Master System Interface Control Document

EOEPCA.ICD.001

TVUK System Team

Version Draft for 0.1, In Progress:

Master System Interface Control Document

1. Introduction . . . 2
   1.1. Purpose and Scope . . . 2
   1.2. Structure of the Document . . . 2
   1.3. Reference Documents . . . 2
   1.4. Terminology . . . 5
   1.5. Glossary . . . 9
2. Overview . . . 12
3. User Management . . . 14
   3.1. Login Service . . . 14
      3.1.1. Description . . . 14
      3.1.2. Context . . . 14
      3.1.3. Provided Interfaces . . . 14
         3.1.3.1. Administration Web Interface . . . 15
         3.1.3.2. OIDC Interface . . . 16
         3.1.3.3. UMA Interface . . . 16
         3.1.3.4. SCIM Interface . . . 17
      3.1.4. Required Interfaces . . . 17
         3.1.4.1. XACML Policy Checks . . . 17
      3.1.5. Example Scenarios . . . 18
         3.1.5.1. End-User Authentication . . . 18
         3.1.5.2. Resource Access . . . 18
         3.1.5.3. Identity Management . . . 18
   3.2. Policy Enforcement Point (PEP) . . . 19
      3.2.1. Description . . . 19
      3.2.2. Context . . . 19
      3.2.3. Provided Interfaces . . . 19
         3.2.3.1. Platform Resource API . . . 20
         3.2.3.2. Secured Proxy . . . 20
      3.2.4. Required Interfaces . . . 20
         3.2.4.1. OIDC Authentication . . . 21
         3.2.4.2. UMA Authorization . . . 21
         3.2.4.3. Protected Resource Server . . . 22
         3.2.4.4. Policy Decision Point (PDP) . . . 22
      3.2.5. Example Scenarios . . . 22
         3.2.5.1. Resource Access . . . 22
         3.2.5.2. Resource Management . . . 23
   3.3. Policy Decision Point (PDP) . . . 25
      3.3.1. Description . . . 25

3.3.2. Context . . . 25
3.3.3. Provided Interfaces . . . 25
3.3.3.1. Platform Policy API . . . 26
3.3.3.2. XACML Policy Checks . . . 26
3.3.4. Required Interfaces . . . 26
3.3.4.1. OIDC Authentication . . . 27
3.3.4.2. UMA Authorization . . . 27
3.3.4.3. SCIM Identity Management . . . 28
3.3.5. Example Scenarios . . . 28
3.3.5.1. Policy Access Check . . . 28
3.3.5.2. Policy Management . . . 28
3.4. User Profile . . . 30
3.4.1. Description . . . 30
3.4.2. Context . . . 30
3.4.3. Provided Interfaces . . . 30
3.4.3.1. Profile Management Web Interface . . . 31
3.4.4. Required Interfaces . . . 31
3.4.4.1. OIDC Authentication . . . 31
3.4.4.2. SCIM Identity Management . . . 32
3.4.4.3. Email Service . . . 32
3.4.5. Example Scenarios . . . 32
3.4.5.1. Attribute Edition . . . 32
3.4.5.2. Account Deletion . . . 33
3.5. License Manager . . . 34
3.6. Billing Service . . . 35
3.7. Pricing Engine . . . 36
4. Processing & Chaining . . . 37
4.1. Application Deployment and Execution Service (ADES) . . . 37
4.1.1. Description . . . 37
4.1.2. Context . . . 37
4.1.3. Provided Interfaces . . . 38
4.1.3.1. WPS 1.0.0 & 2.0.0 . . . 38
4.1.3.2. OGC API Processes . . . 40
4.1.3.3. Application Invocation . . . 41
4.1.4. Required Interfaces . . . 45
4.1.4.1. Resource Managers . . . 46
4.1.5. Example Scenarios . . . 47
4.1.5.1. ADES Processing Flows . . . 47
4.2. Execution Management Service (EMS) . . . 51
4.3. Workflow Engine . . . 52
4.4. Processor Development Environment (PDE) . . . 53

4.5. Interactive Analysis Tool (IAT) . . . 54
5. Resource Management . . . 55
5.1. Resource Catalogue . . . 55
5.2. Data Access Services . . . 56
5.3. Data Access Gateway . . . 57
5.4. Data Access Library . . . 58
5.5. Data Ingestion . . . 59
5.6. Workspace . . . 60

EO Exploitation Platform Common Architecture

Master System Interface Control Document

EOEPCA.ICD.001

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/master-system-icd/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	dd/mm/yyyy	In-progress draft

Chapter 1. Introduction

1.1. Purpose and Scope

This is the Master System Interface Control Document for the Common Architecture.

1.2. Structure of the Document

Section 2 - [mainContext]

Provides the context for Exploitation Platforms within the ecosystem of EO analysis.

Section 3 - [mainDesignOverview]

Provides an overview of the Common Architecture and the domain areas.

Section 4 - User Management

Describes the User Management domain area.

Section 5 - Processing & Chaining

Describes the Processing & Chaining domain area.

Section 6 - Resource Management

Describes the Resource Management domain area.

Section 7 - [mainPlatformApi]

Describes the Platform API, covering all domain areas.

Section 8 - [mainWebPortal]

Describes the Web Portal, covering all domain areas.

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-MSDD]	EOEPCA - Master System Design Document EOEPCA.SDD.001 https://eoepca.github.io/master-system-design	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

Reference	Document Details	Version
[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-WPS]	OGC Web Processing Service, OGC 05-007, http://portal.opengeospatial.org/files/?artifact_id=24151	05-007r7, 08/06/2007
[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://github.com/opengeospatial/wps-rest-binding	1.0-draft.4
[OGC-COMMON]	OGC Web Services Common Standard, OGC 06-121r9, https://portal.ogc.org/files/?artifact_id=38867&version=2	06-121r9 v2.0.0, 07/04/2010
[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

Reference	Document Details	Version
[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
[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

Reference	Document Details	Version
[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
[STAC-SPEC]	SpatioTemporal Asset Catalog specification, https://github.com/radiantearth/stac-spec	1.0.0-beta.2
[STAC-API]	SpatioTemporal Asset Catalog API specification, https://github.com/radiantearth/stac-api-spec/	1.0-beta

1.4. Terminology

The following terms are used in the Master System Interface Control Document.

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)	WPS-T (REST/JSON) 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.
Network of EO 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
MSDD	(EOEPCA) Master System Design Document
MSICD	(EOEPCA) Master System Interface Control Document
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

Term	Definition
WPS	Web Processing Service
WPS-T	Transactional Web Processing Service
XACML	eXtensible Access Control Markup Language

Chapter 2. Overview

This Master System Interface Control Document (MSICD) is a companion to the Master System Design Document [EOEPCA-MSDD].

[EOEPCA-MSDD] defines system building-blocks organised within three domain areas…

- **User Management:** Responsible for all aspects related to user identity, authentication, authorization, accounting and billing in a federated system-of-systems environment.
- **Processing and Chaining:** Provides access to a variety of processing functions, tools and applications, as well as execution environments in which to deploy them.
- **Resource Management:** Responsible for maintaining an inventory of platform and federated resources, and providing services for data access and visualisation.

The MSICD provides a system-level specification of the interfaces between all building-blocks comprising the EOEPCA system architecture. The document is organised by domain, and by building-block. The context of each building block is expressed by enumeration of the interfaces it **provides**, and the interfaces of other building blocks and external entities that it **consumes**. Provided interfaces are consumed by other building blocks, and may also be presented as platform 'public' interfaces for external consumption.

Section User Management

Provides the interface specification for the following building blocks:

- Login Service
- Policy Enforcement Point (PEP)
- Policy Decision Point (PDP)
- User Profile
- License Manager
- Billing Service
- Pricing Engine

Section Processing & Chaining

Provides the interface specification for the following building blocks:

- Application Deployment and Execution Service (ADES)
- Execution Management Service (EMS)
- Workflow Engine
- Processor Development Environment (PDE)
- Interactive Analysis Tool (IAT)

Section Resource Management

Provides the interface specification for the following building blocks:

- Resource Catalogue
- Data Access Services
- Data Access Gateway
- Data Access Library
- Data Ingestion
- Workspace

Chapter 3. User Management

3.1. Login Service

Refer to Login Service component documentation - https://eoepca.github.io/um-login-service/master

3.1.1. Description

The Login Service building block provides an OIDC and UMA compliant solution enabling authentication and authorization mechanisms within an Exploitation Platform. Other building blocks, such as the Policy Decision Point, Policy Enforcement Point and User Profile rely on this Building Block to provide several standard interfaces.

3.1.2. Context

The following context diagram identifies the major components interfacing with the Login Service: (RED ~ consumers of Login Service provided interfaces, BLUE ~ providers of interfaces consumed by the Login Service)

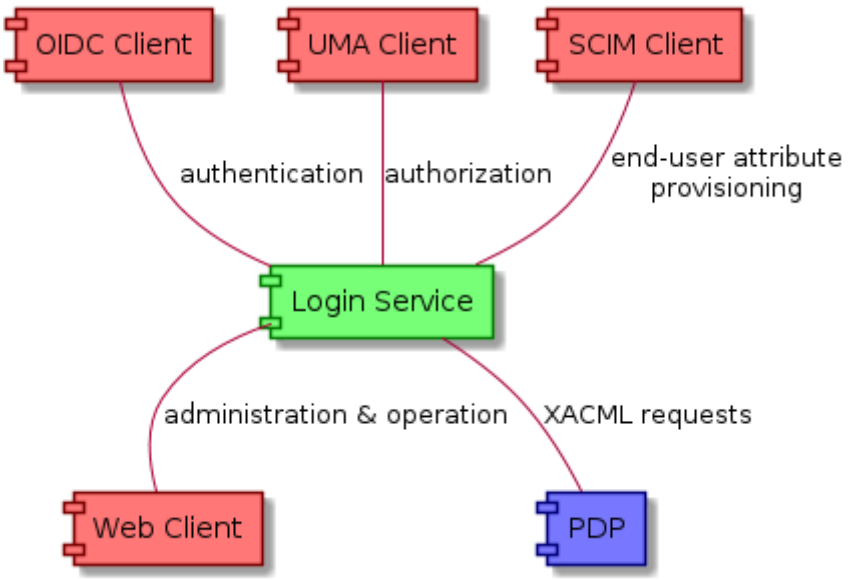

Figure 1. Login Service context diagram

The EOEPCA Login Service provides implementations of the state-of-the-art standards and supports all building blocks that require authentication (through OIDC), authorization (through UMA) and user management capabilities (through SCIM). In addition to this, the Login Service exposes a Web Interface that can be used to perform monitoring and configuration management by operators.

The Login Service can also rely on XACML-based flows to verify access rights during an authorization flow, contacting mainly the XACML endpoints exposed by **Policy Decision Point (PDP)**.

3.1.3. Provided Interfaces

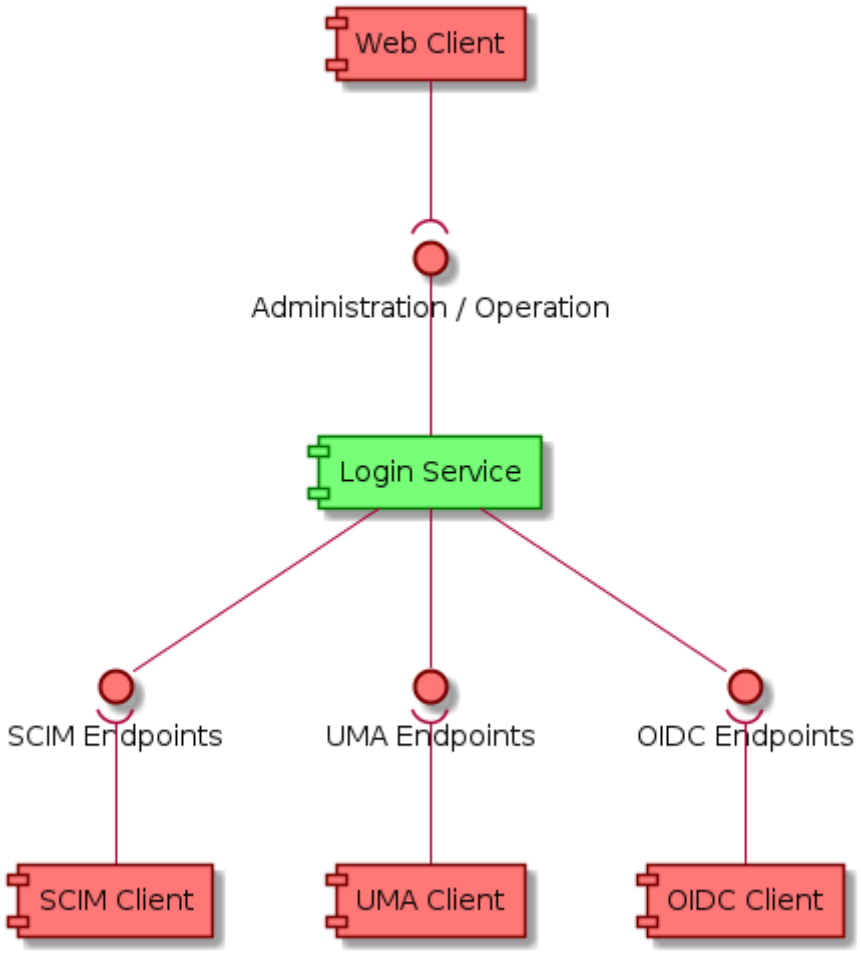

Figure 2. Login Service provided interfaces

The Login Service enables the following interfaces:

- Administration Web Interface - Web Administration interface meant for Operators
- OIDC Interface - OIDC compliant endpoint allowing acting as OpenID Provider (OP)
- [mainUMA] - UMA compliant endpoints acting as Authorization Server (AS)
- [mainSCIM] - SCIM compliant endpoints allowing User Management and user information retrieval mechanisms

3.1.3.1. Administration Web Interface

The Login Service provides administration functionality for users with elevated priviledges. This allows admins to have an overview of the entire system, as well as configure anything necessary from this interface.

This interface allows manual editing, creation and removal of OIDC Client configurations, UMA Scopes, Custom Polices and base configuration parameters.

3.1.3.1.1. Applicable Standards

No applicable standards besides Hypertext Transfer Protocol (HTTP) as means to serve the information to the End-User

3.1.3.1.2. Endpoints

URL(1): /
URL(2): /identity

By default, the Login Service administration portal resides on the root path of the deployment. This root path usually redirects to the management service located at:

3.1.3.2. OIDC Interface

The Login Service provides an OIDC compliant interface that allows Requesting Parties to authenticate End-Users through their use of credentials. It mainly acts as an identity layer on top of OAuth2.0

3.1.3.2.1. Applicable Standards

- IETF - OpenID Connect 1.0

3.1.3.2.2. Endpoints

URL: /.well-known/openid-configuration

The Login Service exposes the standard discovery document, a JSON showcasing all the necessary metadata and endpoint enumeration for a client application to dynamically load all the needed endpoints.

3.1.3.3. UMA Interface

The Login Service provides a UMA compliant interface that allows Resource Servers (or Policy Enforcement Points running in front of these) to request valid Requesting Party Tokens (RPTs). UMA is a custom profile of OAuth2.0 that introduces a ticket-based profile, similar to the "authorization code" flow but keeping it agnostic in terms of authentication.

UMA mainly manages the notion of "resource" as the main managed object, with tickets being a representation of an attempt to access any given resource within a specific realm. In order to facilitate this, several resource management endpoints (registration, deletion, scope assignment) are made available and the standard intentionally leaves Policy resolution (access checks) out of the specification to facilitate integration with other technologies (XACML, and any given policy language).

3.1.3.3.1. Applicable Standards

- Kantara Innitiative - UMA 2.0

3.1.3.3.2. Endpoints

URL: /.well-known/uma2-configuration

The Login Service exposes a dicovery document very much similar to OIDC well-known documents. Although this is not implicitly stated as necessary part of a UMA-compliant solution, serving a well-known endpoint makes client integration more familiar to developers that may be used to OIDC Client integration.

3.1.3.4. SCIM Interface

The Login Service provides a SCIM compliant interface, designed to make managing user identities in cloud-based applications and services easier due to its well-defined schemas and endpoints.

3.1.3.4.1. Applicable Standards

- IETF RFC 7644 - System for Cross-domain Identity Management ~

3.1.3.4.2. Endpoints

URL: /.well-known/scim-configuration

Similarly to OIDC and UMA endpoints, this well-known endpoint allows to discover all relevant SCIM operations. Although this strategy is not enforced, serving a well-known endpoint makes client integration more familiar to developers that may be used to OIDC Client integration.

3.1.4. Required Interfaces

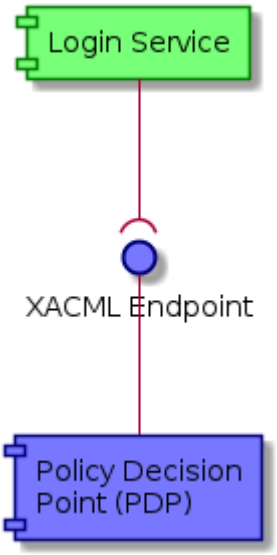

Figure 3. Login Service required interfaces

The Login Service utilizes the following interfaces:

- [loginServiceXACML] - XACML compliant interface used to perform queries against a Policy Decision Point (PDP).

3.1.4.1. XACML Policy Checks

The Login Service might delegate policy checks, as part of its UMA workflow execution, to an external service that exposes an XACML-compliant interface (in this architecture, this could be provided by the Policy Decision Point). This standard mostly covers request and response format for a policy decision service, and includes a JSON binding that fully maps the original XML schemas to JSON schemas that can be used in XACML compliant request generation and response parsing.

3.1.4.1.1. Applicable Standards

- OASIS - eXtensible Access Control Markup Language (XACML) Version 3.0 ~

- OASIS (Public Draft) - JSON Profile of XACML 3.0 Version 1.1 ~

3.1.4.1.2. Remote Endpoints

URL: <pdp_instance>/policy/validate

Within EOEPCA, the remote endpoint stated above and exposed by the PDP will accept XACML compliant requests.

3.1.5. Example Scenarios

3.1.5.1. End-User Authentication

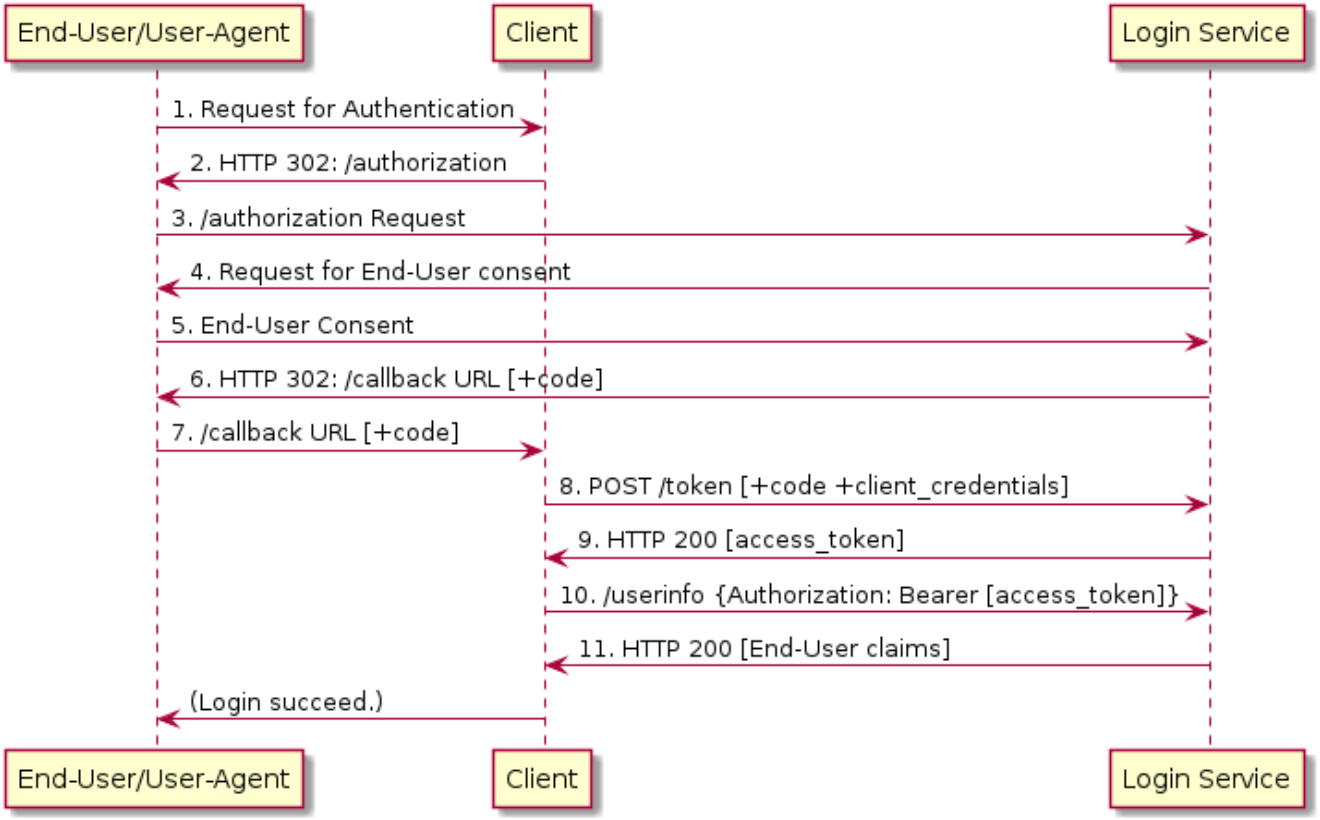

Figure 4. Login Service provided interfaces

3.1.5.2. Resource Access

For an in detail scenario involving resource access, please refer to the Policy Enforcement Point section and its use of the UMA Endpoints

3.1.5.3. Identity Management

For an in detail scenario involving identity resource management, please refer to the Policy Decision Point section and its use of the SCIM Endpoints

3.2. Policy Enforcement Point (PEP)

Refer to PEP building block documentation - https://eoepca.github.io/um-pep-engine/master

3.2.1. Description

The Policy Enforcement Point (PEP) allows unsecured Resource Servers to be integrated in the architecture, by providing the means to securely register, manage and expose resources. Any Web Service provided by the architecture can potentially run an ad-hoc PEP service that serves as secure contact point and handles all the necessary UMA Flow steps.

3.2.2. Context

The following context diagram identifies the major components interfacing with the PEP:
(RED ~ consumers of PEP provided interfaces, BLUE ~ providers of interfaces consumed by the PEP)

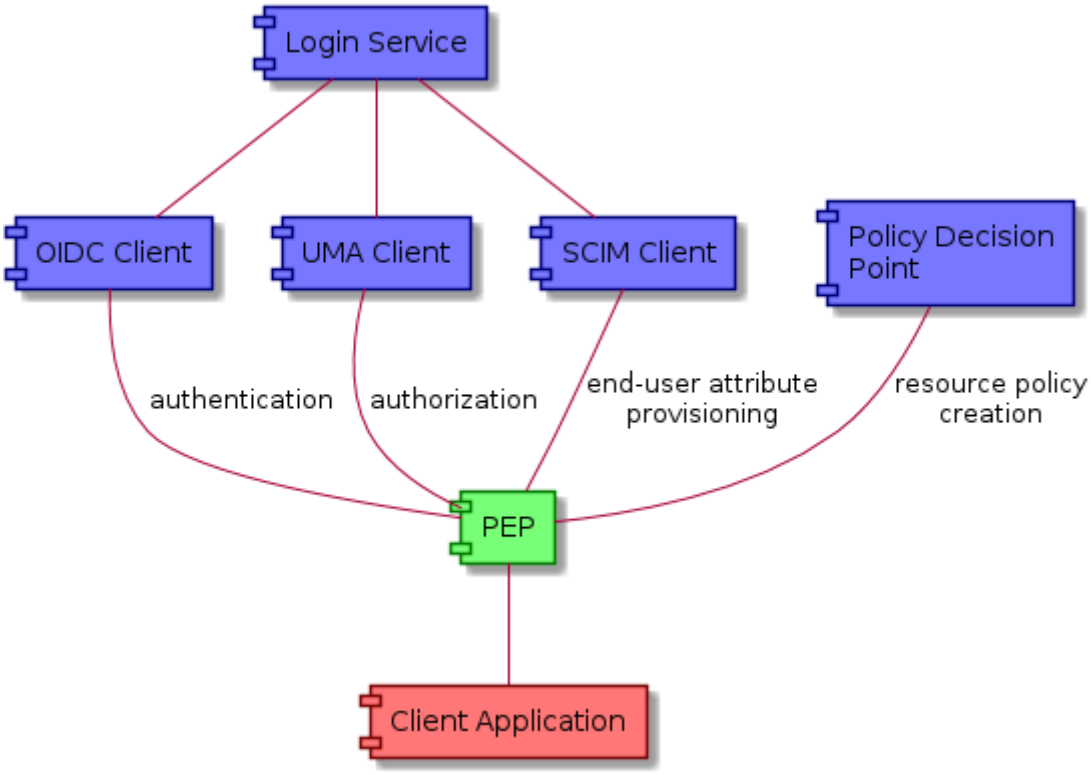

Figure 5. Policy Enforcement Point context diagram

EOEPCA PEP Service provides a Platform Resource management API that abstracts the UMA Standard and facilitates its usage by Resource Owners. It also exposes unsecured endpoints of Resource Servers through its secure proxying endpoint.

The PEP relies on UMA to implement the necessary steps of the ticket-based authorization flow. It also allows propagation of End-User claims towards the Resource Server.

3.2.3. Provided Interfaces

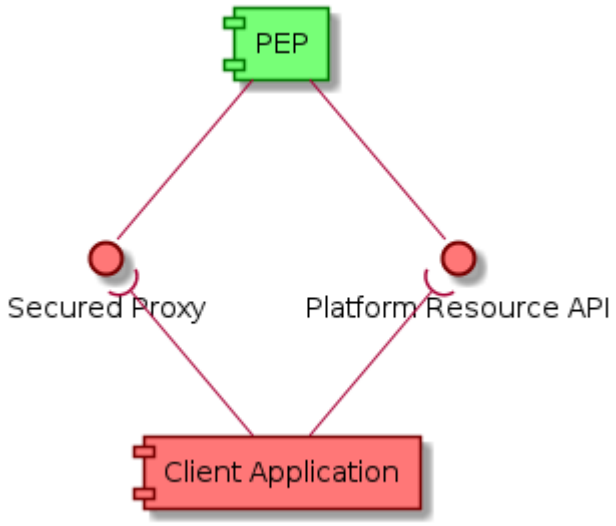

Figure 6. Policy Enforcement Point provided interfaces

3.2.3.1. Platform Resource API

3.2.3.1.1. Applicable Standards

No applicable standards besides Hypertext Transfer Protocol (HTTP) and a RESTful API approach allowing CRUD operations over Resource reference objects.

3.2.3.1.2. Endpoints

URL: <um-pep-engine>:<resources_port>/resources/{id}

3.2.3.2. Secured Proxy

3.2.3.2.1. Applicable Standards

No applicable standards besides the capability to parse and enrich Hypertext Transfer Protocol (HTTP) operations .

3.2.3.2.2. Endpoints

URL: <um-pep-engine>:<proxy_port>/proxy/{back-end-url}

3.2.4. Required Interfaces

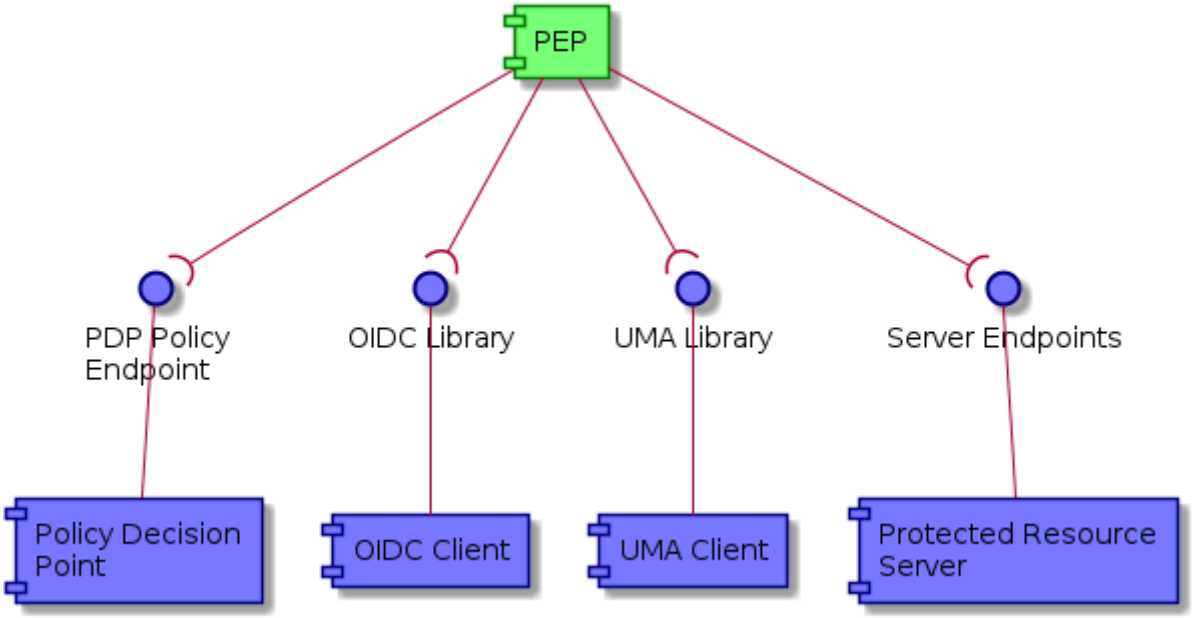

Figure 7. Policy Enforcement Point required interfaces

3.2.4.1. OIDC Authentication

The PEP utilizes an OIDC Client implementation to consume OIDC endpoints, authenticate itself as trusted platform component, generate JWT documents containing End-User information, and secure its Platform Resource API endpoints.

3.2.4.1.1. Applicable Standards

- IETF - OpenID Connect 1.0

3.2.4.1.2. Remote Endpoints

URL: <um-login-service>/.well-known/openid-configuration

The Login Service exposes the standard discovery document, a JSON showcasing all the necessary metadata and endpoint enumeration for a client application to dynamically load all the needed endpoints.

3.2.4.2. UMA Authorization

The PEP utilizes a UMA Client implementation to consume UMA Endpoints, enforcing authorization of all requests made to the secure proxy endpoint (generation of tickets and RPT validation)

3.2.4.2.1. Applicable Standards

- Kantara Innitiative - UMA 2.0

3.2.4.2.2. Remote Endpoints

URL: <um-login-service>/.well-known/uma2-configuration

The Login Service exposes a dicovery document very much similar to OIDC well-known documents. Although this is not implicitly stated as necessary part of a UMA-compliant solution, serving a well-

known endpoint makes client integration more familiar to developers that may be used to OIDC Client integration.

3.2.4.3. Protected Resource Server

The PEP sits in front of a Resource Server and propagates all requests and responses (acting as an intermediate client). The only modifications performed by the PEP consist in the inclusion of an HTTP Header containing relevant End-User information.

3.2.4.3.1. Applicable Standards

No applicable standards besides the capability to parse and enrich Hypertext Transfer Protocol (HTTP) responses.

3.2.4.3.2. Remote Endpoints

The endpoint protected by the PEP is a configurable parameter and depends on the characteristics of the Resource Server running behing the PEP.

3.2.4.4. Policy Decision Point (PDP)

The PEP will interact with the PDP in order to assign a default access policy when registering a new resource.

3.2.4.4.1. Applicable Standards

No applicable standards besides the capability to parse and enrich Hypertext Transfer Protocol (HTTP) responses.

3.2.4.4.2. Remote Endpoints

URL: <um-pdp-engine>/policy
The endpoint name is configurable on the PDP side, with this being the default configuration.

3.2.5. Example Scenarios

3.2.5.1. Resource Access

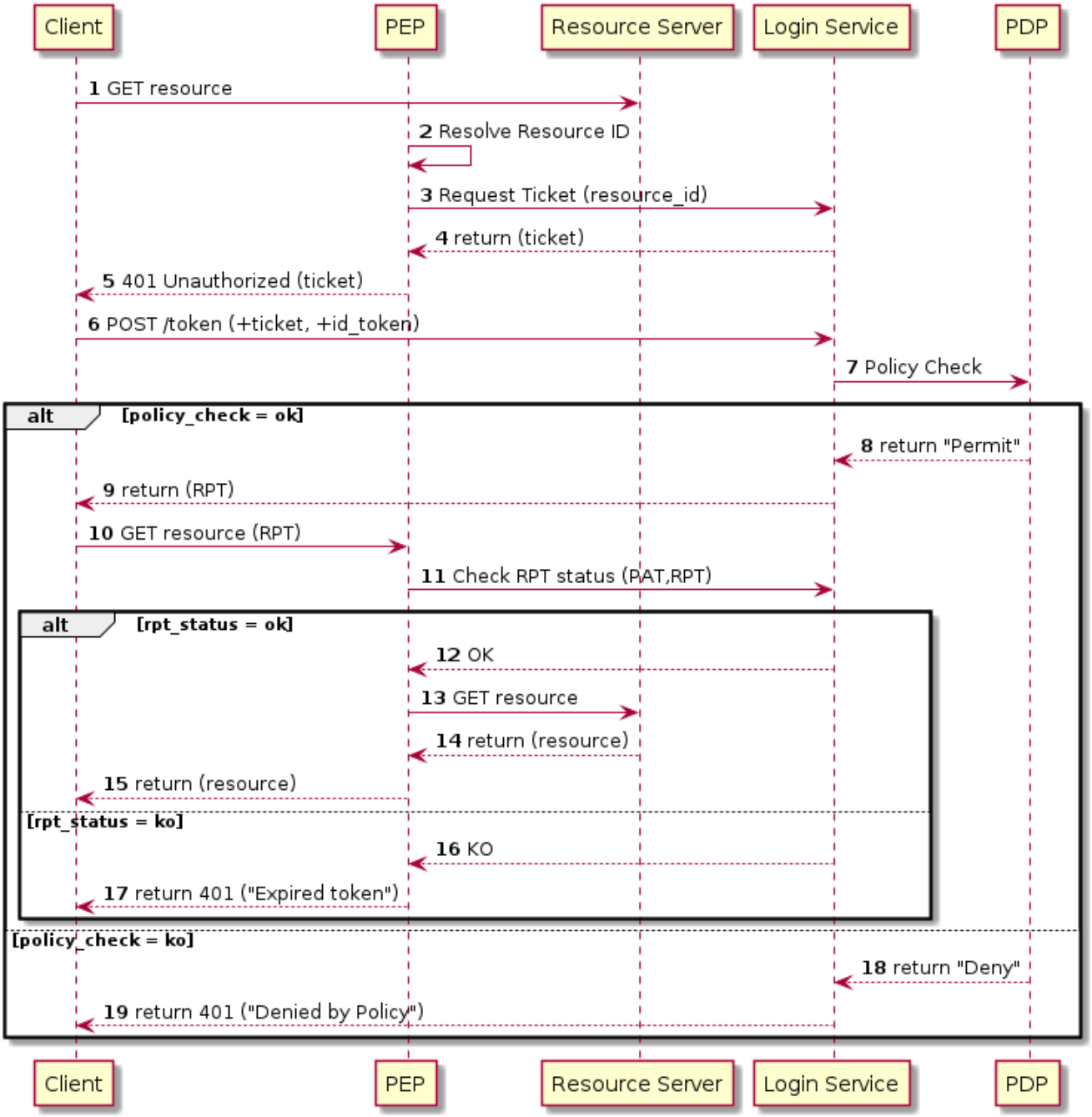

Figure 8. Access Resource Scenario

3.2.5.2. Resource Management

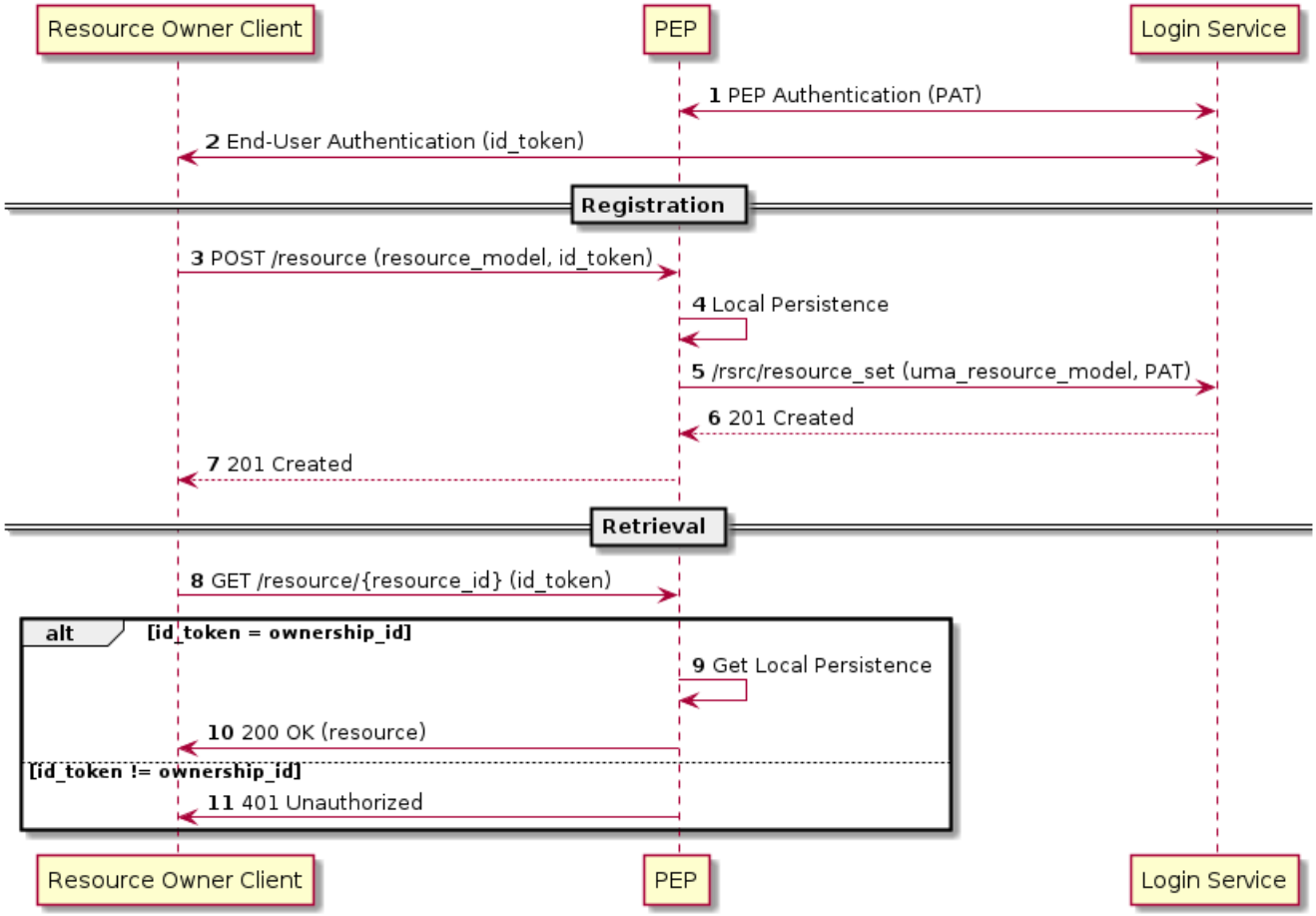

Figure 9. Register and Retrive Resource Reference Scenario

3.3. Policy Decision Point (PDP)

Refer to PDP building block documentation - https://eoepca.github.io/um-pdp-engine/master

3.3.1. Description

The Policy Decision Point (PDP) provides an XACML compliant decision endpoint that can be both consumed from entities which are external to the platform, and the components participanting in a UMA Flow (Login Service and PEP). It also allows to assign specific access policies to a resource.

3.3.2. Context

The following context diagram identifies the major components interfacing with the PDP:
(RED ~ consumers of PDP provided interfaces, BLUE ~ providers of interfaces consumed by the PDP)

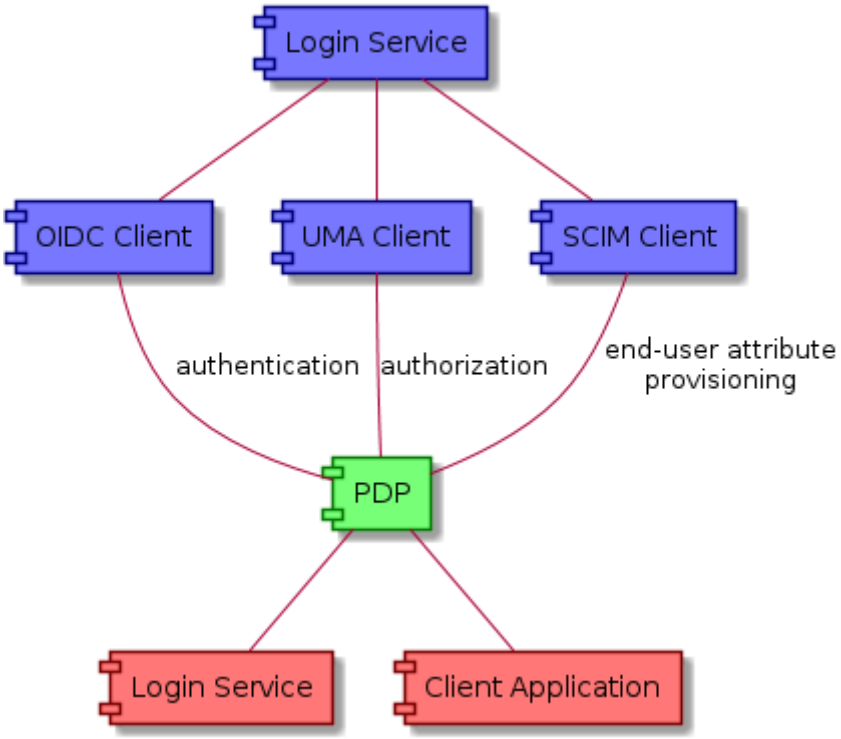

Figure 10. Policy Decision Point context diagram

EOEPCA's PDP Service interacts mainly with the Login Service to both consume its SCIM Endpoints (as source of End-User information) and serve an XACML compliant endpoint to support Policy-based Access Control. On the other hand, Client Applications used by Resource Owners can interact with the Policy Management API to manipulate the access contraints for any given resource.

3.3.3. Provided Interfaces

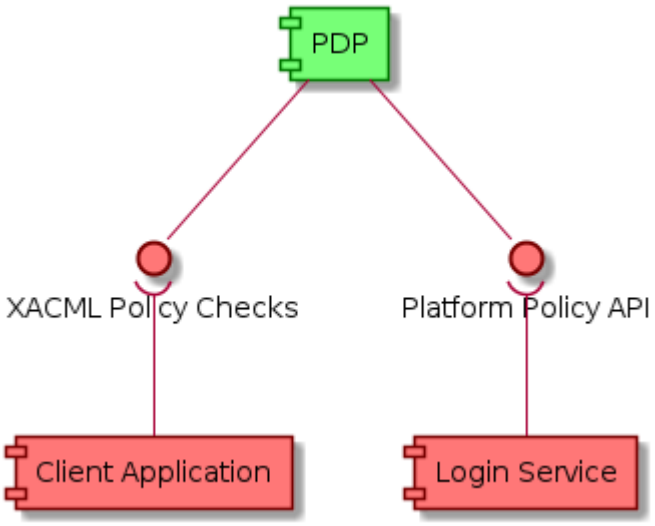

Figure 11. Policy Decision Point provided interfaces

3.3.3.1. Platform Policy API

3.3.3.1.1. Applicable Standards

No applicable standards besides Hypertext Transfer Protocol (HTTP) and a RESTful API approach allowing CRUD operations over Resource reference objects.

3.3.3.1.2. Endpoints

URL: /policies/{id}

3.3.3.2. XACML Policy Checks

3.3.3.2.1. Applicable Standards

- OASIS - eXtensible Access Control Markup Language (XACML) Version 3.0 ~
- OASIS (Public Draft) - JSON Profile of XACML 3.0 Version 1.1 ~

3.3.3.2.2. Endpoints

URL: /policy/validate

This XACML Policy Check endpoint can potentially be available as a Platform interface, allowing external entities to perform policy checks when a Platform Resource is being consumed from outside of its Platform.

3.3.4. Required Interfaces

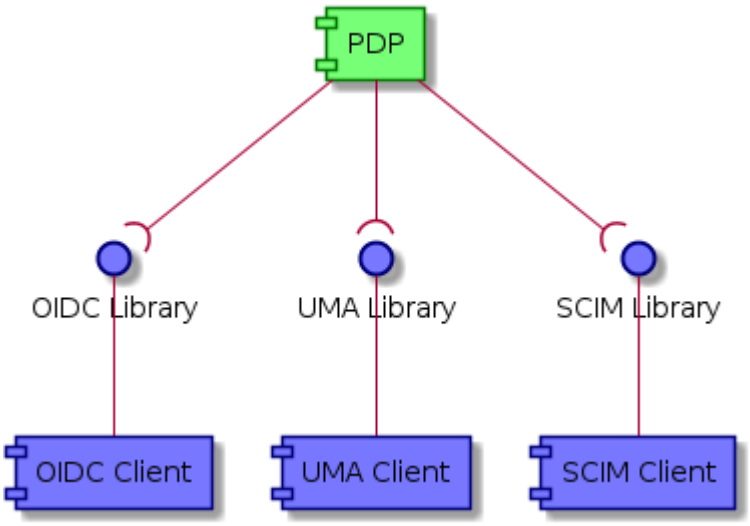

Figure 12. Policy Decision Point required interfaces

3.3.4.1. OIDC Authentication

The PDP utilizes an OIDC Client implementation to consume OIDC endpoints, and authenticate itself as trusted platform component, and secure its Policy API endpoints.

3.3.4.1.1. Applicable Standards

- IETF - OpenID Connect 1.0

3.3.4.1.2. Remote Endpoints

URL: <um-login-service>/.well-known/openid-configuration

The Login Service exposes the standard discovery document, a JSON showcasing all the necessary metadata and endpoint enumeration for a client application to dynamically load all the needed endpoints.

3.3.4.2. UMA Authorization

The PDP utilizes a UMA Client implementation that allows to successfully access SCIM protected endpoints.

3.3.4.2.1. Applicable Standards

- Kantara Innitiative - UMA 2.0

3.3.4.2.2. Remote Endpoints

URL: <um-login-service>/.well-known/uma2-configuration

The Login Service exposes a dicovery document very much similar to OIDC well-known documents. Although this is not implicitly stated as necessary part of a UMA-compliant solution, serving a well-known endpoint makes client integration more familiar to developers that may be used to OIDC Client integration.

3.3.4.3. SCIM Identity Management

The PDP utilizes an SCIM Client implementation that allows to retrieve information about End-Users and take it into account for its policy decision functionality

3.3.4.3.1. Applicable Standards

- IETF RFC 7644 - System for Cross-domain Identity Management ~

3.3.4.3.2. Remote Endpoints

URL: <um-login-service>/.well-known/scim-configuration

Similarly to OIDC and UMA endpoints, this well-known endpoint allows to discover all relevant SCIM operations, allowing the PDP to dynamically load the list of endpoints to use.

3.3.5. Example Scenarios

3.3.5.1. Policy Access Check

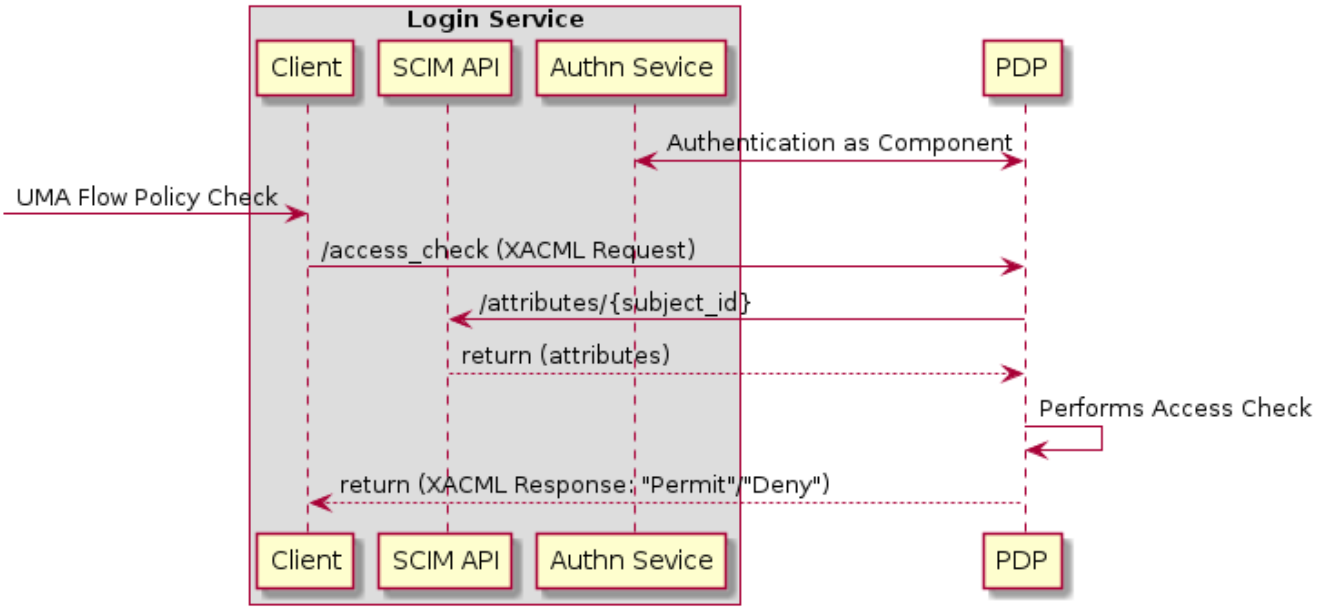

Figure 13. Authorization Server XACML Request Scenario

3.3.5.2. Policy Management

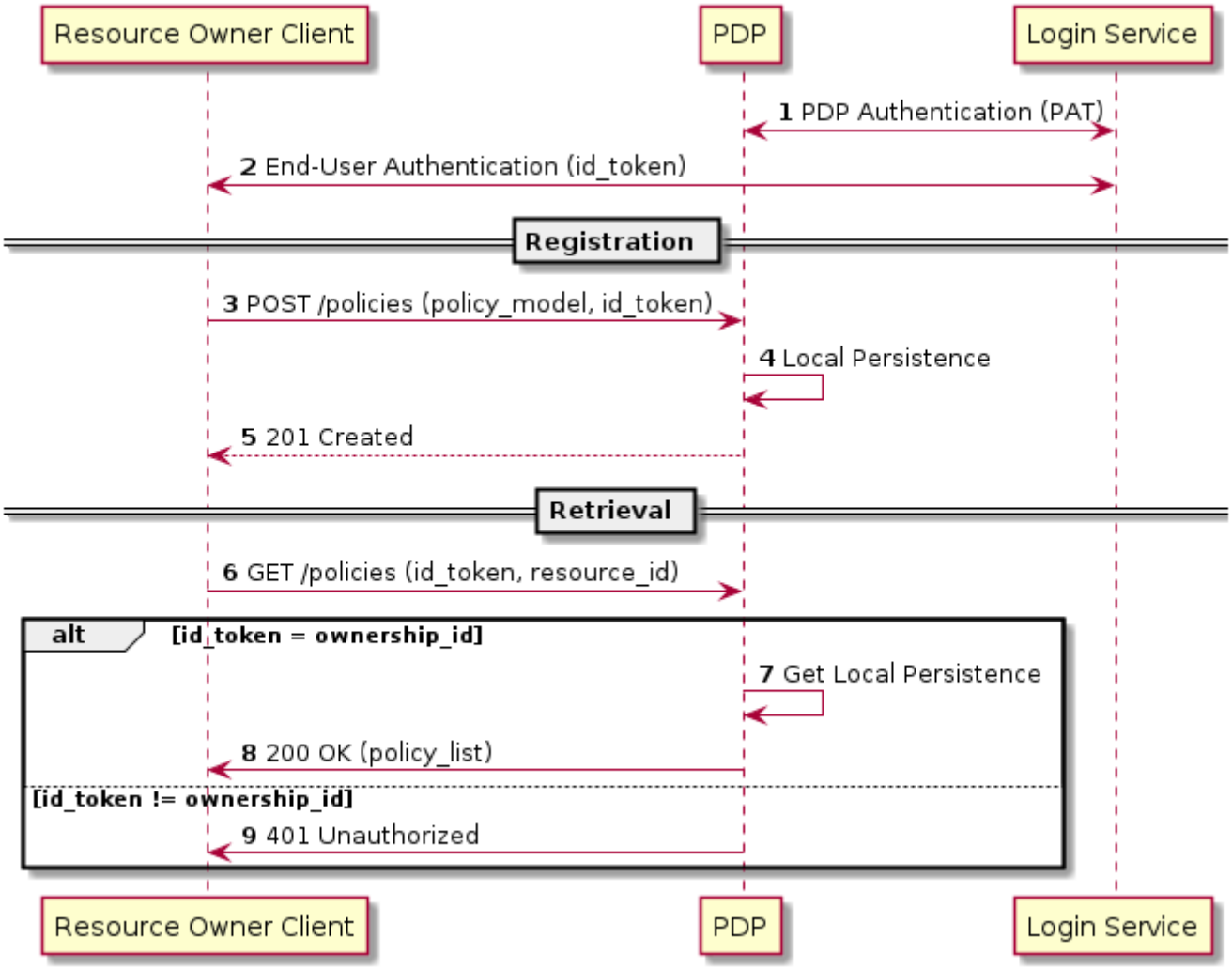

Figure 14. PDP Policy Management Scenarios

3.4. User Profile

Refer to User Profile building block documentation - https://eoepca.github.io/um-user-profile/master

3.4.1. Description

The User Profile building block serves to encapsulate profile actions (such as edit or removal) into a web interface, while at the same time providing the infrastructure upon which to implement other building blocks, such as Billing and Licensing.

3.4.2. Context

The following context diagram identifies the major components interfacing with the User Profile: (RED ~ consumers of User Profile provided interfaces, BLUE ~ providers of interfaces consumed by the User Profile)

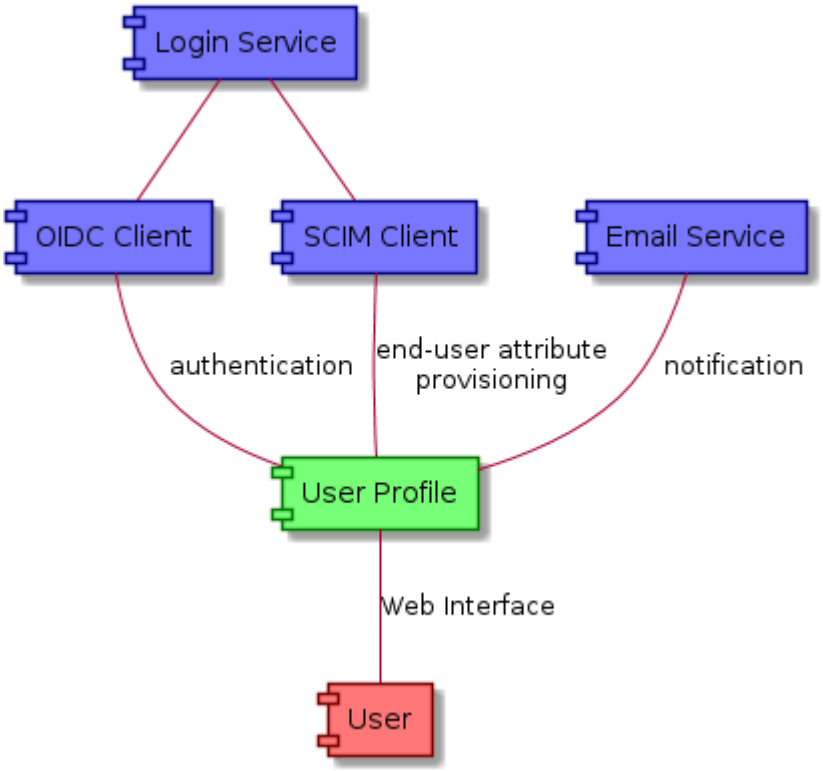

Figure 15. Policy Decision Point context diagram

In order to support the Billing Service, the User Profile building block allows to identify users (leaving a reference to their home IdP), to assign them Billing Identities, Service API keys, License keys and to record Terms and Conditions acceptance. It's a persistence service with interfaces that will be queried by other building blocks (License Manager, Billing Service, Policy Decision Point) and modified by both the License Manager and the Login Service (during creation of a new user profile or assignment of new Licenses).

3.4.3. Provided Interfaces

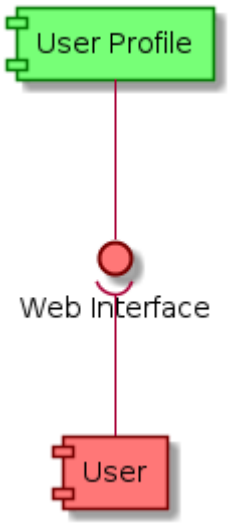

Figure 16. Policy Decision Point provided interfaces

3.4.3.1. Profile Management Web Interface

A web service is made available for users to perform actions related to the building block, such as account removal.

3.4.3.1.1. Applicable Standards

No applicable standards besides Hypertext Transfer Protocol (HTTP) as means to serve the information to the End-User

3.4.3.1.2. Endpoints

URL: <um-login-service>/web_ui

3.4.4. Required Interfaces

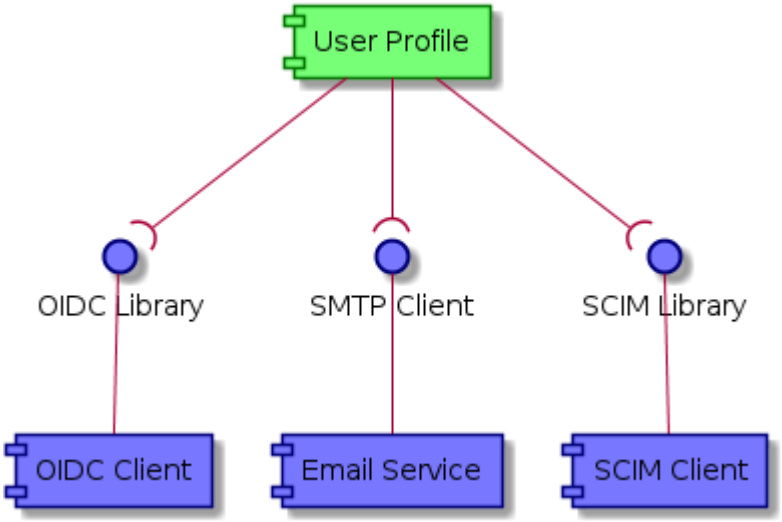

Figure 17. Policy Decision Point required interfaces

3.4.4.1. OIDC Authentication

The User Profile utilizes an OIDC Client implementation to consume OIDC endpoints, authenticate

itself as trusted platform component, generate JWT documents containing End-User information, and secure its Platform Resource API endpoints.

3.4.4.1.1. Applicable Standards

- IETF - OpenID Connect 1.0

3.4.4.1.2. Remote Endpoints

URL: <um-login-service>/.well-known/openid-configuration

The Login Service exposes the standard discovery document, a JSON showcasing all the necessary metadata and endpoint enumeration for a client application to dynamically load all the needed endpoints.

3.4.4.2. SCIM Identity Management

The User profile utilizes a SCIM Client implementation to consume SCIM endpoints, designed to make managing user identities in cloud-based applications and services easier due to its well-defined schemas and endpoints.

3.4.4.2.1. Applicable Standards

- IETF RFC 7644 - System for Cross-domain Identity Management

3.4.4.2.2. Remote Endpoints

URL: /.well-known/scim-configuration

This well-known endpoint allows to discover all relevant SCIM operations. Although this strategy is not enforced, serving a well-known endpoint makes client integration more familiar to developers that may be used to OIDC Client integration.

3.4.4.3. Email Service

The User Profile building block also implements a SMTP Client, that allows for the sending of email notifications that serves as, and implements, a confirmation action in an account removal scenario.

3.4.4.3.1. Applicable Standards

- IETF RFC 2822 - Internet Message Format

3.4.4.3.2. Remote Endpoints

URL: <um-login-service>/confirmation_mail

3.4.5. Example Scenarios

3.4.5.1. Attribute Edition

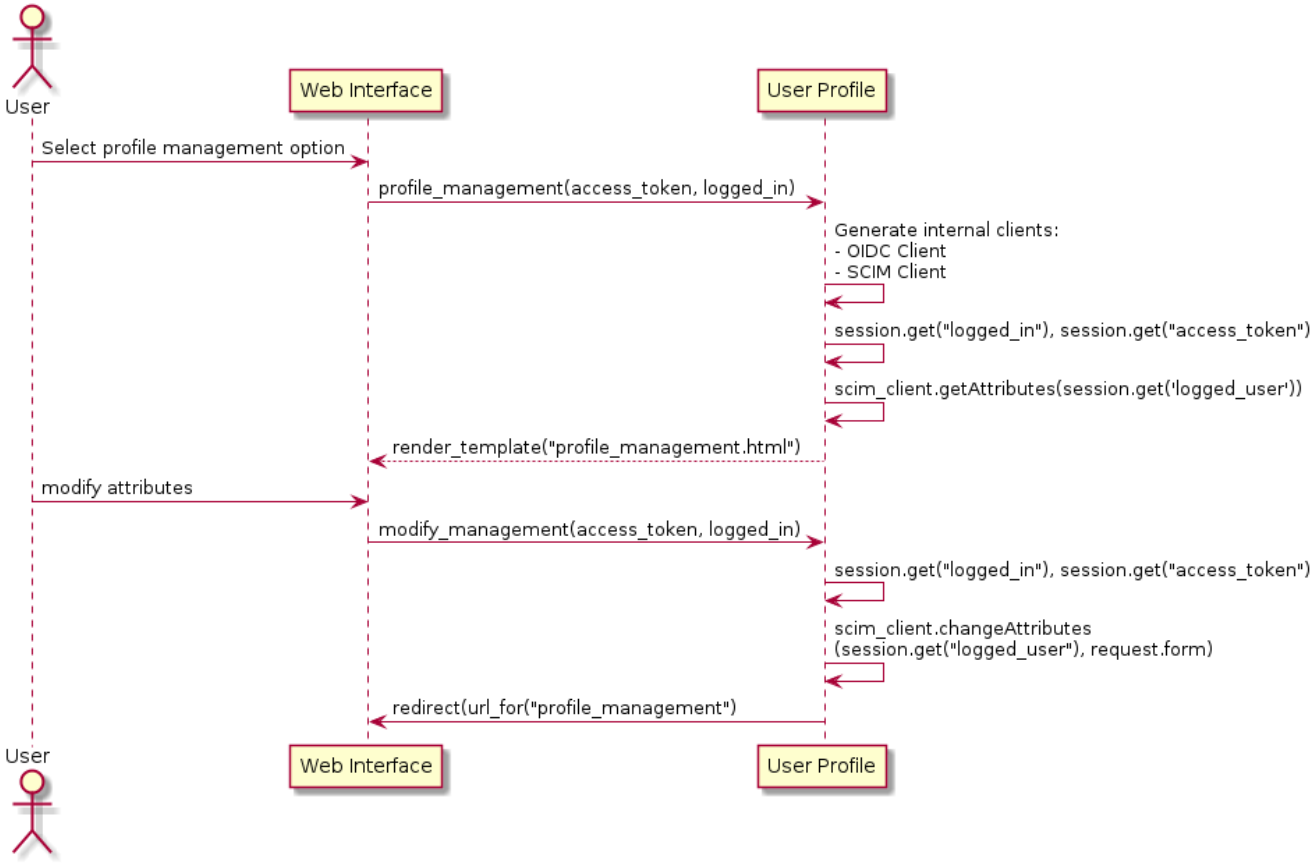

Figure 18. User Profile attribute editing

3.4.5.2. Account Deletion

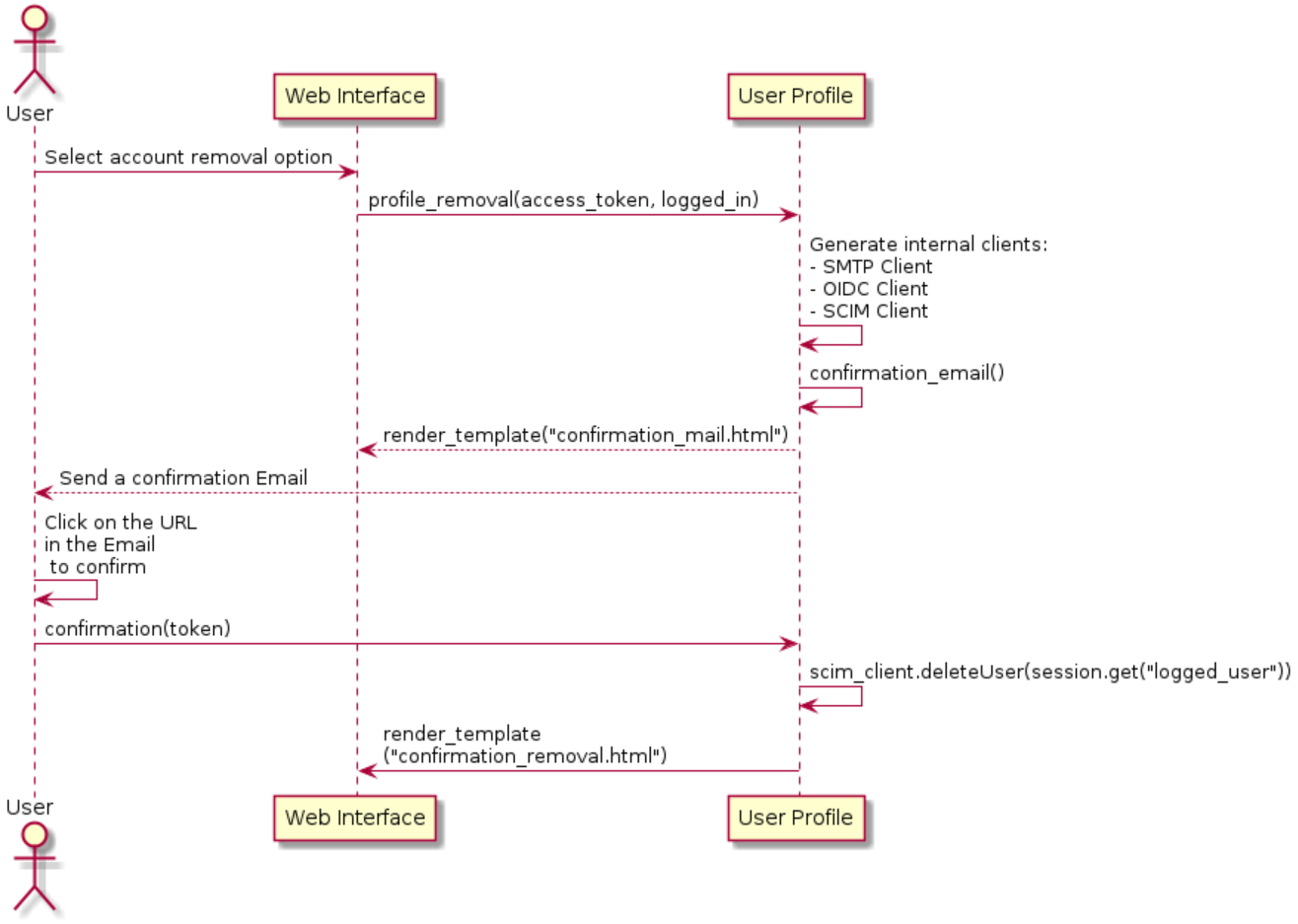

Figure 19. User Profile account removal

3.5. License Manager

Not in the scope of Release 0.2

3.6. Billing Service

Not in the scope of Release 0.2

3.7. Pricing Engine

Not in the scope of Release 0.2

Chapter 4. Processing & Chaining

4.1. Application Deployment and Execution Service (ADES)

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

4.1.1. Description

The ADES provides processing services interfaces via WPS 1.0 & 2.0 service ports 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.

4.1.2. Context

The following context diagram identifies the major components interfacing with the ADES:
(RED ~ consumers of ADES provided interfaces, BLUE ~ providers of interfaces consumed by the ADES)

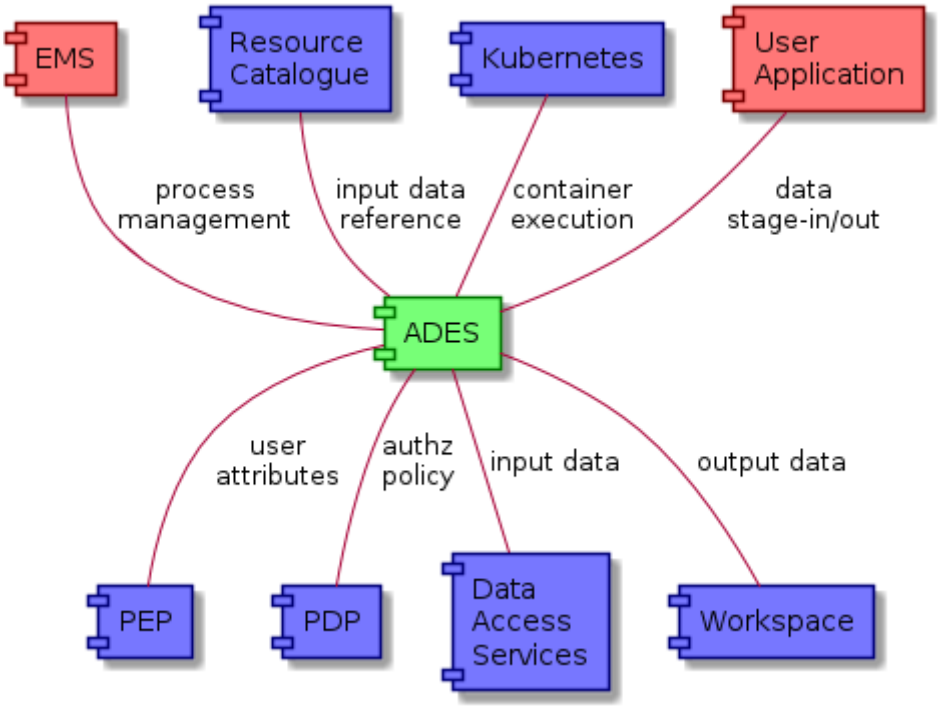

Figure 20. ADES context diagram

The EOEPCA **Execution Management Service (EMS)** is the main client of the ADES that (un)deploys applications, executes & monitor jobs.

All interactions with the ADES should be via the **Policy Enforcement Point (PEP)** that filters authorized requests, and provides user attributes to the ADES, allowing the user's context to be propagated through onward requests. The ADES obtains authorization policy information, as required, from the policy endpoint of the **Policy Decision Point (PDP)**.

The ADES resolves input data references via the **Resource Catalogue**, with the data retrieved through the **Data Access Services** referenced by the catalogue. The ADES interfaces with the ***User Application*** to orchestrate its execution (incl. data stage-in/out). The ADES interfaces to the Kubernetes cluster for execution of the requested application. Processing outputs are persisted to the **Workspace**.

4.1.3. Provided Interfaces

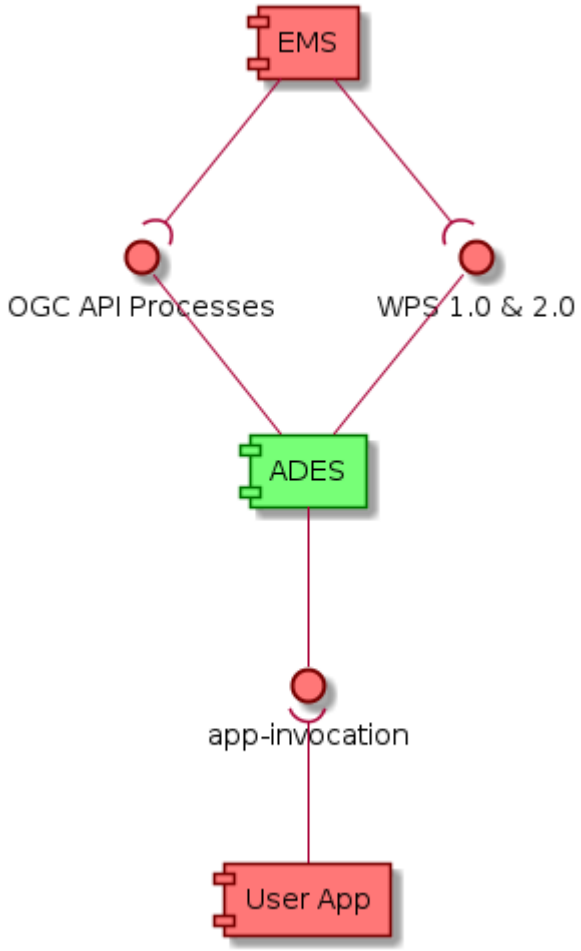

Figure 21. ADES provided interfaces

The ADES enables interfaces for the processing based on the OGC "Web Processing Service". It exposes 2 web services:

- Web services for processing:
  - WPS 1.0.0 & 2.0.0 - OGC Web Services Common [OGC-COMMON] hosting WPS 1.0.0 & 2.0.0 processing services
  - OGC API Processes - OGC API Processes [OGC-API-PROC] hosting WPS in RESTful core OpenAPI
- Application Invocation - the ADES establishes standard conventions with the application being invoked, through which the application receives its inputs and provides its outputs

4.1.3.1. 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.

4.1.3.1.1. Applicable Standards

- OGC Web Processing Service 1.0 - [OGC-WPS]
- OGC Web Processing Service 2.0 - [OGC-WPS2]

4.1.3.1.2. Endpoints

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

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. GET, POST operations according to the payload of the requested service, with bindings defined for HTTP/POST+XML and HTTP/GET+KVP.

Standard Operations

Examples with HTTP/GET+KVP binding…

GetCapabilities

`http://hostname:port/path?service=WPS&request=GetCapabilities`
Allows a client to retrieve service metadata, basic process offerings, and the available processes present on a WPS server

DescribeProcess

`http://hostname:port/path?service=WPS&version=2.0.0&request=DescribeProcess&identifier=ALL`
Allows WPS clients to query detailed process descriptions for the process offerings

GetStatus

`http://hostname:port/path?service=WPS&version=2.0.0&request=GetStatus&jobid=FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66`
Allows WPS clients to query the status of an asynchronously executed job

GetResult

`http://hostname:port/path?service=WPS&version=2.0.0&request=GetResult&jobid=FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66`
Allows WPS clients to query the result of a finished processing job

Only HTTP/POST+XML binding for execute…

Execute

`http://hostname:port/path?service=WPS` + XML POST data
Allows WPS clients to run a specified process implemented by a server

EOEPCA Extension Operations

DeployProcess

`Execute` operation with a process identifier of `eoepcaadesdeployprocess`
Allows a client to deploy a custom process to be made available via the WPS service

UndeployProcess

`Execute` operation with a process identifier of `eoepcaadesundeployprocess`
Allows a client to undeploy a previously deployed custom WPS process

The 2 previously described processes implement the transactional function of the WPS known at WPS-T in WPS 1.0 & 2.0. In the OGC API Processes described in the next section, those functions are embedded as HTTP methods (POST & DELETE) at process level.

4.1.3.2. 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.

4.1.3.2.1. Applicable Standards

- OGC API - Processes ~ [OGC-API-PROC]

4.1.3.2.2. Endpoints

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

Standard Operations

Table 2. ADES OGC API Processes

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
`/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

Resource	Method	Description	Operation
/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

Transactional Extension

Table 3. ADES OGC API Processes (transactional extension)

Resource	Method	Description	Operation
/processes	POST	Allows a client to deploy a custom process to be made available via the Processing service	DeployProcess
/processes/{id}	DELETE	Allows a client to undeploy a previously deployed custom process	UndeployProcess

4.1.3.3. Application Invocation

4.1.3.3.1. CWL as Application Descriptor

CWL is used as the application package descriptor. It covers the following elements necessary to describe the application:

1. Workflow directed acyclic graph orchestrating the steps in order mapping workflow input/output with steps input/output
2. Steps describing a command line with their input/output
3. 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.

4.1.3.3.2. Application Deployment to Web Processing Service

The ADES offers an entry point to deploy the application package (see previous sections).

Regardless of the deployment mechanism, the CWL is used to convey the information to describe the web service inputs/outputs using WPS concepts.

The mapping between CWL and WPS follows the rules described in the following sections.

CWL Workflow definition level

The workflow level is the entry point to the workflow and thus the interface used to map the inputs/outputs.
The inputs/outputs description defined in the command line or workflow steps are not taken into account directly since they are inked between steps or bound at workflow level as foreseen by the CWL specification.
Indeed, at the web processing service level, only the “staged” input and output parameters are

exposed.

4.1.3.3.3. CWL staged (in/out) files

`File` or `Directory` are CWL parameters types corresponding to files on a file system and thus require a specific mapping according to the implementation of the software component exposing the WPS. Indeed, the software component executing the processes must manage physical files in the processing environment. At job submission, physical files may be a reference to a resource manager and the software component fetches them to make them available for processing as specified in the CWL using File or Directory type.

Therefore, the following mapping rules integrate the resource reference management capabilities. This is the type(s) of reference to a physical resource that the software component manages. For instance: HTTP link, S3 link, opensearch URL, STAC catalog, etc.

The interface for the execution process must have a STAC local catalog (catalog.json and all downstream files: collection.json, items and assets including their associated files) referencing all the data staged-in corresponding collections as specified initially in the CWL.

In the above example, the CWL describes a File in input. The parameter is extended with stac:catalog that defines the collection in which the “staged” data must be referenced to.
A detailed example is described in section CWL data staging.

- *CWL File mandatory parameter example **

In this example, we assume the software component is able to handle the following resource description document or references:

- Opensearch description document describing possible search endpoints to find input files
- Atom feed with entries containing the input files (enclosures)
- STAC catalogue with features describing assets as input files

Table 4. CWL File mandatory parameter example

Resource
CWL

```
file_parameter:
      label: Title of file_parameter
      doc: Abstract describing file_parameter
      type: File
      stac:catalog:
         stac:collection: input_file
         stac:href: catalog.json
```

WPS 1.0

Resource

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>int_parameter</ows:Identifier>
  <ows:Title>Title of int_parameter </ows:Title>
  <ows:Abstract>Abstract describing int_parameter</ows:Abstract>
  <ComplexData>
    <Supported>
      <Format>
        <MimeType>application/opensearchdescription+xml</MimeType>
      </Format>
      <Format>
        <MimeType>application/atom+xml</MimeType>
      </Format>
      <Format>
        <MimeType>application/geo+json; profile=stac</MimeType>
      </Format>
    </Supported>
  </ComplexData>
</Input>
```

WPS 2.0.2

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>int_parameter</ows:Identifier>
  <ows:Title>Title of int_parameter </ows:Title>
  <ows:Abstract>Abstract describing int_parameter</ows:Abstract>
  <wps:ComplexData>
    <wps:Format mimeType="application/opensearchdescription+xml" encoding="raw" />
    <wps:Format mimeType="application/atom+xml" encoding="raw"/>
    <wps:Format mimeType="application/geo+json; profile=stac-item" encoding="raw"/>
  </wps:ComplexData>
</Input>
```

OGC API Processes (JSON)

Resource

```
{
  "inputs": [
    {
      "id": "int_parameter",
      "title": "Title of int_parameter",
      "description": "Abstract describing int_parameter",
      "input": {
        "formats": [
          {
            "Type": "application/opensearchdescription+xml"
          },
          {
            "Type": "application/atom+xml"
          },
          {
            "Type": "application/geo+json; profile=stac-item"
          }
        ],
        "minOccurs": 0,
        "maxOccurs": 1
      }
    }
  ]
}
```

CWL data referencing mechanism manages File and Directory data types

Table 5. CWL Data staging

catalog.json

```
{
  "stac_version": "1.0.0-beta.1",
  "id": "my_workflow",
  "description": "Catalog of staged data for execution 45325-34214124 of my_workflow
",
  "links": [
    { "rel": "self", "href": "catalog.json" },
            { "rel": "root", "href": "catalog.json" },
    { "rel": "child", "href": "collection-input_file.json" }
  ]
}
```

collection-input_file.json

```
{
  "stac_version": "1.0.0-beta.1",
  "id": "my_workflow",
  "description": "Catalog of staged data for execution 45325-34214124 of my_workflow
",
  "links": [
    { "rel": "self", "href": "catalog.json" },
            { "rel": "root", "href": "catalog.json" },
    { "rel": "child", "href": "collection-input_file.json" }
  ]
}
```

The complete example at the end of this section describes in details how data are staged in and STAC catalogs generated.

4.1.4. Required Interfaces

The ADES communicates to other components through the interfaces described in this section.

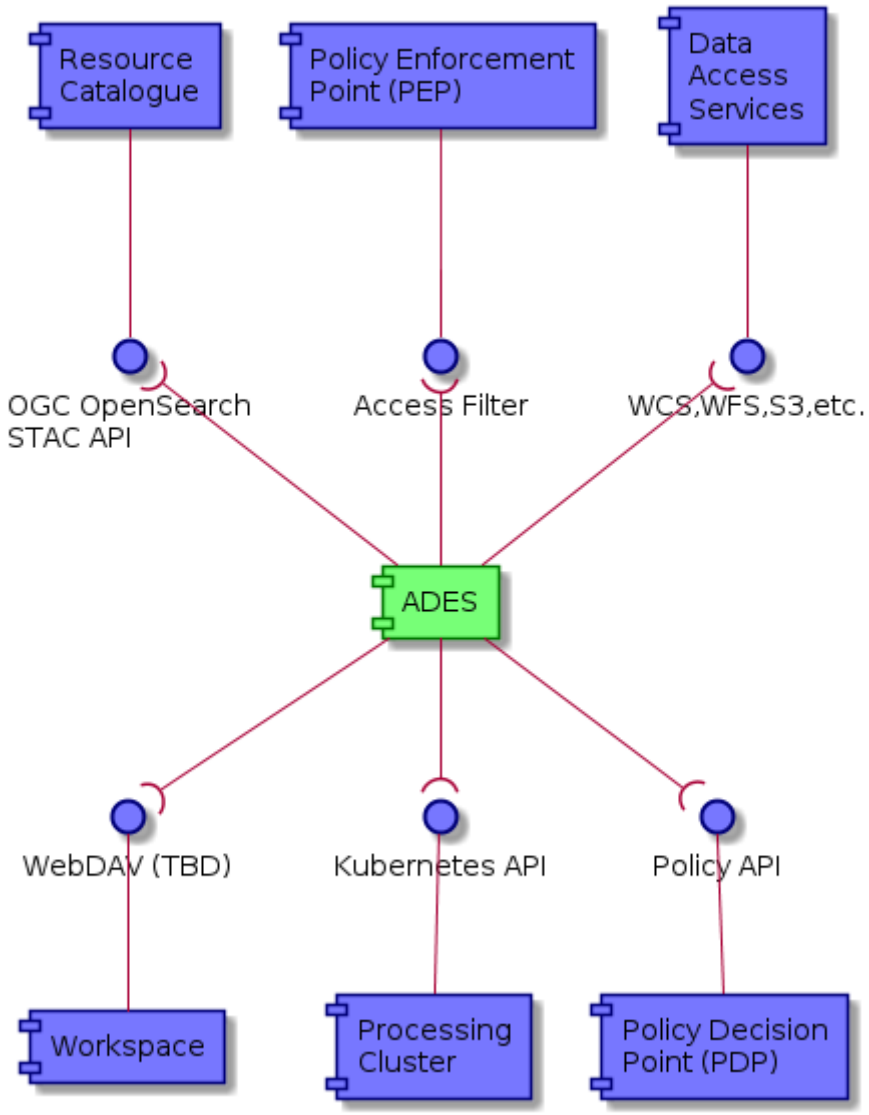

Figure 22. ADES Required (consumed) Interfaces

4.1.4.1. Resource Managers

4.1.4.1.1. Catalogue for referenced data

Since all data (resources) for processing are passed (in or out) by reference, the ADES must be able to contact those resources' manager (e.g. catalog).

In input, the ADES supports both OpenSearch ([CEOS-OS-BP]), [STAC-SPEC] and [STAC API] as resources protocols.

In output, the ADES supports the transactional functions of the [STAC API] to publish catalogs.

4.1.4.1.2. Data Access Services for data stage-in

The referenced assets in the catalogs (see previous section) are accessible from the same or another resource manager. Their enclosure URL allows to download entirely the asset or to access parts or it (e.g. COGs with HTTP range).
The ADES supports the following protocols:
- HTTP download
- S3 Download

- WebDAV upload

4.1.4.1.3. Cluster infrastructure for processing execution

The ADES uses a kubernestes API for deploying its services and to provision resources for processing.

4.1.5. Example Scenarios

This section provides examples of the ADES interfacing flows.

4.1.5.1. ADES Processing Flows

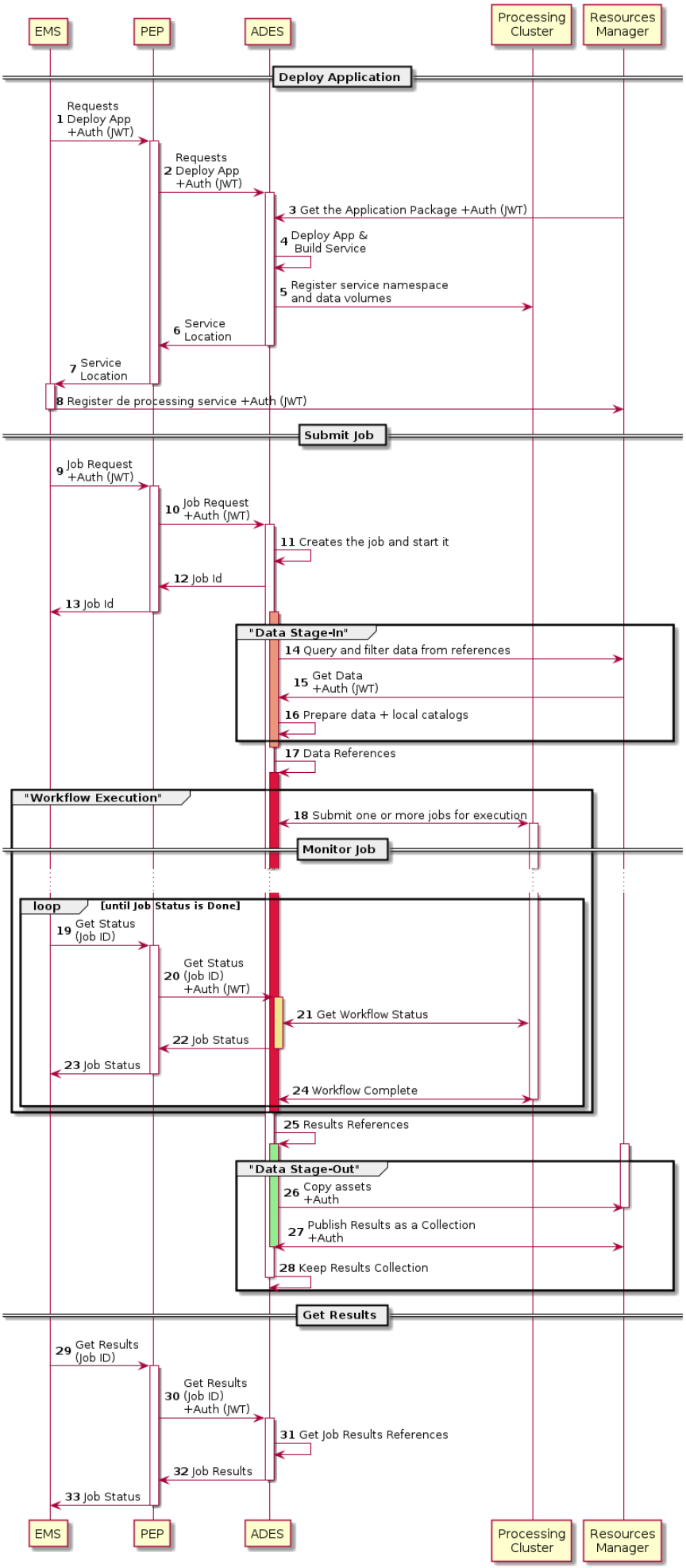
EMS
PEP
ADES
Processing Cluster
Resources Manager
Deploy Application
1 Requests Deploy App +Auth (JWT)
2 Requests Deploy App +Auth (JWT)
3 Get the Application Package +Auth (JWT)
4 Deploy App & Build Service
5 Register service namespace and data volumes
6 Service Location
7 Service Location
8 Register de processing service +Auth (JWT)
Submit Job
9 Job Request +Auth (JWT)
10 Job Request +Auth (JWT)
11 Creates the job and start it
12 Job Id
13 Job Id
"Data Stage-In"
14 Query and filter data from references
15 Get Data +Auth (JWT)
16 Prepare data + local catalogs
17 Data References
"Workflow Execution"
18 Submit one or more jobs for execution
Monitor Job
loop [until Job Status is Done]
19 Get Status (Job ID)
20 Get Status (Job ID) +Auth (JWT)
21 Get Workflow Status
22 Job Status
23 Job Status
24 Workflow Complete
25 Results References
"Data Stage-Out"
26 Copy assets +Auth
27 Publish Results as a Collection +Auth
28 Keep Results Collection
Get Results
29 Get Results (Job ID)
30 Get Results (Job ID) +Auth (JWT)
31 Get Job Results References
32 Job Results
33 Job Status
EMS
PEP
ADES
Processing Cluster
Resources Manager

Figure 23. ADES Processing Flows

Deploy Application

The first sequence is the application deployment. In the EOEPCA scenario, this is initiated by the EMS that uses the transactional functions of the ADES.

1. The EMS invoke a service or POST a request to the ADES with the reference to an Application Package and pass by the PEP that allows the deployment accoring to the policy
2. The request is transferred to the ADES along with an auth token
3. The ADES gets the Application Package referenced in the deployment request
4. The ADES registers internally the application as a service
5. The ADES sets a namespace for the service and data volume if necessary
6. The ADES returns the service location to the PEP
7. The PEP returns the service location to the EMS
8. The EMS registers the service deployed on the ADES on the catalog

Submit Job

The second sequence is the service invokation via a job submission.

9. The EMS makes a job request to the ADES via the PEP
10. The PEP authorizes and transfer the request to the ADES
11. The ADES instantiates the job and returns the job identifier
12. The ADES returns the job id to the PEP
13. The PEP returns the job id to the EMS

Data Stage-In

The first stage is about how the ADES prepares the data for the processing. This is a pre-processing step that must provision all the data needed for processing and referenced in the job execute request.

14. The ADES queries the data references passed as input of the job reequest to the resource manager (e.g. catalog)
15. With the results of the previous request, the ADES retrieves all the assets requested from the resource manager (e.g. data store)
16. The ADES organizes and prepare the data as nedded (e.g. unzip) and rebuild the catalog to point to locally copied data
17. The data references are passed to the next stage of the execution

Workflow Execution

Data is now ready to be processed by the main processing. The software component will ensure that the input data folder will be properly mounted on each cluster node and accessible from

the container that executes the processing. This is the main stage where the processing is actually executed on the data.

18. The ADES submits and coordinates the necessary jobs as defined in the application workflow

Monitor Job

This sequence in a parallel sequence triggered by the EMS each time it requests for the job status during the asynchronous job execution

19. EMS makes a request for status with the Job Id previously returned (13) via the PEP
20. PEP passes the requests along
21. The ADES builds a job progress report based on the workflow execution status
22. The ADES returns the job status document via the PEP
23. The PEP returns the job status to the EMS
24. Once the job is completed, the job status reports the job results location
25. The results refertences are passed to next stage

Data Stage-Out

The last stage is about how the ADES publishes the data after the processing. This is a post-processing step that must copy the selected data to a persistent storage and registers theit reference to a catalog for further searches.

26. The ADES copies the results' assets to the persistent storage of the resource manager (e.g. WebDAV)
27. Publish a collection of the assets produced with the reference to the persistent storage to the catalog
28. Keep in the cache of the ADES a reference to the collection for further results query

Get Results

This ending sequence is the results report from the ADES

29. EMS makes a request for results with the Job Id previously returned (13) via the PEP
30. PEP passes the requests along
31. The ADES builds a results report based on the results collection (28)
32. The ADES returns the job results document via the PEP
33. The PEP returns the job results to the EMS

4.2. Execution Management Service (EMS)

TBD

4.3. Workflow Engine

TBD

4.4. Processor Development Environment (PDE)

TBD

4.5. Interactive Analysis Tool (IAT)

TBD

Chapter 5. Resource Management

5.1. Resource Catalogue

TBD

5.2. Data Access Services

TBD

5.3. Data Access Gateway

TBD

5.4. Data Access Library

TBD

5.5. Data Ingestion

TBD

5.6. Workspace

TBD

<< End of Document >>