



TELESPAZIO

a LEONARDO and THALES company

<Template Service> Design Document

EOEPCA.SDD.xxx

TVUK System Team

Version 0.1, dd/mm/yyyy:

<Template Service> Design 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 | 4 |
| 1.5. Glossary | 9 |
| 2. Overview | 11 |
| 2.1. Building Block Overview | 11 |
| 2.2. External Interfaces | 11 |
| 2.2.1. HTTP(S) (Reverse Proxy Listener) | 11 |
| 2.2.2. XACML (from/to Login Service) | 12 |
| 2.2.3. Resource API (to Policy Enforcement Point) | 12 |
| 2.2.4. OIDC (to Login Service) | 12 |
| 2.2.5. SCIM (to Login Service) | 12 |
| 2.2.6. MongoDB as Database | 12 |
| 2.3. Required resources | 12 |
| 2.3.1. Software | 12 |
| 2.4. Static Architecture | 13 |
| 2.5. Use cases | 13 |
| 2.5.1. Registration of policies | 14 |
| 2.5.2. Policy Retrieval | 14 |
| 2.5.3. Access Check | 14 |
| 3. Building Block Design | 15 |
| 3.1. Policy Decision Engine | 15 |
| 3.1.1. Overview and Purpose | 15 |
| 3.1.2. Software Reuse and Dependencies | 15 |
| 3.1.3. Interfaces | 16 |
| 3.1.4. Data | 16 |
| 3.1.4.1. Configuration | 16 |
| 3.1.4.2. Data flow | 16 |
| 3.1.5. Extensibility | 17 |
| 3.1.6. Applicable Resources | 17 |
| 3.2. POLICY RESPOSITORY | 17 |
| 3.2.1. Overview and Purpose | 18 |
| 3.2.2. Software Reuse and Dependencies | 18 |
| 3.2.3. Data flow | 18 |
| 3.2.4. Applicable Resources | 18 |

| | |
|---|---|
| COMMENTS and ISSUES 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/template-svce/issues . | PDF 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 | Initial in-progress draft |

Chapter 1. Introduction

1.1. Purpose and Scope

This document presents the <Template Service> Design for the Common Architecture.

1.2. Structure of the Document

Section 2 - [Overview](#)

Provides an over of the <Template Service> component, within the context of the wider Common Architecture design.

Section 3 - [\[mainDesign\]](#)

Provides the design of the <Template Service> 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-UC] | EOEPCA - Use Case Analysis EOEPCA.TN.005 https://eoezca.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 |
| [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 |
| [WPS-REST-JSON] | 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 |
| [CWL] | Common Workflow Language Specifications, https://www.commonwl.org/v1.0/ | v1.0.2 |

| Reference | Document Details | Version |
|-------------------|--|-------------------------|
| [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 |
| [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 |

| Reference | Document Details | Version |
|------------|--|---------------------------|
| [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 |
| [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 |

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 <i>Products</i> produced as output of an <i>Interactive Application</i> analysis session. |

| Term | Meaning |
|---|--|
| Analytics | A set of activities aimed to discover, interpret and communicate meaningful patterns within the data. Analytics considered here are performed manually (or in a semi-automatic way) on-line with the aid of <i>Interactive Applications</i> . |
| Application Artefact | The 'software' component that provides the execution unit of the <i>Application Package</i> . |
| 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. |
| Application Descriptor | A file that provides the metadata part of the <i>Application Package</i> . 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 <i>Application Descriptor</i> and the <i>Application Artefact</i> . |
| Bulk Processing | Execution of a <i>Processing Service</i> 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 (interpreted) 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 <i>Processing Service</i> or an <i>Interactive Application</i> . |

| Term | Meaning |
|------------------------------------|--|
| 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). |
| 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 <i>Analysis Results</i> 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 <i>Interactive Application</i> 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. |

| Term | Meaning |
|------------------------------|---|
| 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 <i>Processing Service</i> whose execution is initiated directly by the user on an ad-hoc basis. |
| 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 <i>Products</i> produced as output of a <i>Processing Service</i> execution. |
| Processing Service | A non-interactive data processing that has a well-defined set of input data types, input parameterisation, producing <i>Processing Results</i> with a well-defined output data type. |
| Products | EO data (commercial and non-commercial) and Value-added products and made available through the EP. <i>It is assumed that the Hosting Environment for the EP makes available an existing supply of EO Data</i> |
| 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. |

| Term | Meaning |
|--|--|
| 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 <i>Processing Service</i> 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. |
| 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-dimensional 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 |
| 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 |

| Term | Definition |
|-------------|---|
| 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 |
| 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 EOEPKA

The main functionality of the PDP is to be able to perform complex Policy Decisions based on Policy Documents. In order to do so, several functionality blocks are identified:

- Policy Check Endpoint. A XACML-compliant endpoint that allows to submit JSON XACML requests and receive the corresponding responses.
 - SCIM Client, allowing to retrieve user information that is local to the platform, whenever possible.
 - OIDC Client, allowing to authenticate the component as trusted within the architecture.
 - Resource API Client, allowing to retrieve details about the resources being accessed.
- Policy Repository Management. An exposed API allowing to register and/or query policies assigned to specific resources.

The PDP, when started, follows these steps:

1. Load the configuration from the config file
2. Insert policies using Policy_Storage class
3. Start reverse proxy using Flask and await requests

When launched, the PDP will answer to all requests that start with the configured path. The request has a json that follows an xacml format where access will be requested for a resource.

2.2. External Interfaces

2.2.1. HTTP(S) (Reverse Proxy Listener)

An HTTP listener, which can be configured through the config file.

This is the only input interface to interact directly with the PDP from outside, and is managed by the reverse proxy.

The default listener for this interface is `0.0.0.0:5567/policy/validate`.

This interface will parse accept requests to get a decision about the access to a policy.

2.2.2. XACML (from/to Login Service)

The PDP exposes a policy check endpoint compliant with XACML 3.1 standards, with a content-type application/json due to its JSON Profile.

2.2.3. Resource API (to Policy Enforcement Point)

The PDP queries the corresponding PEP to retrieve information about a resource.

2.2.4. OIDC (to Login Service)

The PDP uses the OIDC protocol in order to authenticate itself as a valid UMA client, and uses this OIDC client in all UMA-related queries.

These queries are done against the Login Service, and the endpoints used are:

- TOKEN_ENDPOINT

2.2.5. SCIM (to Login Service)

The PDP has the capability to auto-register itself as a client if there is no client pre-configured from previous starts or previous configuration.

In order to do this, it utilizes the SCIM protocol, and queries the Login Service.

The endpoints used for SCIM are:

- AUTH_CLIENT_POST

2.2.6. MongoDB as Database

In order to access the policies, these policies will be stored in a non-relational database which will be MongoDB

2.3. Required resources



Content Description

This section contains:

- List of HW and SW required resources for the correct functioning of the building Block
- References to open repositories (when applicable)

2.3.1. Software

The following Open-Source Software is required to support the deployment and integration of the Policy Enforcement Point:

- EOEPKA's SCIM Client - <https://github.com/EOEPKA/um-common-scim-client>

- EOEPKA's Well Known Handler - <https://github.com/EOEPKA/well-known-handler>
- Flask - <https://github.com/pallets/flask>
- MongoDB for python - <https://pymongo.readthedocs.io/en/stable/index.html>

2.4. Static Architecture

Content Description

This section contains:



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

The Policy Decision Point at this moment is composed of only one component: The reverse proxy.

The next section [Design] contains detailed descriptions and references needed to understand the intricacies of this component.

2.5. Use cases

Content Description

This section contains:



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



2.5.1. Registration of policies

The process of registering the policies is performed in the main, for this purpose the Policy_Storage class is used which will allow actions such as inserting policies in the pod where the Mongo database is located

2.5.2. Policy Retrieval

(Represented in the above graph by the request to Login Service called "Get the policies related to the resources ids")

The PDP access through the resource id that was extracted from the XACML in the request and using the Policy_Storage class, allows to access MongoDB and extracts all the policies information stored for that resource id

2.5.3. Access Check

(Represented in the above graph by the request to Login Service called "Validate policies with values extracted from XACML")

When the PDP has obtained the policies, we proceed to compare the content of these policies with the values obtained from the request, mainly they are the resource id and the user_name, in case that some of these do not coincide, it will be response with "Deny" in the json or with a "Permit" if everything is correct

Chapter 3. Building Block 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 general 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. Policy Decision Engine

3.1.1. Overview and Purpose

The Flask-based endpoint allows to query and retrieve XACML-compliant requests.

3.1.2. Software Reuse and Dependencies

All requirements for the executing of the reverse proxy are found under `src/requirements.txt`, and expect Python 3.6.9 or greater to work.

The most important are:

- **EOEPCA-SCIM**: Used as a complementary measure to the XACML passing of client assertions.
- **EOEPCA-OIDC**: Used to auto-register itself as a client to the Auth. Server upon start-up.
- **WellKnownHandler**: Used to dynamically check the configuration of the Authorization Server on each execution. For example, it can get the needed endpoints for any API the PEP needs, such as the token request for OIDC.
- **Flask**: External to EOEPCA's project, this library allows the PDP to expose its endpoints.
- **MongoDB**: Used to storage the policies for every resource, with the possibility of performing actions such as insert policies, modify, delete, etc

3.1.3. Interfaces

This component doesn't have any internal interfaces. For a reference of external interfaces see [External Interfaces](#) on Section 2 [Overview](#)

3.1.4. Data

3.1.4.1. Configuration

The PDP gets all its configuration from the file located under config/config.json.

The parameters that are accepted, and their meaning, are as follows:

- **prefix:** "/path"-formatted string to indicate where the reverse proxy should listen. Default is "/pdp"
- **host:** Host for the proxy to listen on. For example, "0.0.0.0" will listen on all interfaces
- **port:** Port for the proxy to listen on. By default, 5567. Keep in mind you will have to edit the docker file and/or kubernetes yaml file in order for all the prot forwarding to work.
- **check_ssl_certs:** Toggle on/off (bool) to check certificates in all requests. This should be forced to True in a production environment
- **debug_mode:** Toggle on/off (bool) a debug mode of Flask. In a production environment, this should be false.

3.1.4.2. Data flow

The only information the PDP handles are XACML json format from the request, and policies provided by MongoDB.

What follows is an example of the nominal flow for the PDP:



3.1.5. Extensibility

The design of the PDP allows for further improvements if need be. For example:

- The parser script can be expanded to parse all the configuration for a XACML formatted.
- The registration of policies can be expanded to insert the policies without hardcoded values like resource id
- The proxy can be expanded to parse further parameters on top of the HTTP protocol, allowing for any kind of plugin or complex mechanism desired.

3.1.6. Applicable Resources

- XACML 3.0 JSON Profile Specification 1.1 - <http://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html>
- EOEPKA's SCIM Client - <https://github.com/EOEPKA/um-common-scim-client>
- EOEPKA's OIDC Client - <https://github.com/EOEPKA/um-common-uma-client>
- EOEPKA's Well Known Handler - <https://github.com/EOEPKA/well-known-handler>
- Flask - <https://github.com/pallets/flask>

3.2. POLICY RESPOSITORY

3.2.1. Overview and Purpose

It is the database based on MongoDB where the policies are stored and queried for the PDP purposes

Included with the PDP there is a script at the source path that performs queries against a Mongo Database. The main purpose of this script is to reduce the usage of RAM when registering a policy locally and when querying for its content.

It is developed to generate a database called 'policy_db' in case it does not exist. The collection used for the storage of the documents is called 'policies'.

The script defines the following methods:

- **insert_policy**: Will generate a document with the policy data received as input if it already exists it will update it. The main parameters of the policy would be an auto-generated id provided by mongo which identifies each document in the database, the resource_id provided by the login-service, the name for the policy, and the configuration which will be the policy with its resource_id. This would be mandatory parameters in order to perform other kind of queries.
- **update_policy**: Updates the content of a policy stored matched by its ID
- **get_policy_from_resource_id**: Finds the policy, attached to a resource by a resource_id given. Returns a list of policies in json format to the resource_id associated.
- **get_policy_from_id**: Matches the policy by its unique ID
- **delete_policy**: This function will receive a policy id and will find and delete the matched document

This script is manipulated by the API which would intercept the request in order to perform PUT, POST and DELETE methods.

In the future this will be compliant with other databases in order to unify the methods mentioned above.

3.2.2. Software Reuse and Dependencies

The script needs the library 'pymongo', to query the database. It will expect a MongoDB service running on the default's Mongo port 27017

In order to simulate database responses for unit tests, it is needed to use the python mock library.

This class would be called from APIs within the PDP repository. At the moment the usage is mainly for the policy interaction purposes, but this can easily be reused by other agents.

3.2.3. Data flow

The database will only be accessible by the API or the Proxy.

The main methods for the interaction with the database are displayed in this dataflow as a summary of its scope:

3.2.4. Applicable Resources

- MongoDB image from DockerHub - https://hub.docker.com/_/mongo

<< End of Document >>