ADES Design Document

EOEPCA.SDD.xxx

TVUK System Team

Version 2, 06/07/2020:

ADES Design Document

EO Exploitation Platform Common Architecture

ADES Design Document

EOEPCA.SDD.xxx

COMMENTS and ISSUES	**PDF**
If you would like to raise comments or issues on this document, please do so by raising an Issue at the following URL https://github.com/EOEPCA/proc-ades/issues.	This document is available in PDF format here.
EUROPEAN SPACE AGENCY CONTRACT REPORT The work described in this report was done under ESA contract. Responsibility for the contents resides in the author or organisation that prepared it.	**TELESPAZIO VEGA UK Ltd** 350 Capability Green, Luton, Bedfordshire, LU1 3LU, United Kingdom. Tel: +44 (0)1582 399000 www.telespazio-vega.com

AMENDMENT HISTORY

This document shall be amended by releasing a new edition of the document in its entirety. The Amendment Record Sheet below records the history and issue status of this document.

Table 1. Amendment Record Sheet

ISSUE	DATE	REASON
0.1	14/04/2020	Initial in-progress draft
0.1.1	09/06/2020	Update Design with ADES release 0.1
0.1.2	06/07/2020	Updated Design for ADES release 0.1. Added Functional Design and External Interfaces sections
0.2.0	14/09/2020	Updated Design for ADES release 0.2 Updated Functional Design section
0.2.1	07/10/2020	Updated Design for ADES release 0.2 Updated Functional Design and Components Design section

Chapter 1. Introduction

1.1. Purpose and Scope

This document presents the ADES Design for the Common Architecture.

1.2. Structure of the Document

Section 2 - Overview

Provides an overview of the ADES component, within the context of the wider Common Architecture design.

Section 3 - [mainDesign]

Provides the design of the ADES component.

1.3. Reference Documents

The following is a list of Reference Documents with a direct bearing on the content of this document.

Reference	Document Details	Version
[EOEPCA-MSD]	EOEPCA - Master System Design Document EOEPCA.SDD.001 https://eoepca.github.io/master-system-design/published/v1.0/	Issue 1.0, 02/08/2019
[EOEPCA-UC]	EOEPCA - Use Case Analysis EOEPCA.TN.005 https://eoepca.github.io/use-case-analysis	Issue 1.0, 02/08/2019
[EP-FM]	Exploitation Platform - Functional Model, ESA-EOPSDP-TN-17-050	Issue 1.0, 30/11/2017
[TEP-OA]	Thematic Exploitation Platform Open Architecture, EMSS-EOPS-TN-17-002	Issue 1, 12/12/2017
[OGC-COMMON]	OGC Web Service Common, 06-121r9, https://www.ogc.org/standards/common	06-121r9, 04/07/2010
[OGC-WPS]	OGC Web Processing Service, OGC 05-007, http://portal.opengeospatial.org/files/?artifact_id=24151	05-007r7, 08/06/2007

Reference	Document Details	Version
[OGC-WPS2]	OGC® WPS 2.0 Interface Standard, OGC 14-065, http://docs.opengeospatial.org/is/14-065/14-065.html	14-065r2, 05/03/2015
[OGC-API-PROC]	OGC WPS 2.0 REST/JSON Binding Extension, Draft, OGC 18-062, https://raw.githubusercontent.com/opengeospatial/wps-rest-binding/develop/docs/18-062.pdf	1.0-draft
[OGC-OWC]	OGC OWS Context Conceptual Model, OGC 12-080, http://www.owscontext.org/ https://portal.opengeospatial.org/files/?artifact_id=55182	12-080r2, 22/01/2014
[OGC-OWC-ATOM]	OGC OWS Context Atom Encoding Standard, OGC 12-084, http://www.owscontext.org/ https://portal.opengeospatial.org/files/?artifact_id=55183	12-084r2, 14/01/2014
[CWL]	Common Workflow Language Specifications, https://www.commonwl.org/v1.0/	v1.0.2
[TB13-AP]	OGC Testbed-13, EP Application Package Engineering Report, OGC 17-023, http://docs.opengeospatial.org/per/17-023.html	17-023, 30/01/2018
[TB13-ADES]	OGC Testbed-13, Application Deployment and Execution Service Engineering Report, OGC 17-024, http://docs.opengeospatial.org/per/17-024.html	17-024, 11/01/2018
[TB14-AP]	OGC Testbed-14, Application Package Engineering Report, OGC 18-049r1, http://docs.opengeospatial.org/per/18-049r1.html	18-049r1, 07/02/2019
[TB14-ADES]	OGC Testbed-14, ADES & EMS Results and Best Practices Engineering Report, OGC 18-050r1, http://docs.opengeospatial.org/per/18-050r1.html	18-050r1, 08/02/2019
[TB14-WPS-T]	OGC Testbed-14: WPS-T Engineering Report, OGC 18-036r1, http://docs.opengeospatial.org/per/18-036r1.html	18-036r1, 07/02/2019

Reference	Document Details	Version
[OGC-API-HACK2019]	OGC API Hackathon 2019 Engineering Report, OGC 19-062, https://docs.ogc.org/per/19-062.html	19-062, 14/11/2019
[OS-GEO-TIME]	OpenSearch GEO: OpenSearch Geo and Time Extensions, OGC 10-032r8, http://www.opengeospatial.org/standards/opensearchgeo	10-032r8, 14/04/2014
[OS-EO]	OpenSearch EO: OGC OpenSearch Extension for Earth Observation, OGC 13-026r9, http://docs.opengeospatial.org/is/13-026r8/13-026r8.html	13-026r9, 16/12/2016
[GEOJSON-LD]	OGC EO Dataset Metadata GeoJSON(-LD) Encoding Standard, OGC 17-003r1/17-084	17-003r1/17-084
[GEOJSON-LD-RESP]	OGC OpenSearch-EO GeoJSON(-LD) Response Encoding Standard, OGC 17-047	17-047
[PCI-DSS]	The Payment Card Industry Data Security Standard, https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss	v3.2.1
[CEOS-OS-BP]	CEOS OpenSearch Best Practise, http://ceos.org/ourwork/workinggroups/wgiss/access/opensearch/	v1.2, 13/06/2017
[OIDC]	OpenID Connect Core 1.0, https://openid.net/specs/openid-connect-core-1_0.html	v1.0, 08/11/2014
[OGC-CSW]	OGC Catalogue Services 3.0 Specification - HTTP Protocol Binding (Catalogue Services for the Web), OGC 12-176r7, http://docs.opengeospatial.org/is/12-176r7/12-176r7.html	v3.0, 10/06/2016
[OGC-WMS]	OGC Web Map Server Implementation Specification, OGC 06-042, http://portal.opengeospatial.org/files/?artifact_id=14416	v1.3.0, 05/03/2006
[OGC-WMTS]	OGC Web Map Tile Service Implementation Standard, OGC 07-057r7, http://portal.opengeospatial.org/files/?artifact_id=35326	v1.0.0, 06/04/2010

Reference	Document Details	Version
[OGC-WFS]	OGC Web Feature Service 2.0 Interface Standard – With Corrigendum, OGC 09-025r2, http://docs.opengeospatial.org/is/09-025r2/09-025r2.html	v2.0.2, 10/07/2014
[OGC-WCS]	OGC Web Coverage Service (WCS) 2.1 Interface Standard - Core, OGC 17-089r1, http://docs.opengeospatial.org/is/17-089r1/17-089r1.html	v2.1, 16/08/2018
[OGC-WCPS]	Web Coverage Processing Service (WCPS) Language Interface Standard, OGC 08-068r2, http://portal.opengeospatial.org/files/?artifact_id=32319	v1.0.0, 25/03/2009
[AWS-S3]	Amazon Simple Storage Service REST API, https://docs.aws.amazon.com/AmazonS3/latest/API	API Version 2006-03-01
[OPENAPI]	OpenAPI Specification, https://swagger.io/specification/	API Version 3.0.3 2020-02-20

1.4. Terminology

The following terms are used in the Master System Design.

Term	Meaning
Admin	User with administrative capability on the EP
Algorithm	A self-contained set of operations to be performed, typically to achieve a desired data manipulation. The algorithm must be implemented (codified) for deployment and execution on the platform.
Analysis Result	The *Products* produced as output of an *Interactive Application* analysis session.
Analytics	A set of activities aimed to discover, interpret and communicate meaningful patters within the data. Analytics considered here are performed manually (or in a semi-automatic way) on-line with the aid of *Interactive Applications*.
Application Artefact	The 'software' component that provides the execution unit of the *Application Package*.
Application Deployment and Execution Service (ADES)	Web Processing Service that incorporates the Docker execution engine, and is responsible for the execution of the processing service (as a WPS request) within the ‘target’ Exploitation Platform.

Term	Meaning
Application Descriptor	A file that provides the metadata part of the *Application Package*. Provides all the metadata required to accommodate the processor within the WPS service and make it available for execution.
Application Package	A platform independent and self-contained representation of a software item, providing executable, metadata and dependencies such that it can be deployed to and executed within an Exploitation Platform. Comprises the *Application Descriptor* and the *Application Artefact*.
Bulk Processing	Execution of a *Processing Service* on large amounts of data specified by AOI and TOI.
Code	The codification of an algorithm performed with a given programming language - compiled to Software or directly executed (interpretted) within the platform.
Compute Platform	The Platform on which execution occurs (this may differ from the Host or Home platform where federated processing is happening)
Consumer	User accessing existing services/products within the EP. Consumers may be scientific/research or commercial, and may or may not be experts of the domain
Data Access Library	An abstraction of the interface to the data layer of the resource tier. The library provides bindings for common languages (including python, Javascript) and presents a common object model to the code.
Development	The act of building new products/services/applications to be exposed within the platform and made available for users to conduct exploitation activities. Development may be performed inside or outside of the platform. If performed outside, an integration activity will be required to accommodate the developed service so that it is exposed within the platform.
Discovery	User finds products/services of interest to them based upon search criteria.
Execution	The act to start a *Processing Service* or an *Interactive Application*.
Execution Management Service (EMS)	The EMS is responsible for the orchestration of workflows, including the possibility of steps running on other (remote) platforms, and the on-demand deployment of processors to local/remote ADES as required.
Expert	User developing and integrating added-value to the EP (Scientific Researcher or Service Developer)
Exploitation Tier	The Exploitation Tier represents the end-users who exploit the services of the platform to perform analysis, or using high-level applications built-in on top of the platform's services
External Application	An application or script that is developed and executed outside of the Exploitation Platform, but is able to use the data/services of the EP via a programmatic interface (API).

Term	Meaning
Guest	An unregistered User or an unauthenticated Consumer with limited access to the EP's services
Home Platform	The Platform on which a User is based or from which an action was initiated by a User
Host Platform	The Platform through which a Resource has been published
Identity Provider (IdP)	The source for validating user identity in a federated identity system, (user authentication as a service).
Interactive Application	A stand-alone application provided within the exploitation platform for on-line hosted processing. Provides an interactive interface through which the user is able to conduct their analysis of the data, producing *Analysis Results* as output. Interactive Applications include at least the following types: console application, web application (rich browser interface), remote desktop to a hosted VM.
Interactive Console Application	A simple *Interactive Application* for analysis in which a console interface to a platform-hosted terminal is provided to the user. The console interface can be provided through the user's browser session or through a remote SSH connection.
Interactive Remote Desktop	An Interactive Application for analysis provided as a remote desktop session to an OS-session (or directly to a 'native' application) on the exploitation platform. The user will have access to a number of applications within the hosted OS. The remote desktop session is provided through the user's web browser.
Interactive Web Application	An Interactive Application for analysis provided as a rich user interface through the user's web browser.
Key-Value Pair	A key-value pair (KVP) is an abstract data type that includes a group of key identifiers and a set of associated values. Key-value pairs are frequently used in lookup tables, hash tables and configuration files.
Kubernetes (K8s)	Container orchestration system for automating application deployment, scaling and management.
Login Service	An encapsulation of Authenticated Login provision within the Exploitation Platform context. The Login Service is an OpenID Connect Provider that is used purely for authentication. It acts as a Relying Party in flows with external IdPs to obtain access to the user's identity.
EO Network of Resources	The coordinated collection of European EO resources (platforms, data sources, etc.).
Object Store	A computer data storage architecture that manages data as objects. Each object typically includes the data itself, a variable amount of metadata, and a globally unique identifier.
On-demand Processing Service	A *Processing Service* whose execution is initiated directly by the user on an ad-hoc basis.

Term	Meaning
Platform (EP)	An on-line collection of products, services and tools for exploitation of EO data
Platform Tier	The Platform Tier represents the Exploitation Platform and the services it offers to end-users
Processing	A set of pre-defined activities that interact to achieve a result. For the exploitation platform, comprises on-line processing to derive data products from input data, conducted by a hosted processing service execution.
Processing Result	The *Products* produced as output of a *Processing Service* execution.
Processing Service	A non-interactive data processing that has a well-defined set of input data types, input parameterisation, producing *Processing Results* with a well-defined output data type.
Products	EO data (commercial and non-commercial) and Value-added products and made available through the EP. *It is assumed that the Hosting Environment for the EP makes available an existing supply of EO Data*
Resource	A entity, such as a Product, Processing Service or Interactive Application, which is of interest to a user, is indexed in a catalogue and can be returned as a single meaningful search result
Resource Tier	The Resource Tier represents the hosting infrastructure and provides the EO data, storage and compute upon which the exploitation platform is deployed
Reusable Research Object	An encapsulation of some research/analysis that describes all aspects required to reproduce the analysis, including data used, processing performed etc.
Scientific Researcher	Expert user with the objective to perform scientific research. Having minimal IT knowledge with no desire to acquire it, they want the effort for the translation of their algorithm into a service/product to be minimised by the platform.
Service Developer	Expert user with the objective to provide a performing, stable and reliable service/product. Having deeper IT knowledge or a willingness to acquire it, they require deeper access to the platform IT functionalities for optimisation of their algorithm.
Software	The compilation of code into a binary program to be executed within the platform on-line computing environment.
Systematic Processing Service	A *Processing Service* whose execution is initiated automatically (on behalf of a user), either according to a schedule (routine) or triggered by an event (e.g. arrival of new data).
Terms & Conditions (T&Cs)	The obligations that the user agrees to abide by in regard of usage of products/services of the platform. T&Cs are set by the provider of each product/service.

Term	Meaning
Transactional Web Processing Service (WPS-T)	Transactional extension to WPS that allows adhoc deployment / undeployment of user-provided processors.
User	An individual using the EP, of any type (Admin/Consumer/Expert/Guest)
Value-added products	Products generated from processing services of the EP (or external processing) and made available through the EP. This includes products uploaded to the EP by users and published for collaborative consumption
Visualisation	To obtain a visual representation of any data/products held within the platform - presented to the user within their web browser session.
Web Coverage Service (WCS)	OGC standard that provides an open specification for sharing raster datasets on the web.
Web Coverage Processing Service (WCPS)	OGC standard that defines a protocol-independent language for the extraction, processing, and analysis of multi-dimentional coverages representing sensor, image, or statistics data.
Web Feature Service (WFS)	OGC standard that makes geographic feature data (vector geospatial datasets) available on the web.
Web Map Service (WMS)	OGC standard that provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases.
Web Map Tile Service (WMTS)	OGC standard that provides a simple HTTP interface for requesting map tiles of spatially referenced data using the images with predefined content, extent, and resolution.
Web Processing Services (WPS)	OGC standard that defines how a client can request the execution of a process, and how the output from the process is handled.
Workspace	A user-scoped 'container' in the EP, in which each user maintains their own links to resources (products and services) that have been collected by a user during their usage of the EP. The workspace acts as the hub for a user’s exploitation activities within the EP

1.5. Glossary

The following acronyms and abbreviations have been used in this report.

Term	Definition
AAI	Authentication & Authorization Infrastructure
ABAC	Attribute Based Access Control
ADES	Application Deployment and Execution Service
ALFA	Abbreviated Language For Authorization
AOI	Area of Interest

Term	Definition
API	Application Programming Interface
CMS	Content Management System
CWL	Common Workflow Language
DAL	Data Access Library
EMS	Execution Management Service
EO	Earth Observation
EP	Exploitation Platform
FUSE	Filesystem in Userspace
GeoXACML	Geo-specific extension to the XACML Policy Language
IAM	Identity and Access Management
IdP	Identity Provider
JSON	JavaScript Object Notation
K8s	Kubernetes
KVP	Key-value Pair
M2M	Machine-to-machine
OGC	Open Geospatial Consortium
PDE	Processor Development Environment
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RBAC	Role Based Access Control
REST	Representational State Transfer
SSH	Secure Shell
TOI	Time of Interest
UMA	User-Managed Access
VNC	Virtual Network Computing
WCS	Web Coverage Service
WCPS	Web Coverage Processing Service
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tile Service
WPS	Web Processing Service
WPS-T	Transactional Web Processing Service

Term	Definition
XACML	eXtensible Access Control Markup Language

Chapter 2. Overview

2.1. Building Block Overview

Content Description

This section contains:

- High-Level Description of the Building Block
- Context within EOEPCA

The ADES provides a WPS 1.0 & 2.0 service interfaces as well as an OGC API Processes compliant API. It is responsible for the execution of the processing service (as a WPS request) within the 'target' Exploitation Platform (i.e. one that is close to the data). In the global scenario, we assume that the EMS ensures that the processor is deployed as a WPS service before it is invoked.

As illustrated in the EOEPCA Master System Design Document (ref. [EOEPCA-MSD]), the ADES provides an execution engine that handles the complexities of constructing the jobs and interfacing with the processing cluster. See the context wthin the EOEOPA, as depicted below in Figure 1:

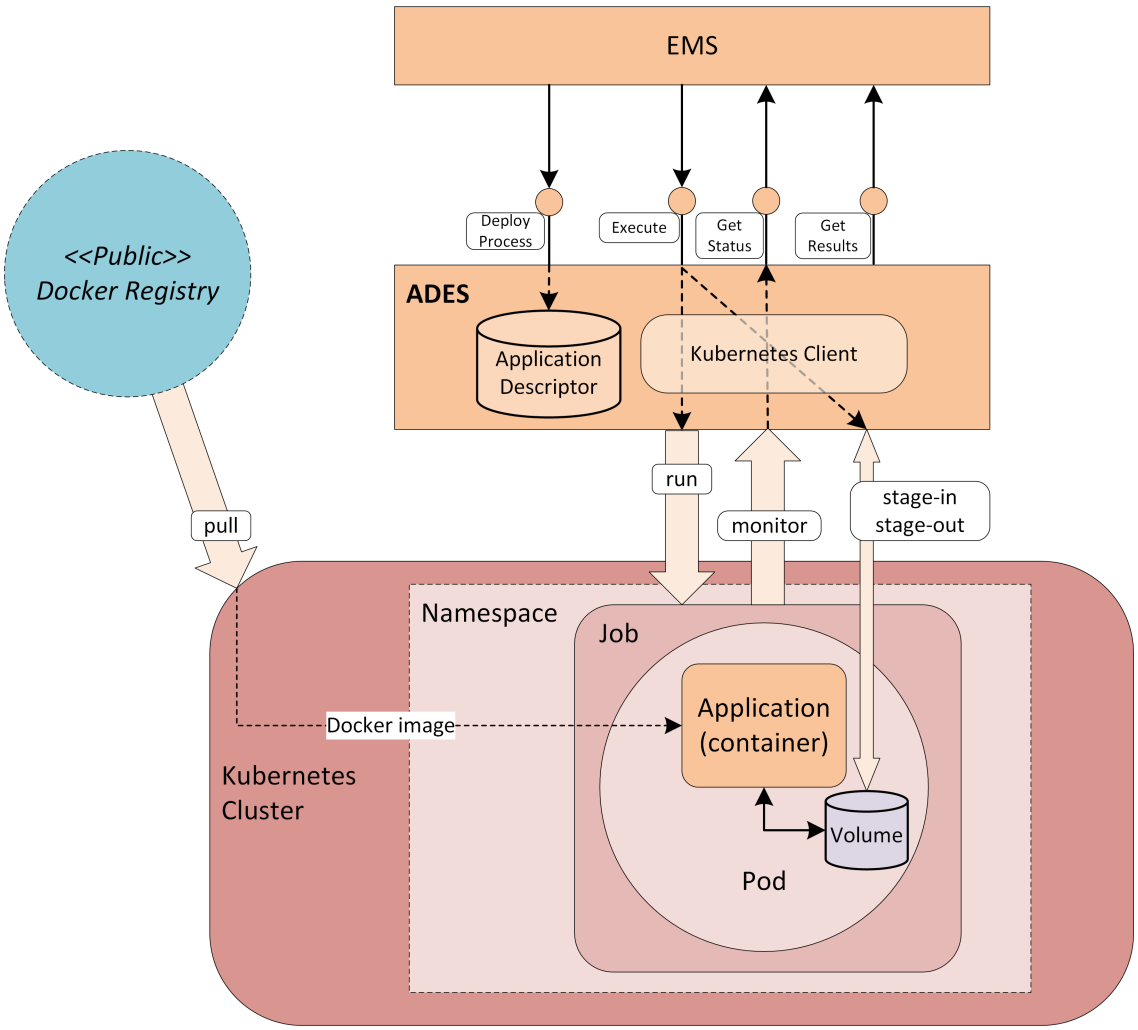


Figure 1. ADES Process Execution

The main responsibilities of the ADES are:

- Validate and accept an execution request from the EMS
- Submit the process execution to the processing cluster
- Monitor the process execution
- Retrieve the processing results

In order to accomplish the execution and monitor steps above, it also need to be responsible for the operations of:

- Data Stage-In for the process inputs
- Data Stage-Out for the process outputs

Those operations are internal sub-steps of the Processing Service execution operation.

2.1.1. Execution

The ADES invokes a Processing Service when a processing job is submitted to the WPS Server.

A processing job execution can be summarized by the execution of the following major steps on the Processing CLuster:

- Data Stage-In
- Processing
- Data Stage-Out

The Workflow Executor is responsible for the internal orchestration of the three steps above.
Additionally, when instructing the processing cluster to execute the various steps (stage-in/out or processing), the ADES provides also the reference of the Docker image to use.

2.1.1.1. Data Stage-In phase

Data Stage-In is the process to locally retrieve the inputs for the processing. Processing inputs are provided as EO Catalogue references and the ADES is responsible to translate those references into inputs available for the local processing.

2.1.1.2. Application Processing phase

Processing is the core step of the Execute operation. During this step input data is transformed into outputs data.

ADES supports it by executing the Application Workflow in Common Workflow Language ([CWL-spec]) provided by the user when the Application Package was <<deployed.

2.1.1.3. Data Stage-Out phase

Data Stage-Out is the process to store persistently the outputs of the processing and publish their results catalogs to the resource managers.

ADES retrieves the processing outputs and automatically stores them onto an external persistent

storage. Additionally, ADES publishes the metadata of the outputs onto a Catalogue, exposing the OpenSearch interface, and provides their references as an output.

2.2. Static Architecture

Content Description

This section contains:

- Diagram and description of the major logical components within the Building Block

The ADES is based on an Web API framework architecture. It offers primarly an OWS Server with WPS 1.0 & 2.0 OGC services and it exposes a REST/JSON interface compliant with the OGC Processes API [OGC-API-PROC] compliant with the OpenAPI specification [OPENAPI].

Figure 2 shows an overview of the ADES building block. It provides the framework for the WPS Server to manage and execute the processing services that interact with the external Kubernetes cluster via the Calrissian tool.

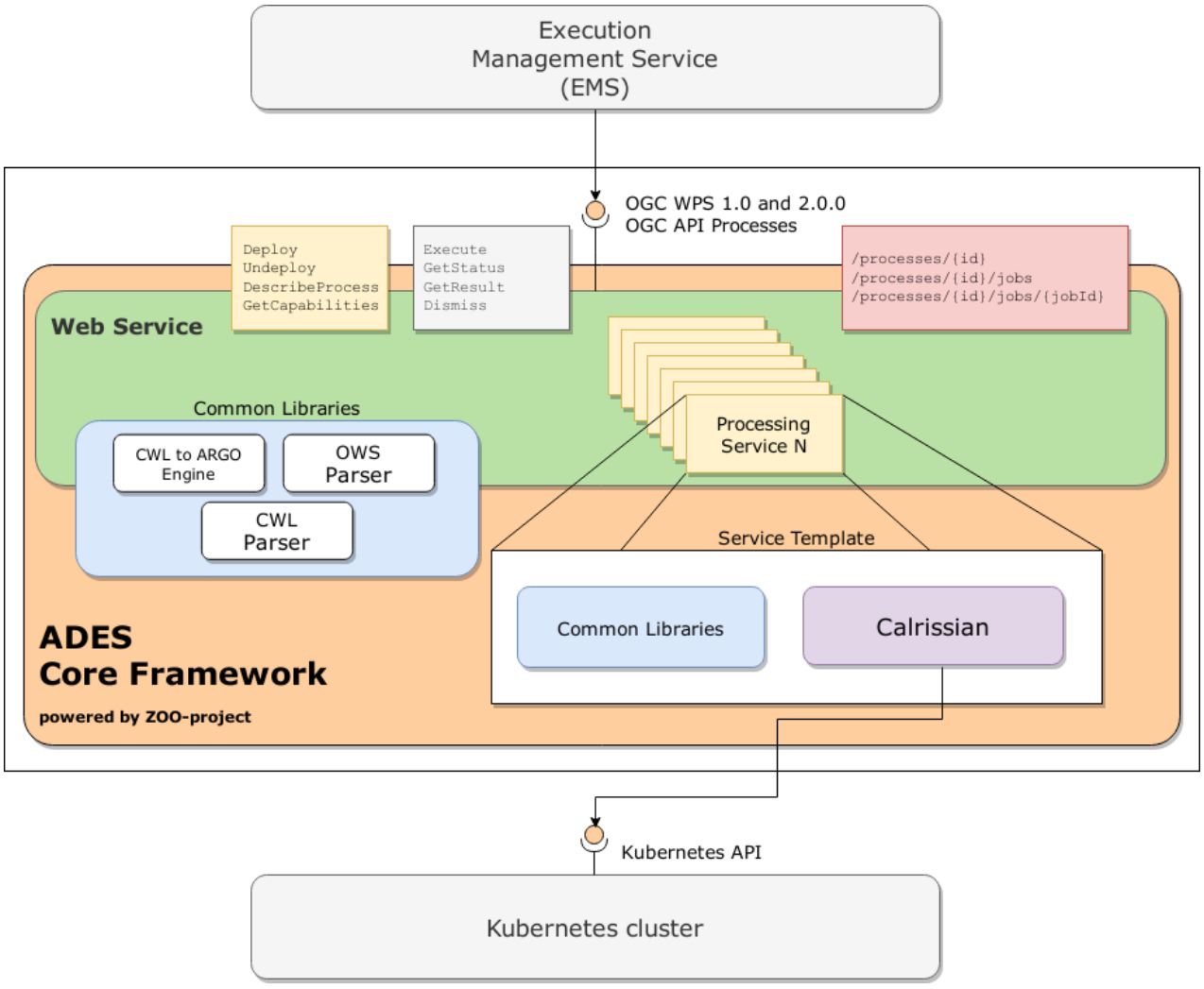


Figure 2. ADES building block overview

The WPS Service provides the external interface of the ADES, available to the EMS only. It enables internal code and configuration for creating the Processing Services via the Deploy/Undeploy WPS

operations.
Then, the WPS service triggers various function oif the Processing Services for the Execute, GetStatus, GetResult and Dismiss WPS operations.
Internally, every Processing Service deployed on the ADES, uses the common library to perform properly it's execution as per the Calrissian to submit, monitor, retrieve the results and dismiss a processing execution.

Section Components Design contains a detailed description of the ADES components.

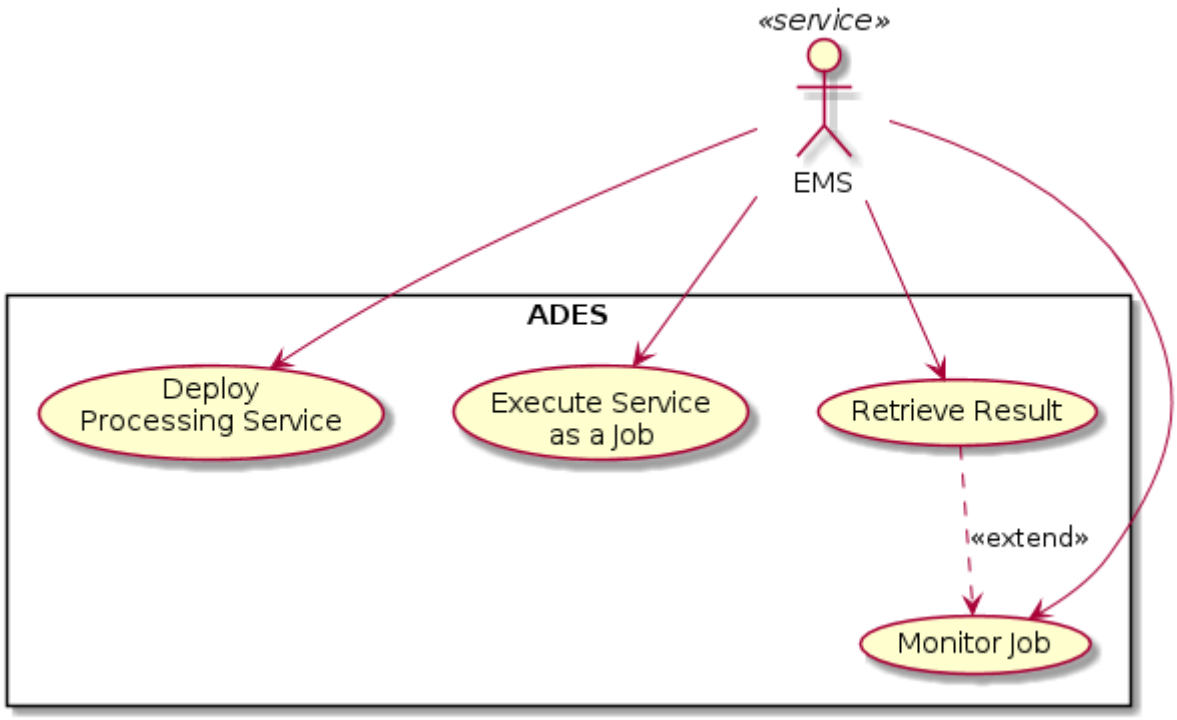
2.3. Use Cases

Content Description

This section contains:

- Diagrams and definition of the use cases covered by this Building Block

The following diagram describes the main use cases of the ADES



2.3.1. Deploy Processing Service

As an EMS service, I want to deploy (and un-deploy) a processing service, using a reference posting a OWS Context from a Resource Catalog referencing a [Common Workflow Language (CWL)] so that I can make it available (or remove) for the Execution, on behalf of my User.

2.3.2. Execute Service as Job

As an EMS service, I want to Execute an available processing service via the WPS & API interface, so that I can submit an execution, on behalf of my User.

2.3.3. Monitor Job

As an EMS service, I want to get the status of a given processing job, via the WPS & API interface, so that I can monitor a current or past execution, and provide the information back to my User.

2.3.4. Retrieve Result

As an EMS service, I want to get the reference of the processing results of a given processing job, via the WPS & API interface, so that I can provide the information back to my User.

2.4. Required Resources

Content Description

This section contains:

- List of data model and related resources used in the building block
- List of HW and SW required resources for the correct functioning of the building Block
- References to open repositories (when applicable)

2.4.1. Data

2.4.1.1. OWS Context

The OGC Web Services Context Document (OWS Context) was created to allow a set of configured information resources (service set) to be passed between applications primarily as a collection of services.

In the ADES, an OWS Context is used as the catalogue entry for referencing the Application Package

2.4.1.2. Common Workflow Language ([CWL-spec])

Common Workflow Language (Common Workflow Language ([CWL-spec])) is used as the application package descriptor. It covers the following elements necessary to describe the application:
* cyclic graph orchestrating the steps in order mapping workflow input/output with steps input/output
* Steps describing a command line with their input/output
* CWL specification extensions that may be used to provide the additional information elements

The application package is thus composed of a CWL file with the role of the application descriptor. The container reference is included in the CWL as a requirement.

Chapter 3. Components Design

Content Description

This section contains:

- A concise breakdown of the Building Block in several independent services (when applicable). For each component, the following subsections are added:
  - Overview and purpose: indicating the functionality covered by the component
  - SW Reuse and Dependencies: indicating reuse of third party open source solutions (if any) and any pre-required Dependencies
  - Interfaces: both internal to the building block and those exposed externally
  - Data: Data usage of the building block, data flow and any GDPR concerns should be addressed here
  - Applicable Resources: links and references to (Reference Docs), and repositories.

When a breakdown is necessary, a genral overview of the building block can be given. On the contrary, no breakdown indicates a single component development with the same expected sections.

3.1. ADES Core Framework

3.1.1. Overview and Purpose

The ADES Core Framework is the main software container for all the components composing the ADES.

3.1.2. Software Reuse and Dependencies

All software and their dependencies are described in the next sections

3.1.3. Interfaces

All interfaces are external and described in the External Interfaces section

3.1.4. Data

Not applicable

3.1.5. Applicable Resources

- [TB13-ADES]
- [TB14-WPS-T]

3.2. WPS Server

3.2.1. Overview and Purpose

The WPS Server provides the processing interfaces of the ADES. Basically, it exposes 2 web services:

- OGC Web Services Common [OGC-COMMON] hosting WPS 1.0.0 & 2.0.0 processing services
- OGC API Processes [OGC-API-PROC] hosting WPS in RESTful core OpenAPI

3.2.1.1. WPS 1.0 & 2.0 compliance

"ZOO-Kernel implements and complies with the WPS 1.0.0 and the WPS 2.0.0 standards edited by the Open Geospatial Consortium".
The Dismiss operation, which is *"only available in WPS 2.0.0, it lets the client ask the server to stop a running service and remove any file it created"*, requires changes in the source code to support the EMS operations. The current ZOO-Kernel implementation relies on local processes which are being removed from the execution environment. This approach prevents the management of remote processes. In this context, the ZOO-Kernel third-party software is modified to allow managing remote execution via APIs, in particular when the remote process exposes an OGC WPS-T 2.0.0 interface. It is foreseen an upstream contribution to the open-source ZOO-Project project.

3.2.1.2. OGC API Processes compliance

The ZOO-Project product, available at www.zoo-project.org, is an open-source implementation of both the Web Processing Service (WPS) standard (version 1.0.0 and 2.0) and the OGC API - Processes specification. The ZOO-Kernel is the main component written in C which enables the deployment of geospatial processes on the Web in a way that conforms to OGC standards.footnote:[https://docs.ogc.org/per/19-062.html#_zoo_project [OGC-API-HACK2019]

3.2.2. Software Reuse and Dependencies

The following open-source software is reused:

- ZOO-Kernel http://zoo-project.org/docs/kernel/

3.2.3. Interfaces

The WPS Server exposes the OGC OWS services WPS 1.0 & WPS 2.0 as well as an OGC API Processes. According to their respective specifications, they allow to deploy, execute, monitor and dismiss processing services.
More details in the External Interfaces section.

3.2.4. Data

Not Applicable.

3.2.5. Applicable Resources

- [OGC-COMMON]
- [OGC-WPS]
- [OGC-WPS2]
- [OGC-API-PROC]
- [OGC-API-HACK2019]

3.3. Workflow Executor

3.3.1. Overview and Purpose

The workflow executor is the component in charge of the scheduling ang the execution of the processing jobs.
When a Processing Service is trigger by the WPS Server via a job submission, this latter invokes the workflow executor to perform the steps necessary for the application to execute properly

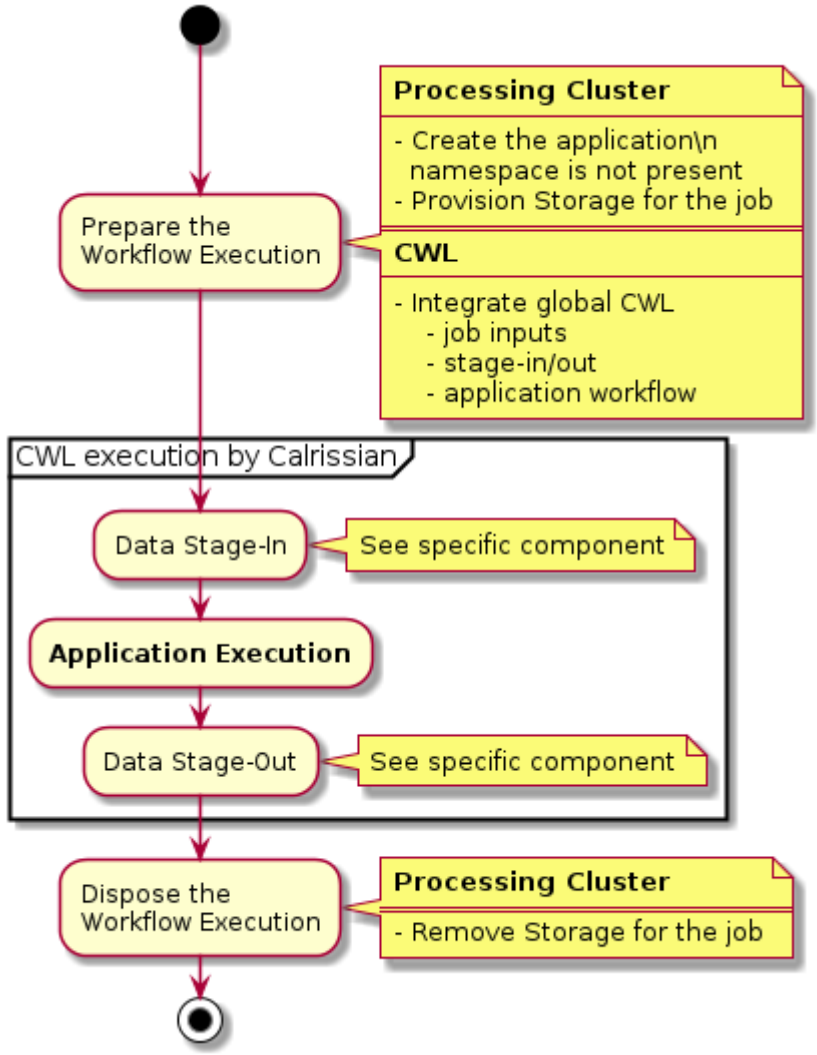


Figure 3. ADES Workflow Executor Activity Diagram

- Prepare the workflow Execution

    - On the **Processing Cluster**
        - Create the application namespace is not present
        - Provision Storage for the job
    - Create the main **CWL** by integrating
        - job inputs
        - stage-in/out
        - application workflow
- Execute the Data Stage-In
- Execute the Application Workflow
- Execute the Data Stage-Out
- Dispose the workflow execution cleaning the storage (e.g. delete the volume)

3.3.2. Software Reuse and Dependencies

The following open-source software is reused:

- FastAPI https://fastapi.tiangolo.com/
- Calrissian https://github.com/Duke-GCB/calrissian
- Data Stage-In
- Data Stage-Out

3.3.3. Data

- Job inputs
- Application Workflow
- Application Image

3.3.4. Applicable Resources

- Common Workflow Language ([CWL-spec])

3.4. Processing Service

3.4.1. Overview and Purpose

Processing Services are the representation of the Application Package deployed on the ADES.
Indeed, when a user deploys a service on the ADES, this latter creates a **Processing Service** from the Application Package.
Internally, a Processing service represents 3 items

- the Application Workflow in Common Workflow Language ([CWL-spec])
- the WPS describeProcess generated from the Application Workflow inputs, outputs and from

the OWS Context description (title, description)

- the Application Image in [Docker]

The following diagram shows the Processing Service's items as components in the ADES once deployed and the relationships with the other components

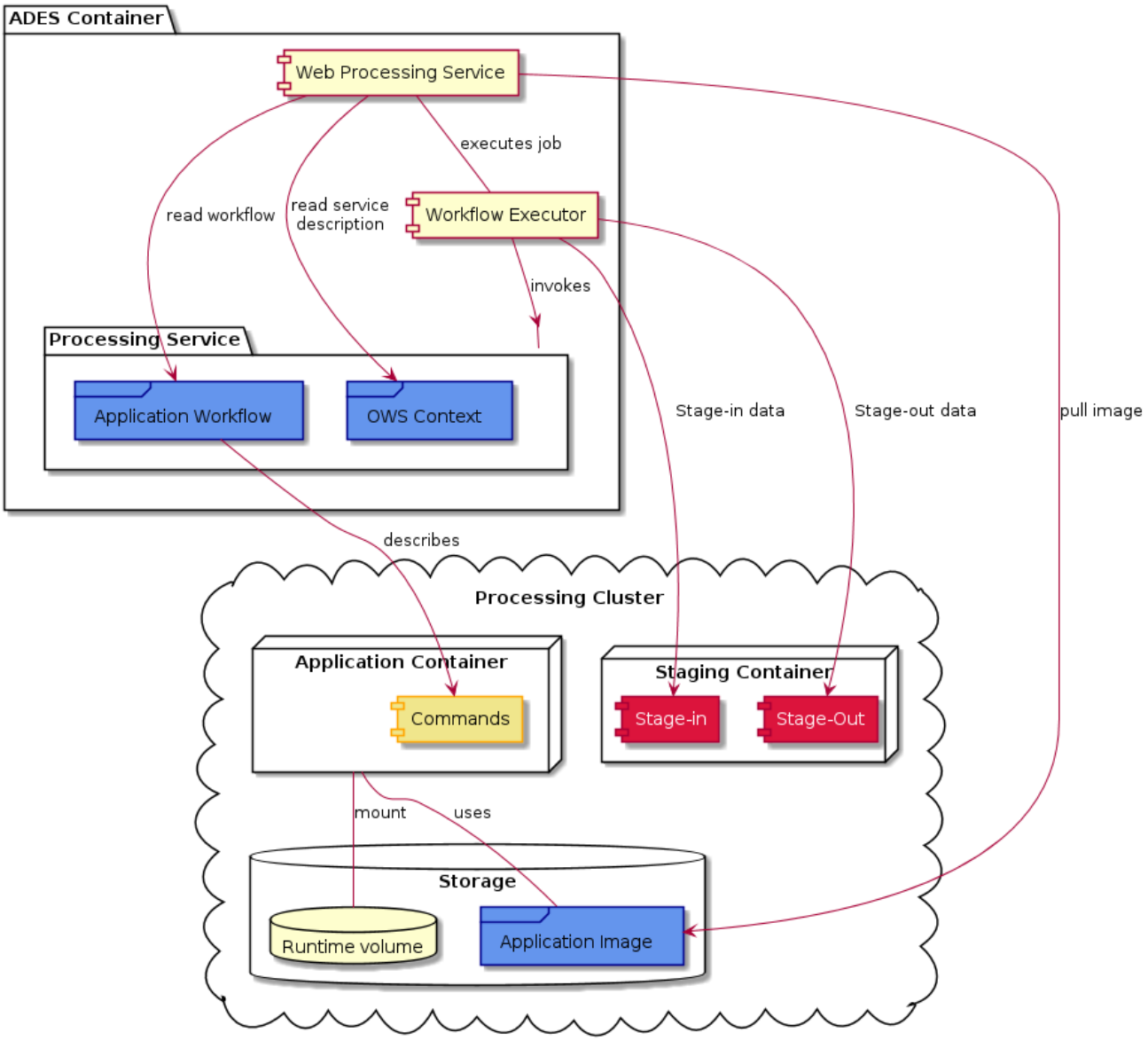


Figure 4. ADES Service Components Diagram

The Processing Service component will be invoked by the Workflow Executor when a processing job is submitted to the WPS Server.

3.4.2. Software Reuse and Dependencies

The following open-source software is reused:

- CWL Tool Common Workflow Language ([CWL-spec])
- Calrissian https://github.com/Duke-GCB/calrissian
- ZOO-Services http://zoo-project.org/docs/services/index.html

3.4.3. Data

- OWS Context
- CWL

3.4.4. Applicable Resources

- Common Workflow Language ([CWL-spec])
- [ZOO]

Chapter 4. Functional Design

Content Description

This section contains:

- A listing of the main functions with regards to the Use Cases. For each function,
  - Overview and purpose: description of the function
  - Activity diagram: graphical description of the function
  - Data Input/Ouput: Data flowing in and out of the function
  - Applicable Resources: links and references to (Reference Docs), and repositories.

When a breakdown is necessary, a general overview of the function can be given. On the contrary, no breakdown indicates a single component development with the same expected sections.

4.1. Application Deployment to Web Processing Service

4.1.1. Overview and Purpose

This is the main function in charge of building a Processing Service from an Application Package. Concretely, this function read an Application Item as an OWS Context referencing the Application Workflow. With all the information in those documents, it builds a Processing Service describing

- General information about the service: identifier, title, abstract
- Input parameters definition: identifiers, types, options, default values,...
- Ouput results options: identifiers, types, format

Regardless of the deployment mechanism, the Application Workflow is used to convey the information that describes the process inputs/outputs into WPS Server data model.

The mapping between CWL and WPS follows the work performed in the OGC Testbed 16.

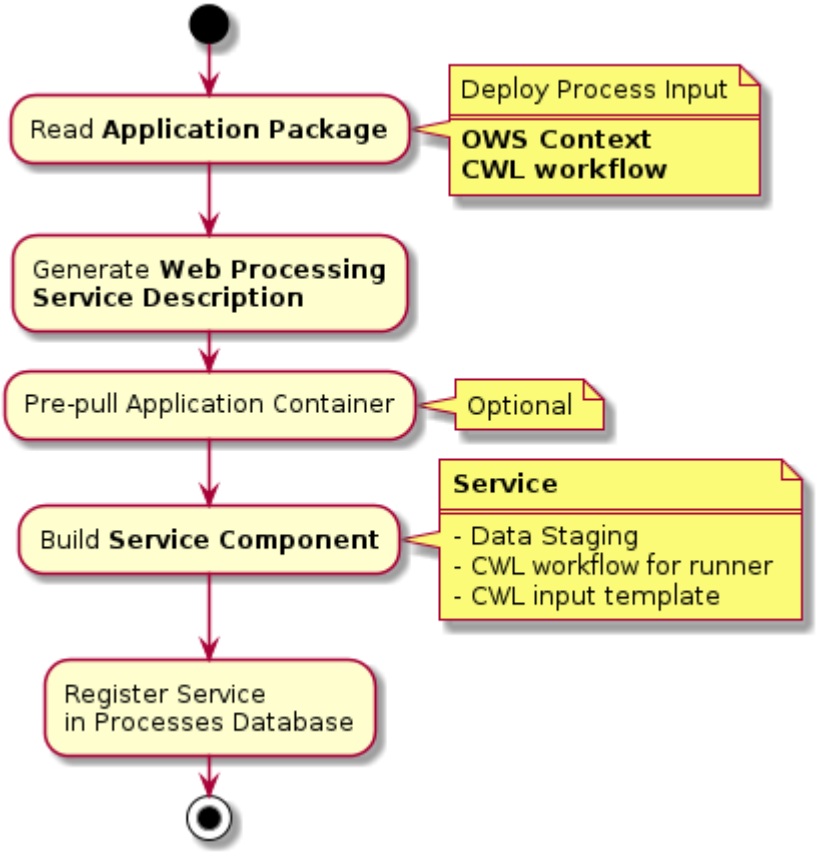


Figure 5. Application Deployment to Web Processing Service Activity Diagram

All software and their dependecies are described in the nextr sections

4.1.2. Data Input/Output

- Input
  - Application Item as an OWS Context describing the application referencing a Application Workflow
- Output
  - Processing Service

4.1.3. Applicable Resources

- [TB13-ADES]
- [TB14-WPS-T]

4.2. Job Submision

4.2.1. Overview and Purpose

The submission of a job starts a process instantiation and execution of an application. Basically, the ADES executes the Service Component built during the application deployment.

This function triggers 3 main operations sequantially

1. Data Stage-In
2. Workflow Execution
3. Data Stage-Out

4.2.2. Data Input/Output

- Input
  - Job Execute Request
- Output
  - Job Identifier (status location)

4.3. Data Stage-In

This is a pre-processing step that provision all the data needed for the workflow execution and referenced in the job execute request.

4.3.1. Overview and Purpose

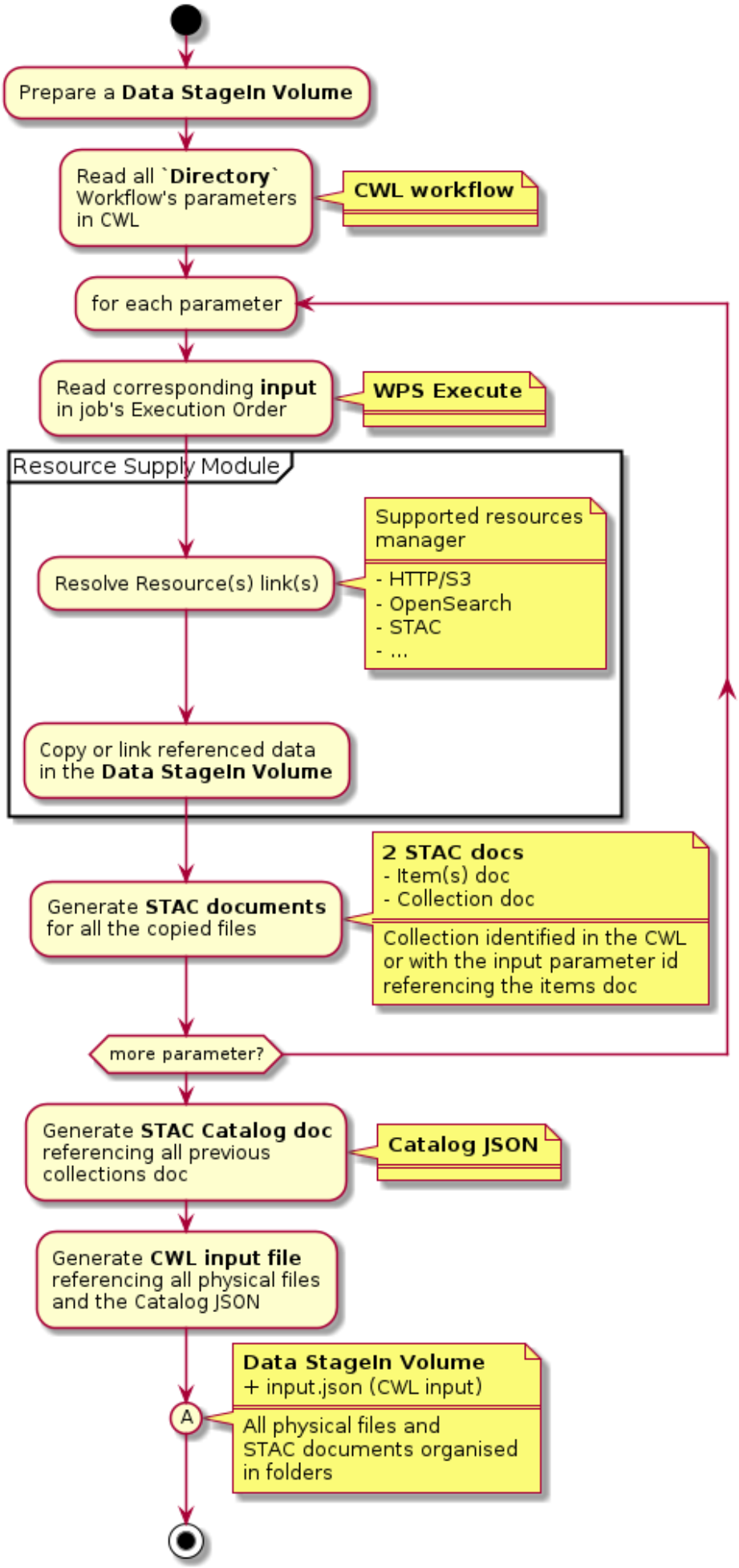

Prepare a **Data StageIn Volume**
Read all `**Directory**` Workflow's parameters in CWL
CWL workflow
for each parameter
Read corresponding **input** in job's Execution Order
WPS Execute
Resource Supply Module
Supported resources manager
Resolve Resource(s) link(s)
- HTTP/S3
- OpenSearch
- STAC
- ...
Copy or link referenced data in the **Data StageIn Volume**
Generate **STAC documents** for all the copied files
2 STAC docs
- Item(s) doc
- Collection doc
Collection identified in the CWL or with the input parameter id referencing the items doc
more parameter?
Generate **STAC Catalog doc** referencing all previous collections doc
Catalog JSON
Generate **CWL input file** referencing all physical files and the Catalog JSON
A
Data StageIn Volume
+ input.json (CWL input)
All physical files and STAC documents organised in folders

Figure 6. ADES Data Stage-In Activity Diagram

The data Stage-In identifies all the **Application Workflow** `inputs` that are of the type Directory. For each of them, if the input is in the **URI** form, it tries to resolve it as a resource manager link and copy the referenced data items on the processing job workspace. It also creates a STAC catalog JSON document referencing the locally copied data. The local directory path containing the catalog.json file and the data is passed to the corresponding CWL `input` at Workflow Execution.

4.3.2. Data Input/Output

- Input
  - Job Execute Request
  - Workflow (CWL)
- Output
  - Input data folder
  - Path to catalog referencing input data folder

4.3.3. Applicable Resources

- Common Workflow Language ([CWL-spec])
- [ZOO]
- [STAC]

4.4. Workflow Execution

Calrissian is the responsible for orchestrating and executing the CWL defining the application. It directly interacts with Kubernetes for submitting jobs on different pods.

4.4.1. Overview and Purpose

4.4.2. Data Input/Output

- Data Staged-in catalog
- Application CWL
- Data produced catalog

4.4.3. Applicable Resources

- [STAC]
- Common Workflow Language ([CWL-spec])

4.5. Data Stage-Out

This is a post-processing step that retrieve all the results produced the workflow execution, copy

and publish them in a resource manager.

4.5.1. Overview and Purpose

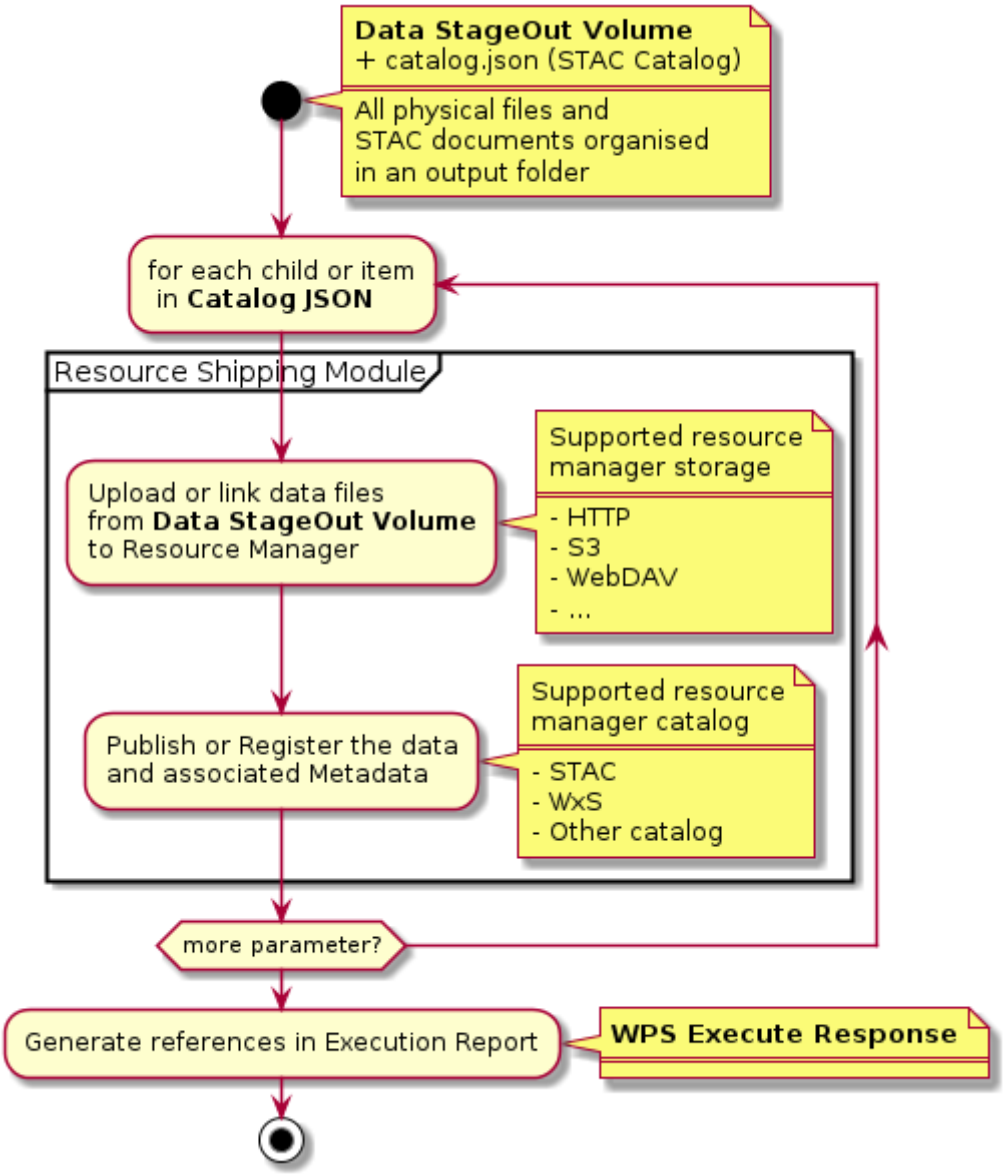


Figure 7. ADES Data Stage-out Activity Diagram

The data Stage-Out identifies all the **Application Workflow** `outputs` that are of the type Directory. For each of them, it creates a STAC catalog JSON document referencing relatively the locally produced data. Then the local directories are stored persistently to the Resource Manager store. A root catalog referencing all the directory catalogs is also stored with the results and published in the Resource Manager Catalog.

4.5.2. Data Input/Output

- Input
  - Job Results folder
  - Catalog referencing results (optional)
- Output

- Reference to catalog published to the resource manager

4.5.3. Applicable Resources

- [STAC]

Chapter 5. External Interfaces

This section focuses on the available functionality published over interfaces.

- List of interfaces

Each interface in the list contains:

- Interface description
- Applicable standards
- Base URL of the interface
- List of exposed port by interface

Each port in the list contains:

- relative URI (*)
- Operation
- Description
- Conveyed information
- Usage examples

5.1. Description

The ADES provides a WPS 1.0 & 2.0 service interfaces as well as an OGC API Processes compliant API. It is responsible for the execution of the processing service (as a WPS request) within the 'target' Exploitation Platform (i.e. one that is close to the data). In the global scenario, we assume that the EMS ensures that the processor is deployed as a WPS service before it is invoked.

Refer to ADES component documentation - https://eoepca.github.io/proc-ades

5.2. Web Processing Services

The ADES exposes interfaces for the processing based on the OGC "Web Processing Service". This service port is available in 3 versions corresponding to the OGC WPS eveolution.
The 2 next sections describes

- The 2 first versions implementing the Web Service Common
- The latest API implementing a REST interface described in OpenAPI

5.2.1. WPS 1.0.0 & 2.0.0

The OpenGIS® Web Processing Service (WPS) Interface Standard provides rules for standardizing how inputs and outputs (requests and responses) for geospatial processing services, such as polygon overlay. The standard also defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of

geospatial processes and clients' discovery of and binding to those processes. The data required by the WPS can be delivered across a network or they can be available at the server.

- Applicable standards - [OGC-WPS] [OGC-WPS2]

Base URL: https://ades.eoepca.org/ows

Table 2. ADES WPS 1.0.0 & 2.0.0 ports

URL	Methods	Function
https://ades.eoepca.org/ows	GET, POST	[ogcowscommon]

OGC Web Service Common port

URL: https://ades.eoepca.org/ows

- Description - The WPS server is self-contained, it provides an initial endpoint that can be used by a WPS client to determine the server's capabilities.
- Operations - GET, POST according to the payload of the requested service
- Usage examples

```
https://ades.eoepca.org/ows?service=WPS&version=1.0.0&request=GetCapabilities
```

- Full references : [OGC-WPS] [OGC-WPS2]

5.2.2. OGC API Processes

The OGC API - Processes enables the execution of computing processes and the retrieval of metadata describing their purpose and functionality. Typically, these processes combine raster, vector, and/or coverage data with well-defined algorithms to produce new raster, vector, and/or coverage information.

- Applicable standards - [OGC-API-PROC]

Base URL: https://ades.eoepca.org/api

Table 3. ADES OGC API Processes ports

Resource	Method	Description	Operation
/	GET	landing page of this API	getLandingPage
/conformance	GET	Lists all requirements classes specified in the standard (e.g., OGC API - Processes Part 1: Core) that the server conforms to	getConformanceClasses
/processes	GET	Lists all available processes this server offers.	getProcesses

Resource	Method	Description	Operation
/processes/{id}	GET	Describes a process.	getProcessDescription
/processes/{id}/jobs	GET	Lists available jobs of a process.	getJobs
/processes/{id}/jobs	POST	Submits a new job.	execute
/processes/{id}/jobs/{jobID}	GET	Shows the status of a job.	getStatus
/processes/{id}/jobs/{jobID}	DELETE	Cancel a job execution and remove it from the jobs list.	dismiss
/processes/{id}/jobs/{jobID}/result	GET	Lists available results of a job. In case of a failure, lists exceptions instead.	getResult

5.3. Application Package

The **Application Package** is the interface for the application developer. It contains 3 essentials elements:

- The **Application Item** is the entry point of the application in a Resource Manager. This item is encoded in a OWS Context document.
- The **Application Workflow** is the processor graph describing the steps execution of the software processor.
- The **Application Image** is the container image including the processor software and its dependencies.

In the following section, we will use the `vegetation-index` application as an example to illustrate the spcifications and convention of the **Application Package**

5.3.1. Application Item

The **Application Item** is the entry point of the application in a Resource Manager. It is an OWS Context document designed to be searchable and queryable as an `Atom entry`.

Below is the **Application Item** of the `vegetation-index` application.
We will describe the essantial elements of this document.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"> ①
    <title type="text">Result for OpenSearch query in eoepca-services</title>
    <id>https://catalog.terradue.com:443/eoepca-services/search?uid=vegetation-
index&amp;format=atom</id>
    <updated>2020-10-12T07:32:51.013305Z</updated>
    <link rel="self" type="application/atom+xml" title="Reference link" href=
"https://catalog.terradue.com/eoepca-services/search?uid=vegetation-index
&amp;format=atom" />
    <link rel="search" type="application/opensearchdescription+xml" title="OpenSearch
Description link" href="https://catalog.terradue.com/eoepca-services/description" />
    <entry> ②
        <id>https://catalog.terradue.com:443/eoepca-services/search?format=atom
&amp;uid=vegetation-index</id>
        <title type="text">Vegetation index</title> ③
        <summary type="html">Vegetation index processor</summary> ④
        <published>2020-10-12T07:32:51.012977Z</published>
        <updated>2020-10-25T10:34:19.721262Z</updated> ⑤
        <link rel="self" type="application/atom+xml" title="Reference link" href=
"https://catalog.terradue.com/eoepca-services/search?format=atom&amp;uid=vegetation-
index" /> ⑥
        <owc:offering code="http://www.opengis.net/eoc/applicationContext/cwl"
xmlns:owc="http://www.opengis.net/owc/1.0"> ⑦
            <owc:content type="application/cwl" href=
"https://raw.githubusercontent.com/EOEPCA/proc-
ades/develop/test/sample_apps/vegetation-index/vegetation-index.cwl" /> ⑧
        </owc:offering>
        <identifier xmlns="http://purl.org/dc/elements/1.1/">vegetation-
index</identifier> ⑨
    </entry>
    <totalResults xmlns="http://a9.com/-/spec/opensearch/1.1/">1</totalResults>
    <startIndex xmlns="http://a9.com/-/spec/opensearch/1.1/">1</startIndex>
    <itemsPerPage xmlns="http://a9.com/-/spec/opensearch/1.1/">20</itemsPerPage>
    <os:Query role="request" xmlns:dct="http://purl.org/dc/terms/" xmlns:param=
"http://a9.com/-/spec/opensearch/extensions/parameters/1.0/" xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:os="http://a9.com/-/spec/opensearch/1.1/"
geo:uid="vegetation-index" />
</feed>
```

① The Atom Feed enclosing the Application Item. in our case, the feed is the result of an OpenSearch query to a Catalog.

② The **Application Item** as an Atom Entry

③ The **Application Title** used to label the Processing Service deployed on the ADES.

④ The **Application Summary** used as the description of the Processing Service deployed on the ADES.

⑤ The **Application Date** used to timestamp the Processing Service deployed on the ADES.

⑥ Following the OpenSearch protocol, the self link is the persistent link to the resource. this is

often managed automatically by the Resource Manager

⑦ The OWS Context `offering` declaring the **Application Workflow** with the specific code `http://www.opengis.net/eoc/applicationContext/cwl`.

⑧ The actual **Application Workflow** resource. It can be either enclosed in the document or referenced with a link like in this example. The Mime-Type is `application/cwl`.

⑨ The **Application Identifier** used to **uniquely** identify the Processing Service deployed on the ADES.

5.3.2. Application Workflow

The **Application Workflow** is the core document of the processor in Common Workflow Language ([CWL-spec]). It describes, as a Directed Acyclic Graph (DAG), the execution of the user processor defining the commands to be executed with their arguments as well as the inputs/outputs.
The ADES relies on the Common Workflow Language ([CWL-spec]) tool via [Calrissian] to execute the workflow. Thus, the ADES supports entirely the Common Workflow Language ([CWL-spec]) specifications. Therefore, we will not explain in details here the Common Workflow Language ([CWL-spec]) specifications available as a Reference Documents.
The purpose of this section is to describe the elemnts used in the CWL in the functional design of the ADES.

Below is the **Application Workflow** Common Workflow Language ([CWL-spec]) of the `vegetation-index` application.
We will describe the essantial elements of this document.

```
$graph:
  cwlVersion: v1.0
  - class: CommandLineTool
    id: clt
    baseCommand: vegetation-index
    inputs:
      inp1:
        inputBinding:
          position: 1
          prefix: --input-reference
        type: Directory
      inp2:
        inputBinding:
          position: 2
          prefix: --aoi
        type: string
    outputs:
      results:
        outputBinding:
          glob: .
        type: Any
    hints:
      DockerRequirement:
        dockerPull: terradue/eoepca-vegetation-index:0.1 ⑦
```

```
    requirements:
      EnvVarRequirement:
        envDef:
          PATH:
/opt/anaconda/envs/env_vi/bin:/opt/anaconda/envs/env_vi/bin:/opt/anaconda/envs/env_def
ault/bin:/opt/anaconda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n
          PREFIX: /opt/anaconda/envs/env_vi
      ResourceRequirement: {}
  - class: Workflow ①
    doc: Vegetation index processor
    id: vegetation-index
    inputs: ②
      aoi:
        doc: Area of interest in WKT
        label: Area of interest
        type: string ③
      input_reference:
        doc: EO product for vegetation index
        label: EO product for vegetation index
        stac:catalog: catalog.json
        stac:collection: source
        type: Directory[] ③
    label: Vegetation index
    outputs: ④
    - id: wf_outputs
      outputSource:
      - node_1/results
      type:
        items: Directory ⑤
        type: array
    requirements:
    - class: ScatterFeatureRequirement ⑥
    steps:
      node_1:
        in:
          inp1: input_reference
          inp2: aoi
        out:
        - results
        run: '#clt'
        scatter: inp1
        scatterMethod: dotproduct
```

① the `Workflow` class object of the Common Workflow Language ([CWL-spec]) is the main element used by the ADES to extract the information about the inputs/outputs of the Application

② the `inputs` section defines all the input necessary for the command. When the **Application Package** is deployed as as Processing Service, the `inputs` used by the WPS Server to declare the processing inputs at the Web Processing Services interface.

③ the `type` used for `input` is replicated at Web Processing Services interface level to describe the

exepected input. A specific convention is applied when using the type `Directory` and is explained in the Data Stage-In section.

④ the `outputs` section defines the outputs of the command. When the **Application Package** is deployed as as Processing Service, the `outputs` used by the WPS Server to declare the processing outputs at the Web Processing Services interface.

⑤ the `type` used for `output` is replicated at Web Processing Services interface level to describe the produced output. A specific convention is applied when using the type `Directory` and is explained in the Data Stage-Out section.

⑥ the scattering functions of CWL is replicated in the ADES with [Calrissian] using different virtual machines instances to run the process in parallel.

⑦ All the **Application Image** specified in the **Application Workflow** are pulled and used to instantiate the processing container on the cluster.

5.3.3. Application Image

One or more **Application Image** can be used in the **[Application PAckage]** to support the execution of the processor. The **Application Image** is specified as a [Docker] Image in the **Application Workflow**

<< End of Document >>