User Profile Design Document

EOEPCA.SDD.xxx

TVUK System Team

Version 0.1, 20/05/2020:

User Profile Design Document

EO Exploitation Platform Common Architecture

User Profile Design Document

EOEPCA.SDD.xxx

COMMENTS and ISSUES	**PDF**
If you would like to raise comments or issues on this document, please do so by raising an Issue at the following URL https://github.com/EOEPCA/um-user-profile/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	Initial in-progress draft

Chapter 1. Introduction

1.1. Purpose and Scope

This document presents the User Profile Design for the Common Architecture.

1.2. Structure of the Document

Section 2 - Overview

Provides an over of the User Profile component, within the context of the wider Common Architecture design.

Section 3 - Building Block Design

Provides the design of the User Profile 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://eoepca.github.io/use-case-analysis	Issue 1.0, 02/08/2019
[EP-FM]	Exploitation Platform - Functional Model, ESA-EOPSDP-TN-17-050	Issue 1.0, 30/11/2017
[TEP-OA]	Thematic Exploitation Platform Open Architecture, EMSS-EOPS-TN-17-002	Issue 1, 12/12/2017
[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 *Products* produced as output of an *Interactive Application* analysis session.

Term	Meaning
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.
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*.

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 *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.

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 *Processing Service* 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 *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.

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 *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.
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
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

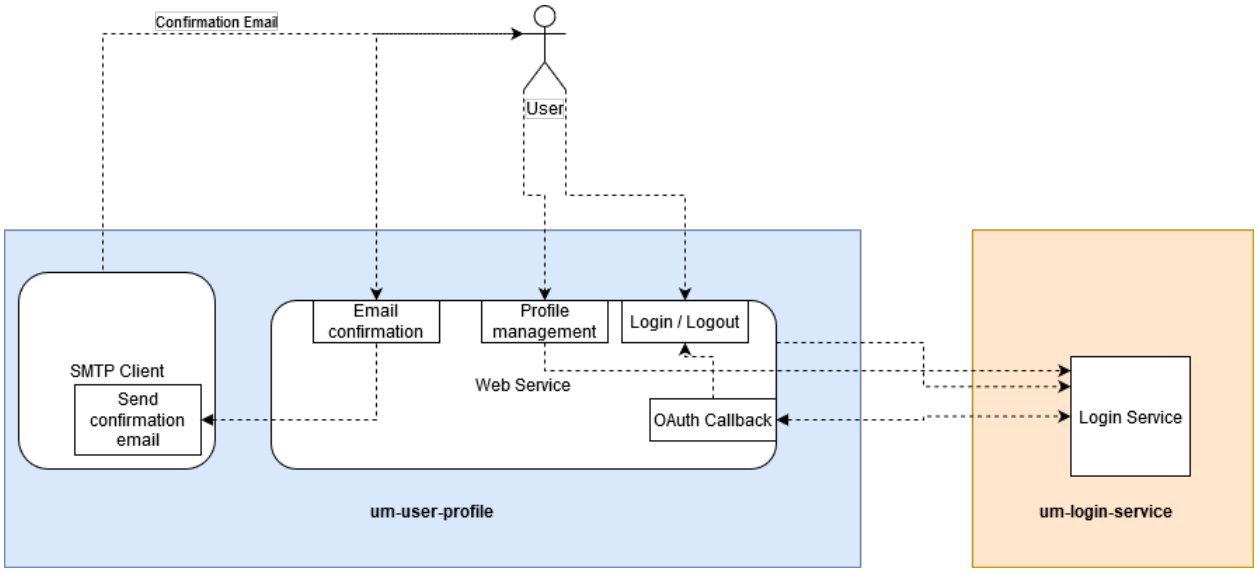
Content Description

This section contains:

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

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).

The figure below, identifies the main workflows on which the User Profile participates, along with it's components:



Based on this workflows, the User Profile enables the following functionality aspects

2.1.1. OIDC Client in User Profile

The User Profile will mainly use the OIDC Client functionality in the User Profile Portal in order to provide Client authentication and End-User authentication functionality according to the OpenIDConnect Authentication Code Flow.

This functionality is provided by a custom script to manage the oauth standard and uses the python WellKnownHandler library that allows for simple parsing and usage of an SSO server's "well-known" endpoints.

2.1.2. Send confirmation Email

This functionality allows you to send an email and notify the user that the account will be deleted. In addition, by clicking on the link in the email the user will confirm the action. This functionality is provided by smtp standards for gluu and custom scripts.

The code within the SMTP is dedicated to prepare both the url to proceed with the account removal and the html that will be in the body of the email.
With this it will make the whole process of creating the SMPT Server connection, generating the message and finally sends the email. Previously we need to have configured the mail that will send the email, choosing the mail (Google Account) and the password to access, which are in the Dockerfile as a container variable.

2.1.3. Account removal

The functionality related to the account removal is implemented thanks to a SCIM Client that allow querying for user information, update User Profiles based on the assignment of a specific license or in this case a account removal previously created and the two previously mentioned functionalities.

2.1.4. API Keys List

It's a feature that allows the user to modify their own list of API keys. It is implemented using the SCIM Client that allows to perform attribute modifications and checks against the Login Service OIDC Auth.

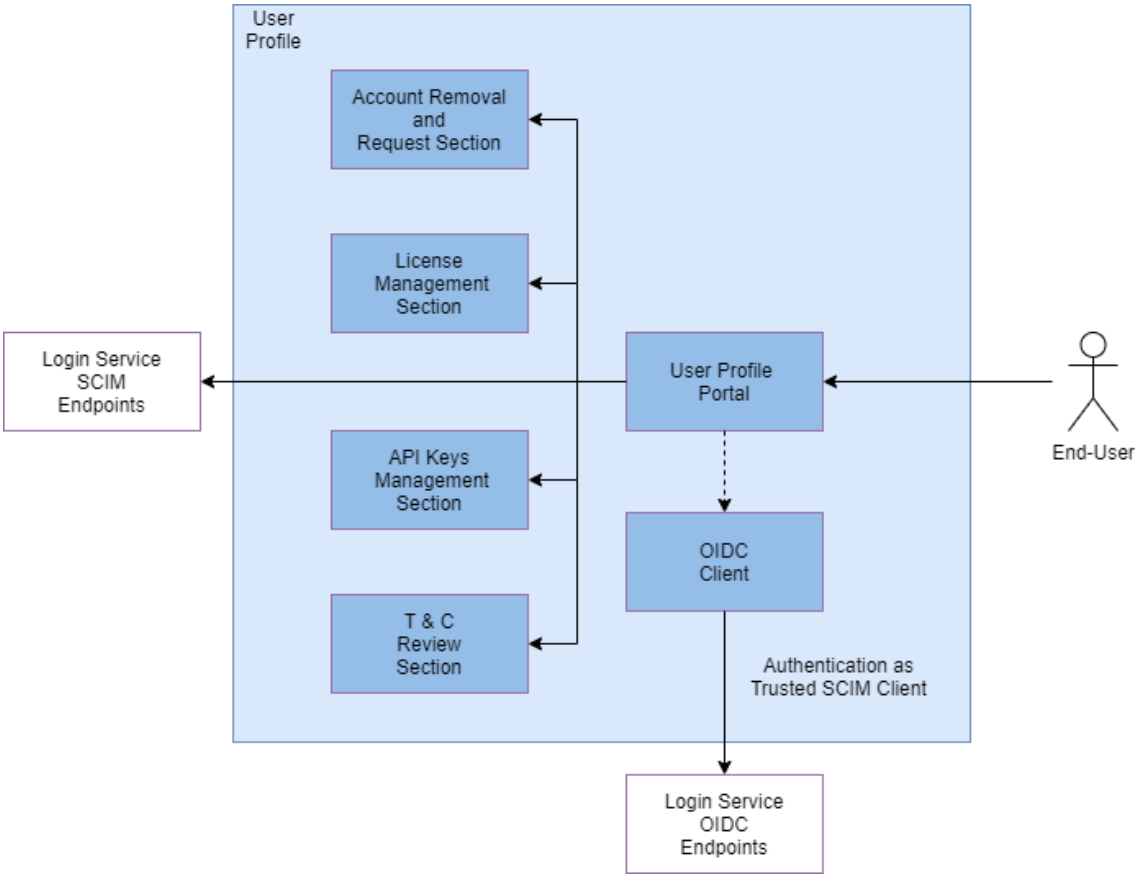
2.2. Static Architecture

Content Description

This section contains:

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

The User Profile, heavily relies on Free and Open Source Software that already implements the interfaces and functionality required. The following image depicts the Static Architecture of this Building Block.



- The Authentication and Authorization Service enables both OIDC and UMA flows for the whole Platform
- The User Profile Portal with SCIM Implementation, allowing direct interaction with the End-User Back-End, in this case, the account removal
- A connection to an SMTP Email server for sending confirmation emails, using at the moment a Google Account.

The Section for the Building Block Design [Design] contains detailed descriptions and references to the Open Source components used in this Building Block.

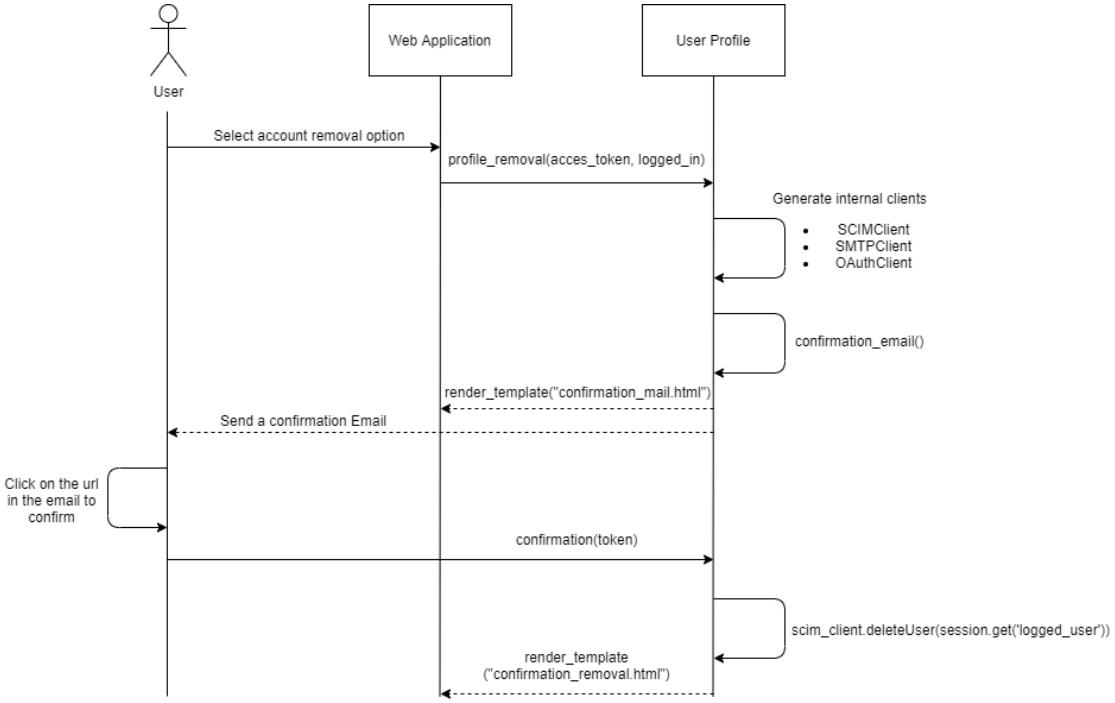
2.3. Use Cases

Content Description

This section contains:

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

- Use Case: Account Removal



- Principal actor: User
- Preconditions:
  - The user must be signed in to perform the action
- Postconditions:
  - The user's account will be removed
- Main successful scenario:
  - The user logs in to the platform
  - The user selects the option to remove account
  - The user will receive a confirmation email
  - When the user clicks on the url in the email will confirm the action of account removal

2.4. External Interfaces

Content Description

This section contains:

- Listing of technical external interfaces (with other Building Blocks)

2.4.1. um-login-service Interface

- Identity Management Interface: The User Profile uses a System for Cross Domain Identity (SCIM) Interface through a .well-known URI that describes all standard endpoints provided by um-login-service building block.
- Authentication (AuthN) Interface: The User Profile uses an OpenID Connect interface through a .well-known URI that describes all standard endpoints provided by um-login-service building block.

2.4.2. Web Interface

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

2.4.3. SMTP Interface

The User Profile building block uses SMTP client in order to implement all the functionality related to sending confirmation emails.

- How to configure the account for sending emails
  - If you want to select the email you want to use to send these emails you must update the Dockerfile, changing the values of the variables EMAIL_ADRESS for the account you want to use and EMAIL_PASSWORD for the account password. Note that the value for EMAIL_PASSWORD should be an app password to make it easy and avoid problems (https://support.google.com/accounts/answer/185833?hl=en)

2.5. 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)

2.5.1. Software

The following Open-Source Software is required to support the deployment and integration of the Login Service:

- Authentication and Authorization Service
  - oxAuth - Gluu Inc. (https://github.com/GluuFederation/oxAuth)
- SSO server's "well-known" endpoints
  - Python library: WellKnownHandler (https://pypi.org/project/WellKnownHandler/)
- Email Service
  - SMTP Client python library: smtplib (https://docs.python.org/3/library/smtplib.html)
  - Email settings library: email (https://docs.python.org/3/library/email.html#module-email)
- Deployment, Configuration and Integration Tooling
  - Kubernetes secret and config Tooling (https://github.com/kubernetes/kubernetes)

Chapter 3. Building Block Design

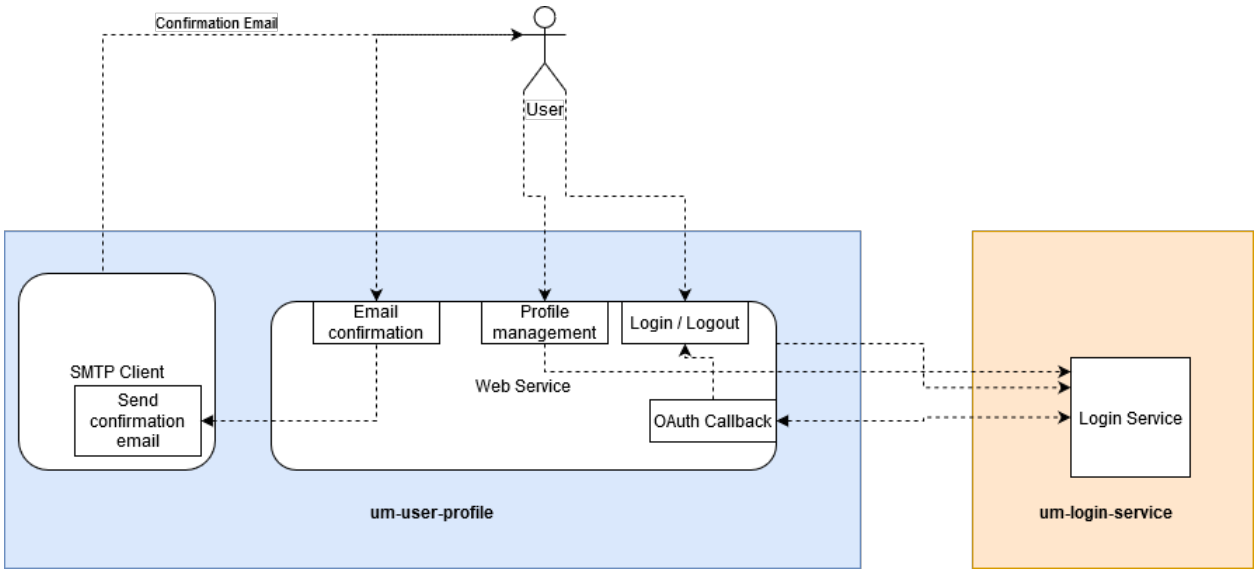
Content Description

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. General UML design



3.2. Web Service

3.2.1. Overview and Purpose

The web service is the user interface for all other components on this module.

This same web service holds an interface for user attributes edition and profile removal for a user.

3.2.2. Software Reuse and Dependencies

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

The most important are:

- **EOEPCA-SCIM**: Used to auto-register itself as a client to the Auth. Server upon startup
- **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 Web Service needs, like the SCIM endpoints.
- **Flask**: External to EOEPCA's project, this library allows the Web Service to handle HTTP requests, to act as a web server.

3.2.3. Interfaces

3.2.3.1. Oauth Callback

A callback for all oauth actions (login/logout), configurable via the `WEB_confg.json` file in case the deployment of the module is unitary, otherwise it will retrieve the environtment variables within the system deployment.
This endpoint manages the information given by the auth server, returned from the Login Service, and allows users to log into the app through the SSO's sessions.

3.2.3.2. Email Confirmation

`<base_uri> + /confirmation_mail`

A callback endpoint for all confirmation emails. It recieves a code generated for the email, and verifies the action that the user wanted to take.

Currently it only handles account removal, but this can be expanded in the future since the token checking procedure is very generic.

3.2.4. Data

3.2.4.1. Configuration

The Web service gets all its configuration from the file located under `src/config/WEB_config.json`.

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

- **sso_url**: hostname or IP of the Auth Server.
- **title**: Title that will be seen when navigating to the web interface
- **scopes**: Scopes used for the internal OAUTH client. Currently, the required are: "openid", "email" and "user_name"
- **client_id**: Client ID used for the internal OAUTH client.
- **client_secret**: Client secret for the corresponding client_id.

- **redirect_uri**: Redirect URI configured in the client, which should point to this service's callback URL.
- **post_logout_redirect_uri**: Redirect URI for post logout of a user.
- **base_uri**: Base URI for all requests against the web server
- **oauth_callback_path**: Callback path for the end of a succesful oauth flow.
- **logout_endpoint**: Endpoint for the logout of a currently logged in user.
- **service_host**: Host to listen on (localhost, 0.0.0.0, etc..)
- **service_port**: Port to listen on for the web server
- **protected_attributes**: Attributes that the user can see about their profiles, but not edit
- **blacklist_attributes**: Attributes that the user can not see or edit.
- **separator_ui_attributes**: Separator used for multi-level attributes
- **color_web_background**: Color used for the background of the web, in HEX.
- **color_web_header**: Color used for the header of the web, in HEX.
- **logo_alt_name**: Alternative name for logo of the web
- **logo_image_path**: Path to logo of the web
- **color_header_table**: Color used for the header of any table
- **color_text_header_table**: Color used for the content of any table
- **color_button_modify**: Color used for the modify button
- **use_threads**: Toggle threads for requests. Enabling this in production is recommended
- **debug_mode**: Toggle debug mode, which enables a debug web interface, more errors and logs.

3.2.5. Applicable Resources

- EOEPCA's SCIM Client - https://github.com/EOEPCA/um-common-scim-client
- EOEPCA's Well Known Handler - https://github.com/EOEPCA/well-known-handler
- Flask - https://github.com/pallets/flask

3.3. SMTP Client

3.3.1. Overview and Purpose

The SMTP Client incorporated with this module serves as a basic mean of sending emails to the user. Currently, the only email configured to be sent is the registration confirmation, but this can be expanded in the future.

3.3.2. Software Reuse and Dependencies

- **smtplib** + **email**: Basic python libraries which provide a communication layer to an SMTP server

3.3.3. Interfaces

No interfaces are provided for this component, since it's integrated as part of the code of the web, as a python client.

3.3.4. Data

3.3.4.1. Configuration

The following configuration is extracted from ENV variables upon start:

- **EMAIL_ADRESS**: Email used to send the email
- **EMAIL_PASSWORD**: Password for the respective email address.

The following are constants hardcoded in code, but easily changable, and will be variables in the future:

- **host**: SMTP server to comunicate to. Default is 'smtp.gmail.com'
- **port**: SMTP port to connect to. Default is '465'

Additionally, the client will work with the system's trusted CA certificates, in order to provide certification validation and all other SSL-related capabilites.

3.3.4.2. Data flow

The SMTP Client doesn't use any data supplied by the user, and thus no data flow is needed. Everything it handles is composed of random tokens, which it sends to the user's email (obtained during login).

3.3.5. Applicable Resources

- smtplib - https://docs.python.org/3/library/smtplib.html

Chapter 4. Use Case Traceability

Content Description

This section contains:

- A traceability matrix against the use case analysis document of the project, indicating which components address each use case

<< End of Document >>