RE4DY
MANUFACTURING DATA NETWORKS

RE4DY TOOLKIT

Name of the Tool	Keycloak
Tool Owner	Industry Commons Foundation
Version	1.0
Date	Nov 2025
Version	V1.0

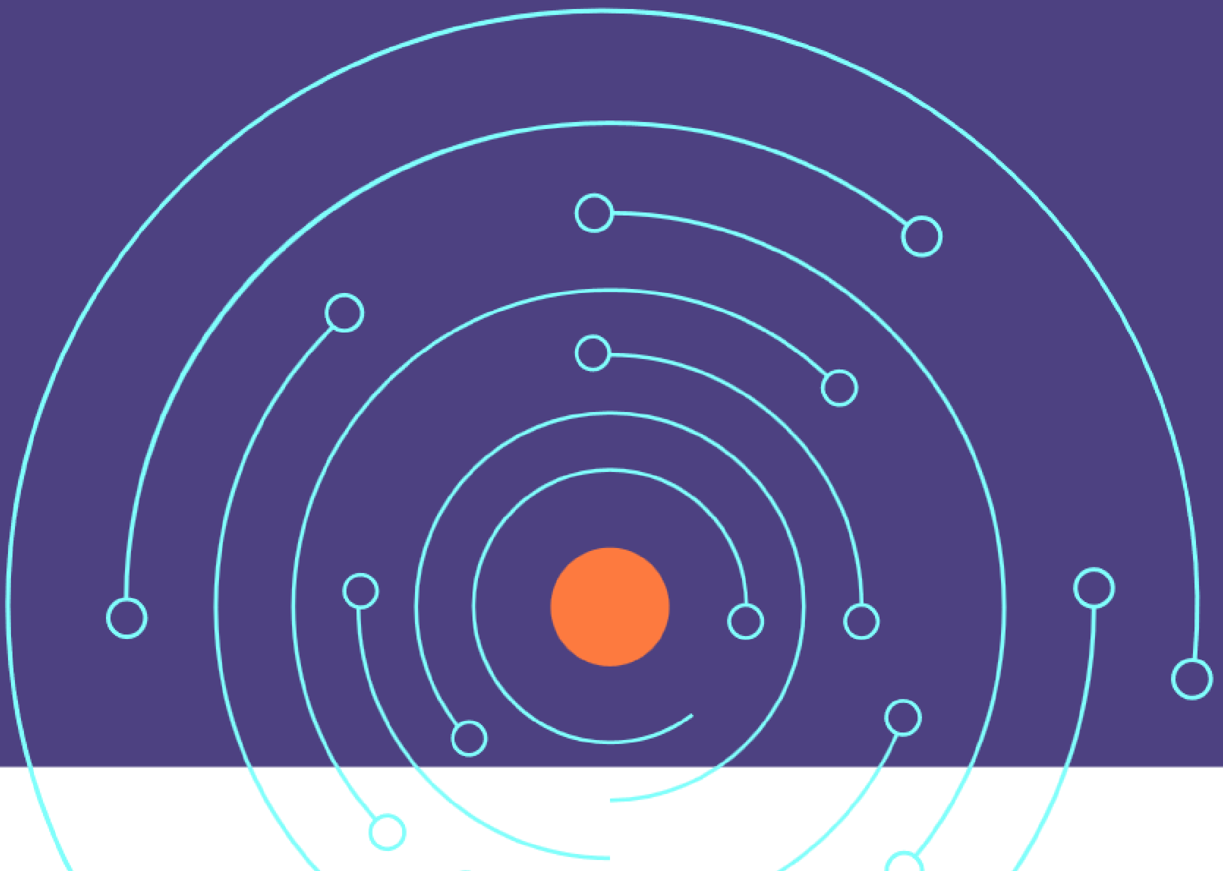

Table of contents

1. Component Description

Keycloak is an open-source identity and access management tool which plays a vital role in the authentication and authorization processes of the RE4DY project. It performs the storage and management of the RE4DY user base and provides authentication and authorization functionality in the form of libraries, APIs and user forms that enable a seamless integration with other tools. It provides Single-Sign On functionality which unifies the authentication and authorization procedures across the entire set of applications comprising RE4DY, allowing users to log in and log out once rather than performing this process for each RE4DY application individually. Moreover, the administrator may employ the Admin Console web UI to manage users and realms, as well as configure security and cryptography related settings. Finally, the Account Management Console grants users the capability to manage their account information and security.

2. Input

The Keycloak identity manager communicates with the back-end to provide the necessary authentication and authorization functionality. Therefore, it only receives input from the back-end during the registration and login phases.

Registration

During the user registration phase, the back-end is responsible for propagating attributes retrieved by the eIDAS node to Keycloak so that new users may be created. In order to create a new user, the user details that are sent to Keycloak as input are enumerated below:

- Username: Username given by the user during the registration process, after having authenticated with their eIDAS credentials. It is always the same as the user's eIDAS username.
- First Name: The user's first name, retrieved from the access token returned by the eIDAS node.
- Last Name: The user's last name, retrieved from the access token returned by the eIDAS node.

In addition, in the current implementation, four custom attributes are sent to Keycloak:

- assertion: The assertion returned by the eIDAS node. It is stored as it is received, in an encoded format.
- birthdate: The user's date of birth, retrieved from the access token returned by the eIDAS node.
- gender: The user's gender, retrieved from the access token returned by the eIDAS node.
- person_identifier: The user's national identification number, retrieved from the access token returned by the eIDAS node.

Furthermore, Keycloak may also accept a password to be set on a user account after it has been successfully created.

Login

Regarding the user login process, two methods that involve Keycloak may be employed. Keycloak natively offers an API endpoint through which users may be authenticated. This API may be used directly; However, the back-end also implements a relevant endpoint which communicates with Keycloak for this purpose, essentially acting as the middleman between the web UI and Keycloak. In both cases, Keycloak accepts the following fields as input:

- username: The user's username, as given during registration.
- password: The user's password, as given during registration.
- grant_type: "password" (constant) - It indicates that the client is requesting an access token using the user's credentials directly.
- client_id: "re4dy-login" - It denotes the application name through which authentication takes place.

3. Output

Keycloak returns standard HTTP response codes for all requests it receives, based on the outcome or result of the request processing.

Registration

Keycloak returns the "201 Created" response code to indicate that the user creation process has successfully completed.

Login

When Keycloak has determined that the given credentials given are correct and correspond to a user stored in the related realm, it returns the following data:

- access_token: An access token corresponding to the user. It may be decoded to retrieve the information regarding the account, its Keycloak roles and access to Keycloak resources, as well as user information stored during the registration process (e.g. last name).
- expires_in: The remaining number of seconds until the access token expires.
- refresh_token: The refresh token which may be used to obtain a new access token.
- refresh_expires_in: The remaining number of seconds until the refresh token expires.
- token_type: Specification of the token type being used. Usually "Bearer".
- not-before-policy: The number of seconds before which the token is not valid.
- session_state: The session identifier or the token associated with the user's session. This identifier may be used to associate the token with the user's session in Keycloak.
- scope: The resources that the client making the request is authorized to access on behalf of the user through the access token.

4. Information Flow

Register

Registering begins through the RE4DY front-end application where the user is initially presented with an eIDAS login form to insert their eIDAS credentials. The credentials are sent from the front-end to the authentication back-end service and from there they are sent to the eIDAS node for authentication.If the credentials match with an existing eIDAS user, the eIDAS node returns an access token to the authentication back-end which is then forwarded to the front-end. Next, the user is presented with a registration form where they are able to create their RE4DY account. Their filled in RE4DY credentials are sent to the back-end along with the token returned by eIDAS. The token is decoded in order to extract information such as name, surname, gender and more. Using this information, the back-end uses the Keycloak Java API to create a new user and set the account password. Finally, upon success, the front-end and subsequently the user is informed that they may login with their RE4DY account.

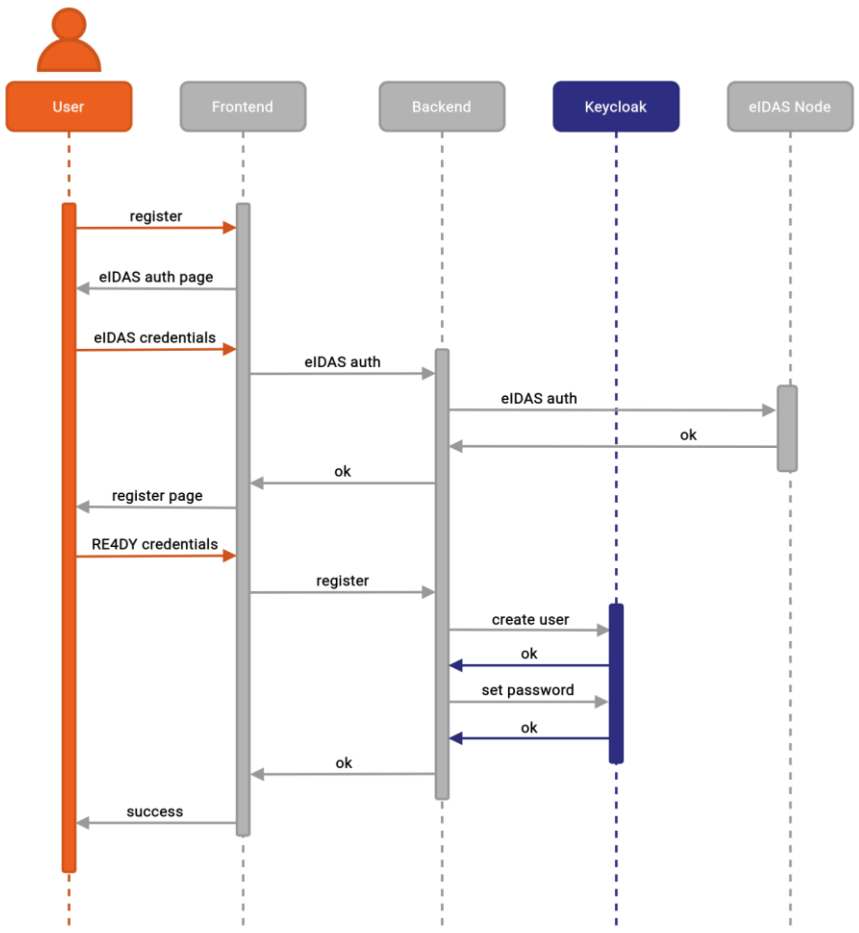

Figura 1 UML Diagram of the registration process and the communication of the front-end, authentication back-end, Keycloak and eIDAS services

Login

The login procedure commences through the RE4DY front-end application where the user is presented with a login form where they are able to insert their username and password. Upon submitting the credentials, a request is sent to the authentication back-end, which communicates with Keycloak in order to determine whether the user exists and the credentials for the given username are valid. Upon successful authentication, an access token is returned to the back-end and propagated to the front-end, which allows the user to access the RE4DY application.

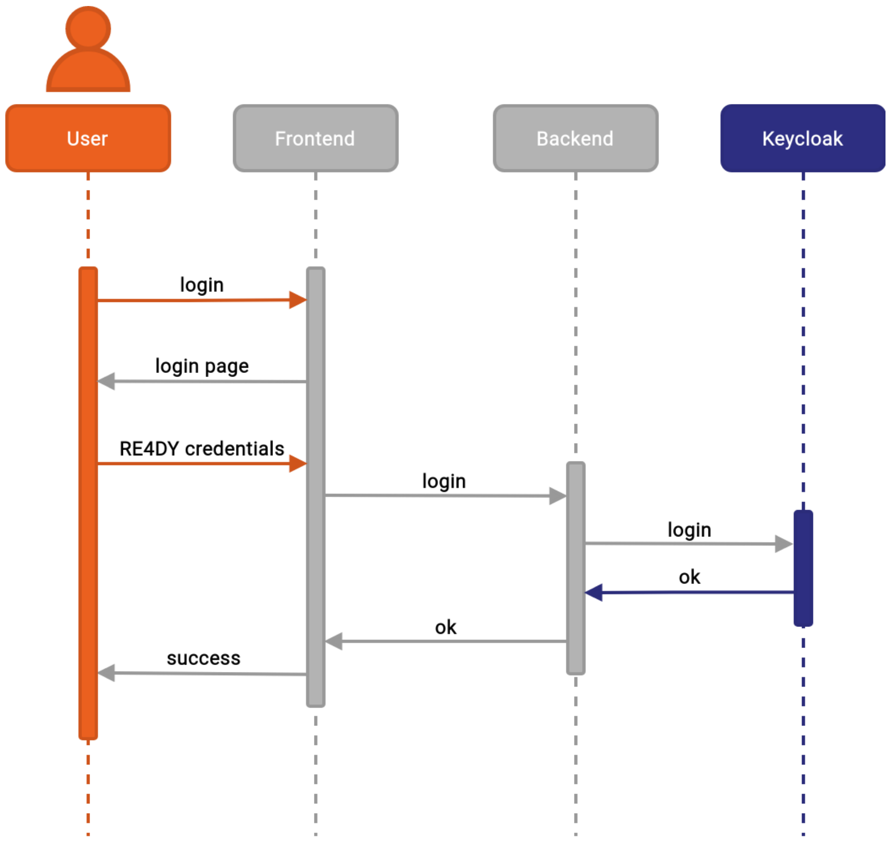

Figura 2 UML Diagram of the login process and the communication of the front-end, authentication back-end and Keycloak services

5. Internal Architecture

Keycloak utilizes a PostgreSQL database for the storage of the RE4DY user base and its configuration settings. It offers the capability for an external application to communicate with it directly through its REST API, or to use a library, such as the Keycloak Java API. Our authentication back-end uses the Java API for the register and login operations. In turn, the RE4DY authentication page in the front-end of the platform uses the exposed API endpoints implemented by the authentication back-end.

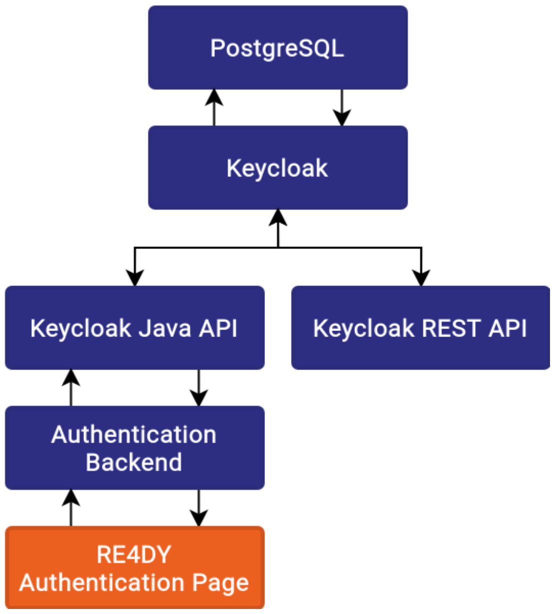

Figura 3 Internal architecture of the Keycloak component and related libraries and services

6. API

- POST /auth/realms/re4dy/protocol/openid-connect/token

  - **Input**:
    ```
    {
       "username": "newuser",
       "password": "1234",
       "grant_type": "password",
       "client_id": "re4dy-login",
    }
    ```
  - **Output**:
    ```
    {
       "access_token": "eyJhb...",
       "expires_in": 300,
       "refresh_expires_in": 1800,
       "refresh_token": "eyJhb...",
       "token_type": "Bearer",
       "not-before-policy": 0,
       "session_state": "6c033496-5551-4bff-898a-78156db13692",
       "scope": "profile email"
    }
    ```
  - **Description**:
    This endpoint is provided natively by Keycloak and it may be used for logging into any application in the re4dy realm. The client_id field denotes the application the user is trying to authenticate through.

The API documentation is also available in the Annex as a Swagger editor .yaml file.

7. Implementation Technology

Keycloak is open-source software which may be installed and executed on bare metal, on a virtual machine, as well as in a containerized manner using Docker and Kubernetes. We have opted for a container-based approach for all the services comprising the authentication back-end of the RE4DY platform, including Keycloak. Therefore, Docker is installed on a virtual machine running the CentOS 7 Linux operating system and the Compose plugin is used to orchestrate the deployment of the necessary services. We use the official Keycloak Docker image as a base for our Keycloak instance.

Keycloak supports the usage of integration APIs for a number of programming languages. In RE4DY, the integration of the other back-end services with the authentication and authorization functionality provided by Keycloak is performed using its Java API. Furthermore, a PostgreSQL database, deployed as a Docker container, is used by the Keycloak container as storage for the realm configuration and user data. Through the Admin Console, we have created the re4dy realm which encompasses the RE4DY user base and a client named re4dy-login, to be used for user registration and login through the RE4DY dashboard.

8. Comments

None.