# Key cloak API Specifications Document

**For**

**OJAS HIRING MANAGEMENT SYSTEM**

### Ver no. 1.0

Disclaimer: This document is classified as 'internal' and used by Ojas Innovative Technologies Pvt. Ltd. employees only until and unless it is specified otherwise.

***Document Revision History***

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver. No.** | **Date**  **(dd-Mmm-yyyy)** | **Name of the Author** | **Change Information** |
| 0.1 | 17-sep-2024 | Nagarjuna Reddy | Initial draft version |
| 1.0 | 17-sep-2024 | Nagarjuna Reddy | Baselined after receiving approval |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver. No.** | **Date**  **(dd-Mmm-yyyy)** | **Name of the Reviewer** | **Name of the Approver** |
| 0.1 | 17-sep-2024 | Venkata Ravindranath Gundugola | Venkata Ravindranath Gundugola |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Template Revision History (For SEPG only)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver. No.** | **Date**  **(dd-Mmm-yyyy)** | **Name of Author** | **Change Information** |
| 0.1 | 30-Nov-2021 | Vanima Prasad | Initial draft copy |
| 1.0 | 24-Dec-2021 | Vanima Prasad | Baselined after Receiving Approval |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver. No.** | **Date**  **(dd-Mmm-yyyy)** | **Name of the Reviewer** | **Name of the Approver** |
| 0.1 | 15-Dec-2021 | Srinivas Bollapragada | Srinivas Bollapragada |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Table of Contents***

1.0 Assets Search API Development............................................................5

2.0 Low Level Design Document 5

3.0 Objective 5

4.0 Process workflow 6

5.0 Sequence Diagram 7

6.0 API's Specifications.................................................................................. 8

API intro.................................................................................................8

Assumptions...........................................................................................8

API Details..............................................................................................8

Request.................................................................................................8

Response...............................................................................................8

7.0 Security Considerations..........................................................................8

8.0 conclusion..............................................................................................9

### 1.0 Asset Search API Overview

### The API in the OHAMS project allows authorized users to interact with Key-cloak, where clients—typically applications or services—authenticate users and obtain tokens. These clients enable a single sign-on experience and allow access to other services using tokens issued by Key-cloak. Some clients can also operate independently, obtaining tokens for self-authenticated access to services.

### 2.0 Low-Level Design

* ****Key cloak Server:****  The identity and access management (IAM) system.
* ****OHAMS Application****: The application that needs to authenticate users.
* ****Clients**:** Represent the services or applications (OHAMS in this case) that interact with Key cloak.
* ****User**:** An entity that needs to be authenticated to access the OHAMS application.
* ****Token (Access Token, Refresh Token)**:** Tokens issued by Key cloak to provide access to various services.

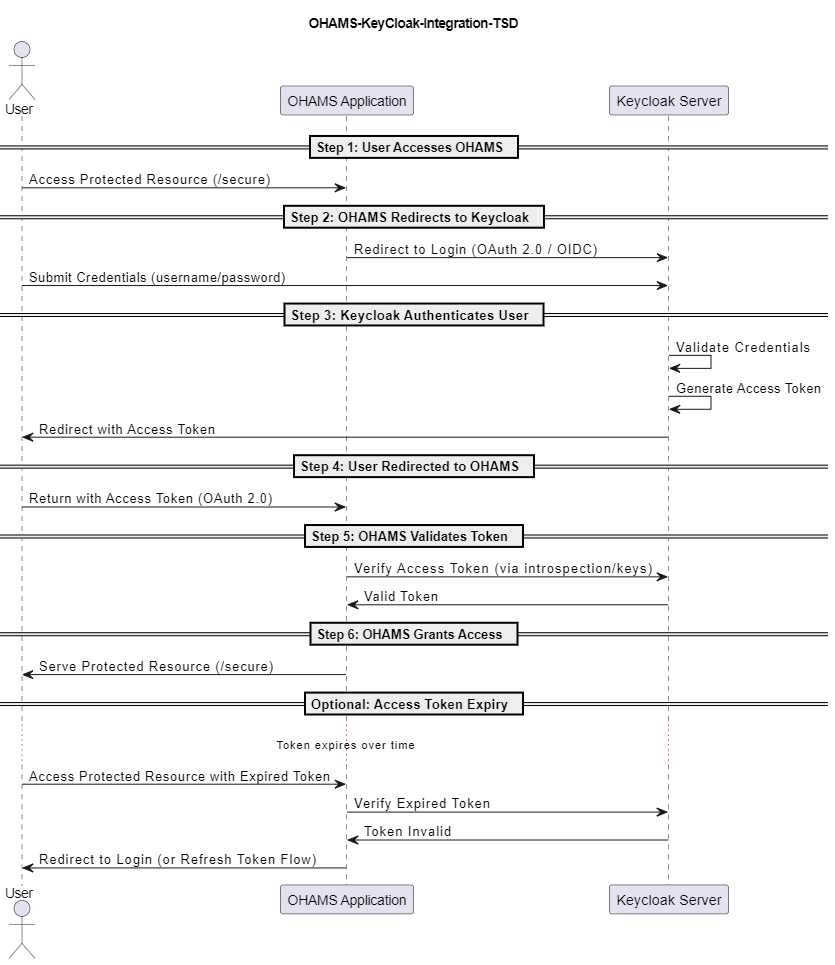
### 3.0 Objective

* **Secure User Login**: Provide safe login for users with tokens generated by Key cloak.
* **Token-Based Access**: Use tokens to let authorized users access specific services securely.
* **Single Sign-On (SSO)**: Allow users to log in once and access multiple services without repeated logins.
* **Manage User Roles**: Control user access based on their role (e.g., Admin, Hiring Manager) using Key cloak.

### 4.0Process Workflow

### Keycloak-Flowchart

5.0 Sequence Diagram:



### 6.0 API Specifications

API Introduction:

The POST API (Application Programming Interface) allows different software applications to communicate by defining the methods and data formats for sending data to a server, enabling integration and interaction between systems.

Assumptions:

* Key cloak can be found and obtained.
* The client's credentials are configured properly.
* For secure communication, use HTTPS.
* Permissions and roles are set up correctly.
* Token administration functions as planned.
* Key cloak's user data is precise.

API Details:

* Endpoint: /realms/multi\_tenant/protocol/openid-connect/token
* Method: POST
* Purpose: The purpose of integrating Key cloak into OHAMS is to enhance security and streamline user access through secure, token-based authentication and role-based access control.

Request:

* URL:/realms/multi\_tenant/protocol/openid-connect/token
* Headers:
  + Authorization: Bearer Token
  + Content-Type: application/json

7.0 Security Considerations

**Token Expiration:** Refresh tokens are longer-lasting than access tokens, which expire faster to lower the chance of theft.  
 **HTTPS Communication:** HTTPS is used to encrypt all data transfers between OHAMS, Key cloak,and services.  
 **Client Confidentiality:** User and front-end access to client secrets is never permitted, as they are safely kept in the OHAMS back-end.

8.0 Conclusion

Smooth access control, easier authentication, and enhanced security are all made possible with Key cloak integration in OHAMS. It strengthens overall system security by ensuring safe user access while safeguarding sensitive data and customer confidentiality through role-based access and secure communication.

1. Submit Order API

* + 1. *Main information*

|  |  |
| --- | --- |
| **Task** | The POST API for user authentication in OHAMS to provide secure login and access tokens. |
| **Business Case** | The OHAMS project requires a secure and efficient authentication mechanism to manage user access. By implementing a POST API for authentication, the project aims to:   * Offer seamless login for users. * Ensure secure access via token-based authentication. * Integrate with Key cloak for centralized management. * Enhance security with short-lived access tokens and longer-lived refresh tokens. |
| **Related documents** | * **API Specification Document**: Details endpoints, methods, and formats. * **Key cloak Integration Guide**: Covers Key cloak setup. * **Security Requirements Document**: Outlines security protocols. |
| **API Flow** | * **Login Request**: Client sends credentials to the authentication endpoint. * **Authentication**: Server validates with Key cloak. * **Token Issuance**: Key cloak returns access and refresh tokens. * **Response Handling**: Client receives and uses tokens. * **Token Renewal**: Client uses refresh token to get a new access token. |
| **Sequence Flow** | Image of the sequence flow diagram or something similar |

* + 1. *Request/Response*

|  |  |  |  |
| --- | --- | --- | --- |
| **API Name** | **Endpoint URL** | **Verb** | **Description** |
| **The API name** | The URL for this endpoint. Also specify the environments if there are more than one. | HTTP  Method | Short description of the API |
|  |  |  |  |
| **e.g. Submit Order AP**I | http://192.168.1.98:8080/realms/multi\_tenant/protocol/openid-connect/token | e.g. /POST | E.g.  The API will submit an order |
| **Headers** | Headers of the request/response Specify the authorization process.  Example:  Content-Type: application/j son Authorization: Bearer <Bearer Token> | | |
| **Request Body** | **Request sample:**  {  "grant\_type": "password",  "client\_id" : "ohams",  "username" : "ayesha",  "password" : "ayesha"  } | | |
| **Response Body** | **Response sample:**  {      "access\_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJHdGpHeEpNUnJTemRJdllpY0NKcTRiYVJFS1VKQlJxMDVsNkFpMHZvSjBVIn0..LHkBGdiikRkYMtKEzsAvQmKQdtAa17nB2flKcaRCyikaqRWFQnpKWxZ5CWAyzKctWDbjDaGxagn\_RVHYL3vrO-o9MrQ04GgStVYURh1Dw8N8wt759lovGPCmOof1lraZeXu\_4YiLNaa1XumjeK3XGPxUO\_mgoN86DNGT7ZmOayiBkmWiJG-Es2DNcD1WooSK7A4PdheguKC0l4BxOiNWOv7i9MXV8ZPdekekhXqgRWawcZ071aQhzlvmAGpUVKJCZB28TNXSEcmRohVQvygWz79NsuFP\_36iT9xop6U1ji1riGx8G131pghRLocvUOe2CVqVaiq\_8YYJQzmZCZTg0Q",      "expires\_in": 1800,      "refresh\_expires\_in": 1800,      "refresh\_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI1MDNhOTkxNy0wNzczLTRjYTQtYTA2Yy0xNzNmZDYxMDAxYjkifQ..qsy2Z38g7ZnxGcbTP5krYpLRcCxwscWYASPGxLKE3y8",      "token\_type": "Bearer",      "not-before-policy": 0,      "session\_state": "a6747404-d091-40ef-aa0d-466c50b9d00a",      "scope": "profile email"  } | | |

## Output parameters

From Example

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Parameter | Data type | Mandatory | Description | Value options |
| 1 | status | Var char string | Y | COMPLETED, IN\_PROGRESS,  ERRORED | IN\_PROGRESS |
| Response:  {  “status”:”IN\_PROGRESS”  } | | | | | |

## Response codes

|  |  |  |
| --- | --- | --- |
| Response code | Message | Meaning |
| 200 - OK |  | The request was successful |
| 400 – Bad Request | Missing argument! | Example will send this if any details are missing or there is a  mismatch with our records. |
| 401 – Authentication error | Authentication error | E.g. bad or missing token. |
| 404 – Not Found |  |  |
| 503 – Service Unavailable |  | The resource is currently busy, i.e. already being processed. |
| 500 – Internal Server Error | TBC | All other exceptions |