.1. Abbreviated terms . . . 8
4. Overview . . . 10
5. Recap Of Work Done In Testbed-13 . . . 11
    5.1. OWS Context . . . 14
6. Discussion of Sponsor Requirements . . . 15
7. Application Chaining . . . 16
    7.1. Common Workflow Language (CWL) . . . 16
8. Solution . . . 20
    8.1. The AP In The Architecture . . . 20
    8.2. The AP Facets . . . 20
        8.2.1. Facet 1: for Alice, the EO scientist . . . 21
        8.2.2. Facet 2: for the M2M interactions . . . 21
    8.3. Example . . . 21
    8.4. Limitations and More Information . . . 25
9. Other Considered Possibilities . . . 26
10. Applications . . . 28
    10.1. Applications related to CFP . . . 28
        10.1.1. Stacker . . . 28
        10.1.2. Feature Extractor . . . 28
        10.1.3. Flood Detector . . . 29
    10.2. Other applications . . . 29
        10.2.1. Applications based on SNAP . . . 29
        10.2.2. Application packaging of ML systems . . . 29
Appendix A: Application graph . . . 30
Appendix B: Revision History . . . 31
Appendix C: Bibliography . . . 32

Publication Date: 2019-02-07

Approval Date: 2018-12-13

Submission Date: 2018-11-21

Reference number of this document: OGC 18-049r1

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D008

Category: Public Engineering Report

Editor: Paulo Sacramento

Title: OGC Testbed-14: Application Package Engineering Report

---

OGC Engineering Report

COPYRIGHT

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 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.

Chapter 1. Summary

This Engineering Report (ER) describes the work performed by the Participants in the Exploitation Platforms Earth Observation Clouds (EOC) Thread of OGC Testbed-14 in regard to the Application Package (AP).

The AP serves as a means to convey different kinds of information describing a certain application - often, but not necessarily, an Earth Observation data processing algorithm - so that different elements of an ecosystem generically known as an Exploitation Platform can exchange information among themselves in a standard and interoperable way. The AP guarantees that, despite potentially very heterogeneous implementations and implementing entities, applications are treated equally. The AP also guarantees that the Earth Observation scientist who developed it on the one hand is shielded from infrastructure details and heterogeneity and on the other hand benefits from the ability to execute the same application on different infrastructure.

Given its suitability for conveying a Common Operating Picture (COP), in OGC Testbed-13 the OGC Web Services (OWS) Context standard had been chosen as the basic encoding for the Application Package. Despite serious consideration, and while acknowledging the advantages of that approach, the consensus among Participants was not to continue along this path in Testbed-14 but instead to opt for an AP encoding, consisting of a WPS-T (Transactional Web Processing Service (WPS)) DeployProcess message encoded in JSON (see Chapter 9 for the rationale). The information model conveyed in this manner does not differ significantly from the one that could be conveyed using OWS Context, and its main, common features can be briefly listed as:

- a link to the application execution unit,
- a description of the application's inputs and outputs,
- links to required Earth Observation data catalogues,
- and the possibility to pass other auxiliary information.

An important difference in Testbed-14 with respect to Testbed-13 is that the application execution unit is not limited to a Docker container, but can also be a workflow described in Common Workflow Language (CWL), something which stems directly from one of the Sponsor requirements. Finally, it is important to note that this route does not preclude from embedding an OWS Context structure in the enclosing DeployProcess document if this is desired.

Starting from the lessons learned and limitations identified in Testbed-13, and embracing the new and changed Sponsor requirements, this ER explains the trade-offs, decisions and conclusions taken by the Participants throughout the project.

1.1. Requirements & Research Motivation

The Application Package in the Exploitation Platforms context is required to:

- Reference the executable unit that implements the application functionality
- Describe its input/output interface
- List additional required or optional resources that impact the application execution, such as

catalogue endpoints, base map layers, etc.

The work done in Testbed-13 addressed these basic requirements and provided a good starting point for the work in Testbed-14. Indeed, lessons learned and limitations were identified that the work herewith described attempts to address.

1.2. Prior-After Comparison

The work in the EOC thread of both Testbed-13 and Testbed-14 establishes the existence of a Dockerized application as a pre-requisite for using the specified Application Package formats. This assumption is therefore unchanged after this Testbed. Particularly due to the new requirement in Testbed-14 of supporting application chaining, a workflow description language, CWL has been introduced. This was not needed in Testbed-13 and this part of the work might be of interest to the Workflow DWG more than for the WPS Standards Working Group (SWG).

Catalogue endpoint information that in Testbed-13 was conveyed within an OWS Context document encoded in Extensible Markup Language (XML) is conveyed in Testbed-14 within a WPS-T DeployProcess document encoded in JavaScript Object Notation (JSON). The OpenSearch for EO standard is used for catalogue searches but it was agreed during the Testbed that the TEP (Thematic Exploitation Platform) Client would not perform such searches on behalf of the user and it would be actually the EMS (Execution Management Service) deliverable doing this. Such considerations are therefore not considered of particular interest to the OpenSearch for EO SWG in the context of this ER. More information on this part of the work can be found on the "EMS & ADES Best Practices ER" of the Testbed-14 EOC thread ([1]).

Given that the adopted solution is totally based on WPS and its transactional extension (WPS-T), the work described in this ER, and more broadly the work done in the EOC thread of Testbed-14, is particularly relevant to the WPS SWG. More so because the actual Application Package consists of a WPS-T DeployProcess document. Whilst in Testbed-13 the XML encoding of WPS requests/responses was the basis for the Client-ADES interface, in Testbed-14 a JSON encoding of WPS is used and RESTful bindings of WPS-T constitute the basis of both the Client-EMS and EMS-ADES interfaces. The work done in the EOC thread of Testbed-14 has followed closely the work of the WPS SWG and indeed fed back to it, particularly for what concerns the WPS Representational State Transfer (REST) Application Programming Interface (API) - not the transactional extension.

1.3. Recommendations for Future Work

Given that the Application Package encoding is based on the WPS standard, the work described in this ER is considered relevant to the corresponding SWG activities. In particular, some Change Requests may be identified in Testbed-14 or, at least, the preparation of Best Practices on how to use a WPS-T DeployProcess message for an Exploitation Platform Application Package could be envisaged and suggested.

For OGC, the work described in this Engineering Report is deemed extremely relevant in satisfying the needs of the European Space Agency as one of its Strategic Partners. The ER makes the case for how OGC standards are helpful in such a densely populated ecosystem of very heterogeneous entities, technologies and implementations such as the Exploitation Platforms one.

Future work should be dedicated to consolidating the WPS REST API that has been proposed by the WPS SWG and contributed to by Testbed-14 EOC thread Participants, as well as initiating a standardization path for the transactional extension of WPS (WPS-T). Even though those two topics are more relevant in the context of the separate Testbed-14 ER on the EMS and ADES Best Practices (D009, [1]), given that in the adopted solution the Application Package is itself an element of the WPS protocols, such future work is also relevant for this ER. A specific issue of interest (but also risk) resides in how to transform a message-based protocol such as WPS (with its traditional *GetCapabilities*, *DescribeProcess*, *Execute* operations) into a resource-based protocol (WPS REST), where resources such as *processes* and *jobs* are managed using the basic HTTP operations (*GET*, *POST*, *PUT*, etc.). A straightforward conversion (i.e. simply mapping the XML messages into REST/JSON) may result in APIs which are not intuitive and are semantically unclear. It is therefore important to dedicate time and effort to properly designing a resource-based version of a previously existing XML-based protocol. Another relevant and general issue in this discussion is how to transform/encode XML into JSON content.

The relevance of CWL to the OGC should also be investigated. It could be particularly interesting for the Workflow DWG and could be considered as a potential future Community Standard.

1.4. Document contributor contact points

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

Contacts

Name	Organization
Paulo Sacramento	Solenix
Christophe Noel	Spacebel
Patrick Jacques	Spacebel
Peter Vretanos	CubeWerx
Tom Landry	CRIM
Jerome Gasperi	Geomatys
Guilhem Legal	Geomatys

1.5. 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 2. References

The following normative documents are referenced in this document:

- OGC: OGC 06-121r9, OGC® Web Services Common Standard, 2010 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 14-065r2, OGC Web Processing Service 2.0.2 Interface Standard Corrigendum 2, 2018 [https://portal.opengeospatial.org/files/14-065r2]
- CWL group: Common Workflow Language Specifications, v1.0.2 [https://www.commonwl.org/v1.0/]
- ISO: ISO 19510:2013, Information technology - Object Management Group Business Process Model and Notation, 2013 [http://www.bpmn.org/]

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.

3.1. Abbreviated terms

- ADES Application Deployment and Execution Service
- AP Application Package
- BPEL Business Process Execution Language
- BPMN Business Process Model and Notation
- CFP Call For Participation
- CWL Common Workflow Language
- DL Deep Learning
- DWG Domain Working Group
- EMS Execution Management Service
- EO Earth Observation
- EOC Earth Observation Clouds
- ER Engineering Report
- ESA European Space Agency
- GPT Graph Processing Tool
- GUI Graphical User Interface
- ICT Information and Communications Technology
- IT Information Technology
- JSON JavaScript Object Notation
- JSON-LD JSON for Linked Data
- M2M Machine-2-Machine (interface)
- MEP Mission Exploitation Platform
- ML Machine Learning
- OWC OWS Context
- OWS OGC Web Services
- REST REpresentational State Transfer
- SNAP Sentinel Application Platform
- SWG Standards Working Group
- TAR Tape ARchive

- TEP Thematic Exploitation Platform
- TIE Technical Interoperability Experiment
- VHR Very High Resolution
- WPD WPS Process Description
- WPS Web Processing Service
- WPS-T WPS Transactional
- YAML YAML Ain’t Markup Language

Chapter 4. Overview

Section 5 briefly recaps the work done in Testbed-13 for what concerns the Application Package, as a technical introduction to the topics dealt with in this ER.

Section 6 discusses the sponsor requirements to which the identified solutions respond to.

Section 7 deals specifically with the application chaining aspect of the EOC thread in Testbed-14, introducing the CWL given its prominent role in the work that was undertaken.

Section 8 describes the Application Package solution as implemented by Participants for the interactions between the TEP client and the several EMS and ADES instances. The two Application Package facets are introduced, highlighting how these were already suggested by the sponsor in the CFP and how the proposed solution took this on board.

Section 9 presents other possibilities considered particularly during the first part of the activity, noting their pros and cons. It can be seen as part of the rationale for arriving at the solution.

Section 10 describes the applications to be used in the EOC thread as found in the CFP. It also presents other applications delivered by the participants, to demonstrate that the proposed Application Package format is generic and can be used for other applications not involved in the EOC thread.

Chapter 5. Recap Of Work Done In Testbed-13

The final consensus in Testbed-13 concerning the Application Package was to use the ATOM encoding of the OGC OWS Context 1.0 standard, conveying the elements considered necessary within a single *entry* element, through the use of several *owc:offering* elements contained in that *entry*. Three elements in particular are considered fundamental:

- the link to the executable application in the form of a Docker container;
- a WPS Process Description containing the information of the application's inputs and outputs;
- one or more catalogue endpoints so that users can look for EO data of interest required as one of the application's inputs.

The following snippet shows an example of such an OWS Context file.

Application Package in OWS Context format

```
<?xml version="1.0" encoding="UTF-8"?>
<feed
   xmlns="http://www.w3.org/2005/Atom"
   xmlns:wps="http://www.opengis.net/wps/2.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:owc="http://www.opengis.net/owc/1.0" xml:lang="en">
   <title>EOC Land Cover Application Package</title>
   <subtitle type="text">Landcover EOC APP PKG</subtitle>
   <id>http://www.opengis.net/tb13/eoc/examples/app_pkg/landcover</id>
   <updated>2017-08-15T01:28:00Z</updated>
   <link rel="profile" href="http://www.opengis.net/spec/owc-atom/1.0/req/core"
         title="This file is compliant with version 1.0 of OGC Context"/>
   <link rel="profile" href="http://www.opengis.net/tb13/eoc"
         title="This file is compliant with Testbed-13 EOC Thread for Application
Packing"/>
(...)
   <entry>
      <title>EOC Land Cover Application</title>
      <id>http://www.opengis.net/tb13/eoc/LandCover</id>
      <updated>2017-09-04T15:23:09Z</updated>
      <content type="text/plain">Land Cover Application.</content>
      <!-- DOCKER IMAGE -->
      <owc:offering code="http://www.opengis.net/tb13/eoc/docker">
         <owc:content type="text/plain">
registry.hub.docker.com/cnlspacebel/landcover</owc:content>
      </owc:offering>
      <!-- THE WPS PROCESS DESCRIPTION -->
      <owc:offering code="http://www.opengis.net/tb13/eoc/wpsProcessOffering">
         <owc:content type="application/xml">
            <wps:ProcessOffering
               jobControlOptions="async-execute dismiss"
```

```
                outputTransmission="value reference"
                xmlns:wps="http://www.opengis.net/wps/2.0"
                xmlns:ows="http://www.opengis.net/ows/2.0"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.opengis.net/wps/2.0
                                    http://schemas.opengis.net/wps/2.0/wps.xsd">
                <wps:Process>
                   <ows:Title>Land Cover Mapping</ows:Title>
                   <ows:Abstract>Land Cover Mapping is based on the Sentinel-2
processing workflow generated for the F-TEP platform.</ows:Abstract>
                   <ows:Identifier>LandCoverMapping</ows:Identifier>
                   <wps:Input>
                      <ows:Title>Sentinel-2 Image</ows:Title>
                      <ows:Abstract>URL of Sentinel-2 Level 1C image product in the
format offered by AWS or IPT, with a size of up to multiple gigabytes.</ows:Abstract>
                      <ows:Identifier>Image</ows:Identifier>
                      <wps:ComplexData>
                         <wps:Format mimeType="text/directory" default="true"/>
                      </wps:ComplexData>
                   </wps:Input>
                   <wps:Input>
                      <ows:Title>Reference Data</ows:Title>
                      <ows:Abstract>Representative training data set with land cover
class attributes, in OGR vector format supported by GDAL, such as ESRI shapefile, in a
flat zip structure containing .shp and the supporting files.</ows:Abstract>
                      <ows:Identifier>ReferenceData</ows:Identifier>
                      <wps:ComplexData>
                         <wps:Format mimeType="application/zip" encoding="base64"
                            default="true"/>
                      </wps:ComplexData>
                   </wps:Input>
                   <wps:Input>
                      <ows:Title>Area Of Interest</ows:Title>
                      <ows:Identifier>AreaOfInterest</ows:Identifier>
                      <wps:LiteralData>
                         <wps:Format mimeType="text/plain" default="true"/>
                         <LiteralDataDomain>
                            <ows:AnyValue/>
                            <ows:DataType ows:reference=
"http://www.w3.org/2001/XMLSchema#string">String</ows:DataType>
                         </LiteralDataDomain>
                      </wps:LiteralData>
                   </wps:Input>
                   <wps:Input>
                      <ows:Title>EPSG Code</ows:Title>
                      <ows:Abstract>Coordinate reference system expressed as a code
from the EPSG database using the format "EPSG:NNNN".</ows:Abstract>
                      <ows:Identifier>EPSGCode</ows:Identifier>
                      <wps:LiteralData>
                         <wps:Format mimeType="text/plain" default="true"/>
                         <LiteralDataDomain>
```

```
                        <ows:ValuesReference ows:reference="http://...."/>
                        <ows:DataType
                           ows:reference="http://www.w3.org/2001/XMLSchema#string"
>String</ows:DataType>
                     </LiteralDataDomain>
                  </wps:LiteralData>
               </wps:Input>
               <wps:Input>
                  <ows:Title>Target Resolution</ows:Title>
                  <ows:Abstract>Target Resolution</ows:Abstract>
                  <ows:Identifier>TargetResolution</ows:Identifier>
                  <wps:LiteralData>
                     <wps:Format mimeType="text/plain" default="true"/>
                     <LiteralDataDomain>
                        <ows:AnyValue/>
                        <ows:DataType
                           ows:reference="http://www.w3.org/2001/XMLSchema#long"
>Long Integer</ows:DataType>
                     </LiteralDataDomain>
                  </wps:LiteralData>
               </wps:Input>
               <wps:Output>
                  <ows:Title>GeoTIF Image</ows:Title>
                  <ows:Abstract>Labeled GeoTIF file containing, for each pixel, one
of the class codes specified in the training reference data.</ows:Abstract>
                  <ows:Identifier>Image</ows:Identifier>
                  <wps:ComplexData>
                     <wps:Format mimeType="image/tiff" encoding="raw"
                        default="true"/>
                  </wps:ComplexData>
               </wps:Output>
            </wps:Process>
         </wps:ProcessOffering>
      </owc:content>
   </owc:offering>
   <!-- OPENSEARCH OFFERING FOR IPT POLAND -->
   <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/opensearch">
      <ows:content type="application/opensearchdescription+xml"
         href=
"http://geo.spacebel.be/opensearch/description.xml?parentIdentifier=EOP%3AIPT%3ASentin
el2"/>
   </owc:offering>
   <!-- OPENSEARCH OFFERING FOR AWS -->
   <owc:offering code="http://www.opengis.net/spec/owc-atom/1.0/opensearch">
      <ows:content type="application/opensearchdescription+xml"
         href=
"http://geo.spacebel.be/opensearch/description.xml?parentIdentifier=EOP%3SENTINEL-
HUB%3ASentinel2"/>
   </owc:offering>
</entry>
</feed>
```

Naturally, given how broad the OWS Context standard is, there is a large amount of other (optional) information that can be provided in addition to this basic one. An example of such additional information is given in the Testbed-13 Application Package ER ([2]), in the form of tailored base map layers that a sophisticated client could use in its mapping framework as a way to facilitate the user's usage of the application (see [3] for a related discussion).

Additionally, in the Testbed-13 Cloud ER [4], container-based application packages were used to demonstrate cloud interoperability and application portability for the Canadian Forestry Service sponsor. The execution unit, a container hosted in a Docker registry, was provided as a mandatory input to a WPS 1.0 service. At execution, the application package was pulled asynchronously from a registry by a worker process, listening to a message queue. The AP implementation described in the Cloud ER is based on the Sentinel Application Platform (SNAP). It receives an XML graph file as an input that is passed to the GPT process. This graph file is describing a SNAP EO workflow and is essentially the business logic of the AP.

5.1. OWS Context

There were a few reasons that justified the choice of OWS Context as a way to convey the Application Package.

One was that OWS Context is already an OGC standard and therefore a new standardization path would not be required before its wide adoption.

Another one was that, particularly in its ATOM encoding, it is a very browser-friendly format, enabling the best of two worlds, i.e. an OGC standard and a general-purpose, well-tolerated format which is accepted by the mainstream community.

Finally, the OWS Context standard was created to convey a "Common Operating Picture" to systems involved in a potentially complex, distributed flow. This makes it particularly adequate for the use case at hand, in which different systems from different vendors need to cooperate to implement such a flow.

Chapter 6. Discussion of Sponsor Requirements

In the Call for Participation (CFP), the European Space Agency (ESA), as one of the EOC thread's sponsors, defined the following main requirements for the activities related to the Application Package:

- that in the first few months, Participants seek and reach consensus on aspects of the AP left open in Testbed-13, issuing clear recommendations on the way forward during Testbed-14
- that the AP has two facets, to address two types of use: one for application developers who are not IT experts (e.g. scientists), with a simple encoding, for example based on YAML or JSON; one for Machine-To-Machine (M2M) application package exchange, containing *"all the information necessary to ensure that the application will behave in the same expected way in any platform supporting it"*
- that the AP supports 2 types of application: interactive (e.g. with a Graphical User Interface (GUI)) and non-interactive (e.g. batch processing)
- that the AP conveys any information required for security/privacy/visibility aspects

In addition, even if not strictly defined as a requirement for the AP, since users are expected not only to be able to register single applications but also applications consisting of the chaining of other applications, and clients involved in the flow are required to support such chaining, specific provisions are made at the level of the AP to deal with this. Indeed, as explained in this ER, this has had important implications in the finally adopted solution.

Where and when relevant throughout this ER, reference is made to these requirements, but a few general considerations, assumptions, as well as decisions taken by Participants are presented in this section.

As requested by the sponsor, Participants sought and reached consensus on aspects of the AP left open in Testbed-13.

In particular, the road initiated in Testbed-13, with an AP format based on OWS Context was not confirmed (see Chapter 9 for the rationale).

Concerning the third requirement in the list above, with agreement of all Participants and the sponsor, the support of interactive applications was not considered throughout Testbed-14 for several reasons. First of all, ESA clarified that such support was a desirable rather than fundamental requirement. Secondly, this would have introduced significant additional complexity to a thread that already needed to tackle a large number of issues. Finally, it was a common opinion amongst Participants that it is not straightforward to conciliate interactive applications with WPS-based process executions, both in their synchronous and asynchronous modes of operation (e.g. how can you show a GUI to a user to collect inputs during a WPS execution? What is the value of showing a GUI to the user at the end of a WPS execution, so he can explore the result?).

Chapter 7. Application Chaining

As briefly mentioned in the previous chapter, the AP needs to be able to support scenarios in which a user using a client wants to chain previously discovered applications and register the resulting (chained) application.

For this, Participants identified the need for a machine-readable, formal way of describing this chaining, which results in a workflow. Two main candidates were discussed during the Testbed-14 Kick-Off: BPMN (Business Process Model and Notation) and CWL (Common Workflow Language) [5]. Note that BPMN is also known as ISO 19510:2013.

Even though more than one Participant had proposed the usage of BPMN in answering the CFP, during the Kick-Off there was consensus that it was likely too complex for the needs of the thread and also that, in itself, BPMN is not executable and therefore extra steps would be required to transform BPMN into executable code. The sponsor, in turn, highlighted that its needs in terms of chaining were relatively simple and therefore complexity was undesirable. CWL was proposed as a potentially suitable alternative and following the first weeks of work and research into CWL, Participants agreed to rely on CWL for the Testbed-14 EOC needs. It should furthermore be noted that the finally adopted solution relies on CWL not only for the chaining and workflow description aspect, but also in a broader sense.

7.1. Common Workflow Language (CWL)

As explained at https://www.commonwl.org/index.html (accessed on 30/08/2018), “The Common Workflow Language (CWL) is a specification for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments. CWL is designed to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, Physics, and Chemistry. CWL is developed by a multi-vendor working group consisting of organizations and individuals aiming to enable scientists to share data analysis workflows. The CWL project is maintained on Github and follows the Open-Stand.org principles for collaborative open standards development. CWL builds on technologies such as JSON-LD for data modelling and Docker for portable runtime environments.”

There are a few aspects in the description above worth noting. First of all, CWL originates from and is specified by the scientific community, which is also the main user community of interest in the EOC thread. Even though it does not target Earth Observation, Remote Sensing or generically Space scientists, it targets several other scientific communities which are very diverse one from the other. Secondly, it provides mechanisms for the formal description of data analysis workflows in a way that is **portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high-performance computing (HPC) environments**, also **aiming to enable scientists to share data analysis workflows** - these are all highly desirable characteristics in the Exploitation Platforms context. Finally, CWL builds on JSON-LD and Docker, which are two technologies that are:

- already highly popular and pervasive in Exploitation Platforms,
- are increasingly relevant to the OGC and,

- particularly in the case of Docker, were already tightly linked to the solutions proposed in Testbed-13 and the Application Package format for the EOC thread.

The following snippet shows a very simple example of CWL, wrapping a command-line tool (the *echo* command in this case) that takes one file as input, reads a message from that file and writes that message to a console.

Simple CWL example

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []
```

The second snippet below shows a slightly more complex example of CWL, highlighting how it can be used to describe a simple workflow consisting of two steps, one running after the other, with the first step's output(s) being the second step's input(s). Each step of the workflow is itself described in CWL, in its own file (*tar-param.cwl* and *arguments.cwl*). In this specific case, the workflow consists of a first step in which a Java source code file is extracted from a TAR file (untarred) and a second step in which the Java code is compiled.

Example of workflow in CWL

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow
inputs:
  tarball: File
  name_of_file_to_extract: string

outputs:
  compiled_class:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: tarball
      extractfile: name_of_file_to_extract
    out: [extracted_file]

  compile:
    run: arguments.cwl
    in:
      src: untar/extracted_file
    out: [classfile]
```

The following aspects are particularly relevant:

- the *class* attribute is set to *Workflow*
- the *compiled_class* workflow output points (using *outputSource*) to the output of the second and final step (the step is called *compile* and the output is called *classfile*, marked within the step as *[classfile]*)
- the *tarball* and *name_of_file_to_extract* workflow inputs are used as inputs of the first step (*untar*), pointed to using their names. The workflow inputs could also be used as inputs of other steps of the workflow.
- the *src* attribute of the *compile* step's input points to the output of the first step (*untar*) called *extracted_file*, marked within that step as *[extracted_file]*. This is the mechanism that implements the chaining of the two steps.
- the workflow steps are not necessarily ran in the order in which they are listed. Instead, the order is determined by the dependencies between the steps and steps without dependencies on other steps can actually be run in parallel.

Participants did not dedicate substantial effort to investigating and collecting potential disadvantages and limitations of CWL and its associated tool-chain, but the following can be highlighted:

- CWL is well-suited for command-line applications but not for interactive, GUI applications nor for orchestrating Web Services (this is not necessarily relevant because - as done by EOC Testbed-14 Participants and explained in [1] - a Web Service endpoint, e.g. WPS, can be placed in front of the CWL executing logic)
- CWL comes with and requires (at least for straightforward and easy use) an associated tool-chain (*cwl-runner*, etc.), which is a dependency

Chapter 8. Solution

8.1. The AP In The Architecture

The figure below shows how the Application Package and its two facets fit in the overall architecture, with its three main building blocks: the TEP Client, the EMS and the ADES.

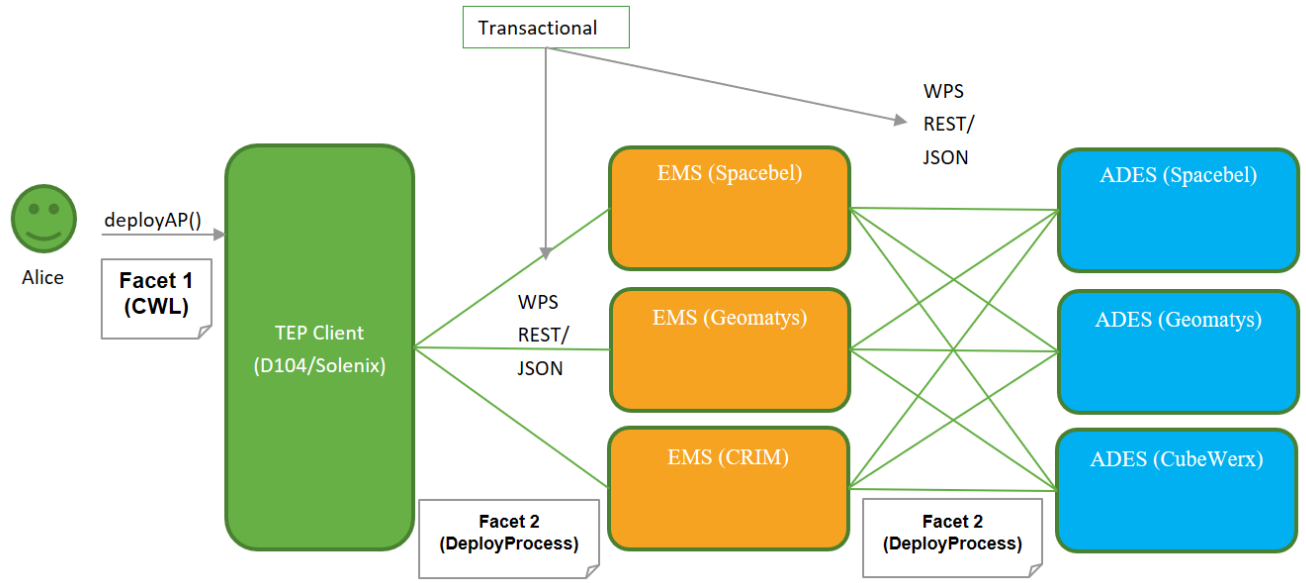

Figure 1. WPS-T DeployProcess message as Application Package in the Testbed-14 architecture

The finally adopted solution for Facet 2 thus consists of a WPS-T DeployProcess document, through which the application's inputs and outputs are described. The execution unit can be conveyed as a Docker container for simple applications or a CWL file for workflows. Any other additional required information is passed using *ows:metadata* fields.

With respect to Testbed-13, it continues to be a pre-requisite that the application has been previously Dockerized and is registered in a Docker Hub. CWL offers a simple way of pointing to a Docker container (by adding a *hints* or *requirements* section and a *DockerRequirement* attribute to the CWL file). It is not strictly required for a developer *Alice* to already have a CWL file prepared as the TEP Client offers a CWL editor which can be used to upload an existing CWL file or to create one from scratch. Facet 1 of the Application Package consists of this. The TEP Client will also assist in generating Facet 2 of the Application Package by building a WPS-T DeployProcess document that includes Facet 1, i.e. the CWL file.

In line with the architectural constraints provided by the sponsor, the WPS-T DeployProcess document is encoded in JSON, not XML, and the interfaces all throughout are based on a RESTful protocol fully described in another Testbed-14 EOC ER, gathering the ADES & EMS Best Practices ([1]).

8.2. The AP Facets

As mentioned in Chapter 6, the Sponsor requires an AP with two facets, one based on a simple encoding, targeting the application developer, *Alice*, who is an Earth Observation scientist that is an expert in remote sensing and writes Earth Observation data processing algorithms, often in a

programming language convenient for such use (Matlab, Fortran, Python, bash script); and one for the Machine-To-Machine (M2M) exchanges of the AP, therefore needing to contain all the information required by the several architectural building blocks for executing the application.

It should be noted that, as is the case in Testbed-14 with the TEP Client deliverable, clients can exist to greatly simplify and automate the work of putting together an AP and, specifically, the two facets described below. In the particular case of Facet 1, a CWL file, the application developer *Alice* may have prepared it already (i.e. Facet 1 is pre-existing), or she can use the client and its CWL editor to generate one (i.e. Facet 1 is a client output).

8.2.1. Facet 1: for Alice, the EO scientist

Given the adopted solution, of embedding (or pointing to) a CWL file in a WPS-T DeployProcess message, Facet 1 corresponds to the CWL file, that can exist outside the context of an AP in a Thematic Exploitation Platform, to describe an existing command-line application. Separate tools can be used to build, edit and run CWL files.

8.2.2. Facet 2: for the M2M interactions

Facet 2 corresponds to the WPS-T DeployProcess document itself, which thus includes Facet 1 as well. This document completely describes the application and is used for the M2M interactions, allowing downstream elements of the flow (e.g. EMS, ADES) to understand and configure the application for execution and possible chaining with other applications.

8.3. Example

The following snippet provides an example of Facet 2 of the Application Package, a WPS-T DeployProcess document encoded in JSON. This specific example is for a workflow.

Application Package (WPS-T DeployProcess) encoded in JSON (workflow)

```
{
    "processDescription": {
        "process": {
            "id": "multiSensorNDVIStacker_2collections",
            "title": "multiSensorNDVIStacker_2collections",
            "abstract": "",
            "keywords": [],
            "inputs": [
                {
                    "id": "image-collection1",
                    "title": "Input Image",
                    "formats": [
                        {
                            "mimeType": "application/zip",
                            "default": true
                        }
                    ],
                    "minOccurs": 1,
```

```
                    "maxOccurs": "unbounded",
                    "additionalParameters": [
                        {
                            "role":
"http://www.opengis.net/eoc/applicationContext/inputMetadata",
                            "parameters": [
                                {
                                    "name": "EOImage",
                                    "values": [
                                        "true"
                                    ]
                                }
                            ]
                        }
                    ]
                },
                {
                    "id": "image-collection2",
                    "title": "Input Image",
                    "formats": [
                        {
                            "mimeType": "application/zip",
                            "default": true
                        }
                    ],
                    "minOccurs": 1,
                    "maxOccurs": "unbounded",
                    "additionalParameters": [
                        {
                            "role":
"http://www.opengis.net/eoc/applicationContext/inputMetadata",
                            "parameters": [
                                {
                                    "name": "EOImage",
                                    "values": [
                                        "true"
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ],
            "outputs": [
                {
                    "id": "output",
                    "title": "Stacked Image",
                    "formats": [
                        {
                            "mimeType": "image/tiff",
                            "default": true
```

```
                        }
                    ]
                }
            ]
        },
        "processVersion": "1.0.0",
        "jobControlOptions": [
            "async-execute"
        ],
        "outputTransmission": [
            "reference"
        ]
    },
    "executionUnit": [
        {
            "href": "https://raw.githubusercontent.com/spacebel/testbed14/master/cwl-
examples/multiSensorNDVIStacker_2collections/multiSensorNDVIStacker_2collections-
v4.cwl"
        }
    ],
    "deploymentProfileName": "http://www.opengis.net/profiles/eoc/workflow"
}
```

As explained above, Facet 1 of the AP - a CWL file - is contained in Facet 2, the WPS-T DeployProcess message. In the example, this can be seen at the end, whereby the *deploymentProfileName* attribute is set to *http://www.opengis.net/profiles/eoc/workflow* -, and the *executionUnit* points (using *href*) to the CWL workflow file (*https://raw.githubusercontent.com/spacebel/testbed14/master/cwl-examples/multiSensorNDVIStacker_2collections/multiSensorNDVIStacker_2collections-v4.cwl*).

As explained in [1], the adopted architecture relies on a WPS-T interface using JSON encoding. WPS-T is a transactional extension of WPS, as discussed in [6]. The information model proposed in WPS-T foresees that applications can be written in any one of a number of programming or description languages (e.g. Java, BPEL, etc.) and the so-called "execution unit" allows associating the WPS process to the set of instructions concretely implementing the application. In the Testbed-14 EOC thread, execution units based on CWL are considered. The *deploymentProfileName* and *executionUnit* attributes used in the example and solution are features of WPS-T.

Apart from the *deploymentProfileName* and *executionUnit* attributes, the third element visible in the example is a WPS Process Description encoded in JSON, which describes the application's inputs and outputs, still a major feature of any AP format.

The second snippet below provides another example of Facet 2 of the Application Package, in this case for a single application (not a workflow).

Application Package (WPS-T DeployProcess) encoded in JSON (application)

```
{
    "processDescription": {
        "process": {
            "id": "GeomatysNDVIStacker",
```

```
            "title": "NDVIStacker",
            "owsContext": {
                "offering": {
                    "code": "http://www.opengis.net/eoc/applicationContext/cwl",
                    "content": {
                        "href":
"https://raw.githubusercontent.com/Geomatys/Testbed14/master/application-
packages/NDVIStacker/NDVIStacker.cwl"
                    }
                }
            },
            "abstract": "",
            "inputs": [
                {
                    "id": "files",
                    "title": "Input NDVI Image",
                    "formats": [
                        {
                            "mimeType": "image/tiff",
                            "default": true
                        }
                    ],
                    "minOccurs": "1",
                    "maxOccurs": "unbounded",
                    "additionalParameters": [
                        {
                            "role":
"http://www.opengis.net/eoc/applicationContext/inputMetadata",
                            "parameters": [
                                {
                                    "name": "EOImage",
                                    "values": [
                                        "true"
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ],
            "outputs": [
                {
                    "id": "output",
                    "title": "NDVI Image",
                    "formats": [
                        {
                            "mimeType": "image/tiff",
                            "default": true
                        }
                    ]
                }
```

```
            ]
        },
        "processVersion": "1.0.0",
        "jobControlOptions": [
            "async-execute"
        ],
        "outputTransmission": [
            "reference"
        ]
    },
    "immediateDeployment": true,
    "executionUnit": [{
            "href": "images.geomatys.com/ndvis:latest"
        }],
    "deploymentProfileName":
"http://www.opengis.net/profiles/eoc/dockerizedApplication"
}
```

Whilst in the first snippet *executionUnit* points to a CWL file describing a workflow (*https://raw.githubusercontent.com/spacebel/testbed14/master/cwl-examples/multiSensorNDVIStacker_2collections/multiSensorNDVIStacker_2collections-v4.cwl*) and *deploymentProfileName* is set to *http://www.opengis.net/profiles/eoc/workflow*, in the second *executionUnit* points to a Docker container (*images.geomatys.com/ndvis:latest*) and *deploymentProfileName* is set to *http://www.opengis.net/profiles/eoc/dockerizedApplication*.

This illustrates how the same AP format consisting of a WPS-T DeployProcess message encoded in JSON can be used to request the deployment both of simple applications (second snippet) and workflows (first snippet).

It is finally important to note that, even in the second case, the *owsContext* element is used to point to a CWL file associated to the process/application. This is because CWL has been used in the EOC Thread of Testbed-14 not only to describe workflows but also to address one of the most important issues left open in Testbed-13 (see [7]), related to how inputs/outputs are made available by the ADES component to the Docker containers for proper execution. See [1] for more information.

8.4. Limitations and More Information

For all details pertaining to relevant security aspects - authentication, authorization - as well as quotation and billing, the reader is referred to [8].

It is accepted as a limitation that interactive applications are not supported.

Chapter 9. Other Considered Possibilities

Very early in Testbed-14 there was consensus about using CWL at least for Facet 1 of the AP and other options were not considered. CWL consists simply of a JSON (or YAML) document.

For what concerns Facet 2 instead, several different possibilities were considered.

On the top part of the figure below, the two different AP possibilities that had been experimented with in Testbed-13 are schematized. On the bottom left, a natural evolution of the AP based on OWS Context is depicted.

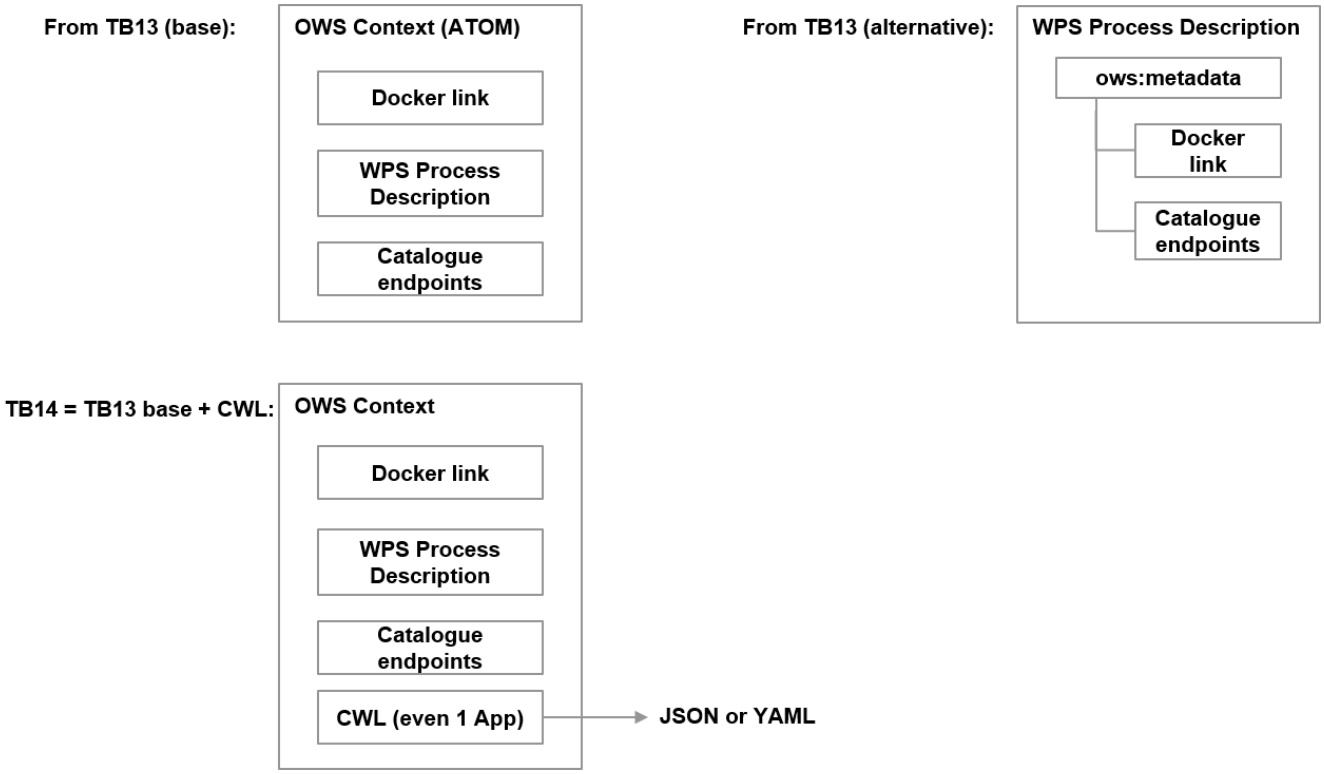

Figure 2. Application Packages in Testbed-13 and Testbed-14 AP based on OWS Context (possibility)

It should be highlighted that all three alternatives convey the same information model, consisting of three main elements: a link to the Docker container with the application code; the input/output description; and information on catalogue endpoints required for users to discover data of interest. In all three cases, the input/output description is done using a WPS Process Description (WPD) document. In the first case, this document is embedded in the OWS Context document, in the second case it is the document itself. In all three cases there are several ways of providing additional information. In the case of OWS Context, additional *owc:entry* and *owc:offering* elements can be used, as well as all general mechanisms allowed by the OWS base schemas. The latter can be used in the second case - in particular *ows:metadata* elements - to provide any application-specific information.

The option in the bottom left of the picture is simply an evolution of the one on the top left, with the addition of a (embedded) CWL file (in JSON or YAML) that could be used in the simplest case to describe a single application and, in more complex cases, to describe a chaining of more than one application.

The following figure provides some more detail on this option. Logically, it would convey

application description information (link to a Docker container and a CWL file encoded in JSON) and auxiliary information aimed at helping an application user identify proper inputs for the execution (adequate catalogues, base maps highlighting features relevant to the specific application - e.g. volcanoes in an application that requires selecting a volcano, etc.).

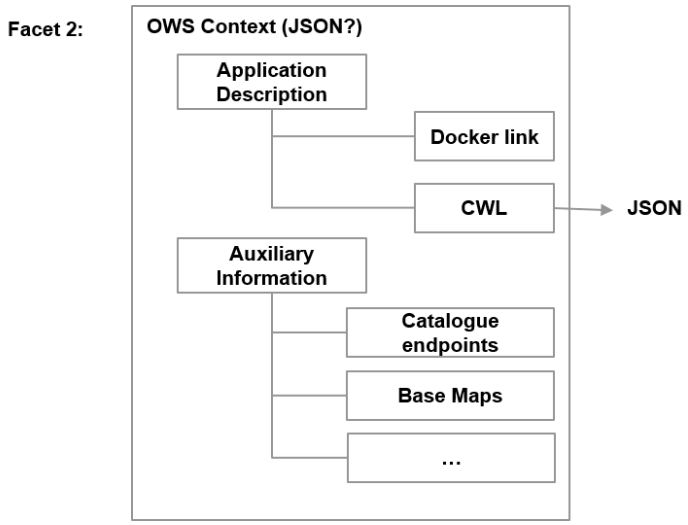

Figure 3. AP based on OWS Context (possibility)

As explained in the beginning of this ER, after several discussions among the participants it was finally concluded that an AP built around the WPS protocol and WPD was better suited to the objectives of the Testbed and the Sponsor requirements. The participants believe that the agreed solution addresses the issue of partial redundancy between the link to the Docker container and the CWL file that could be created with the option above, and also welcomed and took on-board the feedback received during the OGC EOEP Hackathon ([3]) that the WPD was generally considered more intuitive to developers than the OWS Context as a way to convey the required information.

Specifically, developer feedback during the Hackathon was that the information relevant for ADES developers to register (and subsequently execute) applications consisted of the WPD embedded within the OWS Context and that everything else was substantially superfluous (i.e. the value of the OWS Context wrapper around it was not understood). Even when some of the uses for the information conveyed through the OWS Context (e.g. convenient base maps) were explained, there was consensus among the Hackathon Participants that there were other ways to provide that information. For example, it is possible to include a complete OWS Context document, if so desired, within the WPD document, by using the *ows:metadata* mechanism. In general, the Testbed-14 EOC thread participants shared this opinion as well.

Chapter 10. Applications

Please refer to [1] for fully comprehensive examples of APs corresponding to applications used in the actual Technical Interoperability Experiment (TIEs).

10.1. Applications related to CFP

From the three applications below, a workflow can be created to chain the Stacker, then the Feature Extractor and, finally, the Flood Detector.

10.1.1. Stacker

The Stacker automatically select the overlapping region between all input files, selects appropriate bands and outputs a GeoTiff, BEAM_DIMAP or NetCDF-CF stack. The bands are automatically selected, but could be parametrized. The application supports both optical and Synthetic Aperture Radar (SAR) data (Sentinel 1/2, Radarsat) and can be adapted to support other missions too.

The Stacker takes as input :

- Input images filename (Must be made available inside the Docker container storage space. Unlike the Geomatys application, this application needs the files, not a bounding box, time range and catalogue
- Filename for output
- Output format (GeoTiff, BEAM_DIMAP or NetCDF-CF)

The processing consists of the following steps:

- Every image is stacked into the same file
- Using the first image, a reference band is chosen
- Every other band is updated to fit the same resolution, projection and coordinate system
- The bands intersection is determined and bands are cropped accordingly

10.1.2. Feature Extractor

The Feature Extractor takes, as input, a stack and computes features for each pixel. At the moment, the standard deviation is used and is judged sufficient for demonstration purposes. The output is a GeoTiff, BEAM_DIMAP or NetCDF-CF stack, same extents and projections as input, and containing normalized grayscale values.

The Feature Extractor takes as input :

- Filename of a stack generated by the previous application
- Filename for output
- Output format (GeoTiff, BEAM_DIMAP or NetCDF-CF)

10.1.3. Flood Detector

The Flood Detector takes, is input, a feature stack and produces a stack of binary images of water-covered areas. This is the step where Machine Learning models would be of most use to generally select features and classify elements.

The Flood Detector takes as input :

- Filename of a feature stack generated by the previous application
- Filename for output

10.2. Other applications

10.2.1. Applications based on SNAP

One participant provided on a registry a Docker image name *snap6-base* containing SNAP and custom operators, with their accompanying CWL files. This image prepares the environment to allow the running user to have permissions to write files via GPT. This is required since the custom operators are not found by GPT when running as root, and CWL specifies a specific user dynamically on every machine to run the Docker image.

A script placed at */bin/gpt-graph-runner* is prepared to allow any user to run a graph file located at */opt/graph.xml*, and this script is automatically called as entry point to the image. Specific Docker images such as a stacker (*snap6-stack-creation*) and a feature extractor (*snap6-sfs*) override the above /opt/graph.xml file with their corresponding implementations. An example of a graph can be found in Appendix. Call to the images are accomplished by :

```
docker run {images-volumes-mappings} --user={uid:gid [any-but-root]} {snap-base}
{specific-graph.xml-parameters}
```

10.2.2. Application packaging of ML systems

A Machine Learning system described in [9] was containerized in order to be packaged according to findings in the present ER. This semantic segmentation application allowed runs on a Deep Learning (DL) model trained on Very High Resolution (VHR) imagery. It is expected that this DL application can be used in CWL-based EO workflows.

Appendix A: Application graph

StackCreation.xml graph file, with read and stack operation as SNAP operators

```
<graph id="Graph">
<version>1.0</version>
<node id="ProductSet-Reader">
  <operator>ProductSet-Reader</operator>
  <sources/>
  <parameters class="com.bc.ceres.binding.dom.XppDomElement">
    <fileList>${files}</fileList>
  </parameters>
</node>
<node id="StackCreation">
  <operator>StackCreation</operator>
  <sources>
    <sourceProduct.3 refid="ProductSet-Reader"/>
  </sources>
  <parameters class="com.bc.ceres.binding.dom.XppDomElement"/>
</node>
</graph>
```

Appendix B: Revision History

Table 1. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
November 22, 2018	P. Sacramento	1.0	all	issue for submission as pending to OGC TC Charlotte
October 09, 2018	P. Sacramento	.9	all	first version for review

Appendix C: Bibliography

1. Sacramento, P., others: OGC Testbed-14: ADES & EMS Results and Best Practices Engineering Report. OGC 18-050,Open Geospatial Consortium, http://www.opengis.net/doc/PER/t14-D009 (2018).
2. Goncalves, P., others: OGC Testbed-13: EP Application Package Engineering Report. OGC 17-023,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-023.html (2017).
3. Simonis, I., others: OGC Earth Observation Exploitation Platform Hackathon 2018 Engineering Report. Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2018).
4. Chen, C., others: OGC Testbed-13: Cloud ER. OGC 17-035,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-035.html (2017).
5. Amstutz, P., Crusoe, M.R., (editors), N.T., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Ménager, H., Nedeljkovich, M., Scales, M., Soiland-Reyes, S., Stojanovic, L.: (2016).
6. Cauchy, A.: OpenGIS® WPS2.0 Transactional Extension Implementation Standard. OGC 13-071r2,Open Geospatial Consortium, http://www.opengeospatial.org/docs/discussion-papers (2013).
7. Goncalves, P., others: OGC Testbed-13: Application Deployment and Execution Service ER. OGC 17-024,Open Geospatial Consortium, http://www.opengeospatial.org/per/17-024.html (2017).
8. Gasperi, J., others: OGC Testbed-14: Authorisation Authentication and Billing Engineering Report. OGC 18-057,Open Geospatial Consortium, http://www.opengis.net/doc/PER/t14-D010 (2018).
9. Landry, T., others: OGC Testbed-14: Machine Learning Engineering Report. OGC 18-038,Open Geospatial Consortium, http://www.opengis.net/doc/PER/t14-D030 (2018).