VOTING Ausmittlung

Authentication Model

Author	Abraxas Informatik AG
Classification	public
Version	1.0
Date	3. Jun 2024

For digital Switzerland. For Sure.

abraxas

Contents

1. Principles

The following principles apply to the authentication model:

- **Standards:** Wherever possible, proven standards are used. Deviations from them, or own solutions are used if existing standards are not suitable.
- **Hiding details:** No information is made public that is not required.
- **External & internal security:** The same guidelines apply to calls within the Abraxas Apps Plattform (AAP) as to external calls. Only authenticated and authorized calls are consistently allowed, regardless of where they come from. Thus, there is no "trust" between the backend services, for example.

2. VOTING IAM

The VOTING IAM software developed by Abraxas is used as the IAM solution. This includes user management, rights management, and the use of VOTING IAM as an identity provider.
The setup of applications, roles and clients are done by VOTING IAM administrators (employees of Abraxas Informatik AG). However, user and rights management can be delegated to responsible persons per tenant. These can, for example, create new users or change the permissions of a user within their tenant. However, it is not possible for such tenant administrators to authorize users for other tenants (or applications or roles of other tenants). Only VOTING IAM administrators can perform such actions. VOTING IAM supports OpenID Connect (OIDC).

3. Procedure

The following is a brief description of the various authentication flows.

3.1 Login flow

As login flow the Authorization Code Flow with PKCE (Proof Key for Code Exchange) is used. After a successful login flow, the user receives, among other things, an access token (a JSON Web Token, JWT), which can be used to call the API of VOTING.

3.2 Logout flow

During a logout, all tokens and information about the user are deleted from the frontend. This happens purely on the client side, neither VOTING IAM nor the backend are informed about the logout process.

3.3 Refresh flow

The refresh flow is not yet implemented. This will be implemented later until go-live (see Exclusion-List).

3.4 Token check in the VOTING Ausmittlung backend

The VOTING Ausmittlung backend must receive the following information as an HTTP header for authentication to be successful:

HTTP Header	Description	Example value
Authorization	Access token of the user	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc...
x-app	Application(s) to be accessed	VOTING-AUSMITTLUNG-ERFASSUNG
x-tenant	Tenant, in which context the API call should be executed	132543547687919

The steps to a successful authentication are as follows:

1. The user's Access Token is checked for validity (not expired, valid audience, etc.).
2. The application(s) are checked to see if they are included in the configured whitelist of audiences.
3. The access token and the application(s) are sent to the token endpoint.
4. In response, a "Role Token" is returned containing the user's roles for the listed applications.
5. The "Role Token" is checked for validity.
6. Those roles are extracted which are valid for the client (value from x-tenant) (roles in the "Role Token" are only filtered by application, not by tenant).
7. If everything ran without errors, the user is considered authenticated.
8. User, tenantand roles saved for authorization for the current request.

3.4.1 Audiences

Two audiences are configured for VOTING.

Name	Description	Usage
VOTING-AUSMITTLUNG-ERFASSUNG	Audience for VOTING Ausmittlung Erfassung	VOTING Ausmittlung Erfassung Frontend, VOTING Ausmittlung Backend
VOTING-AUSMITTLUNG-MONITORING	Audience fpr VOTING Ausmittlung Monitoring	VOTING Ausmittlung Monitoring Frontend, VOTING Ausmittlung Backend

There is one frontend per audience, since from the user's point of view these are two different applications with different functions. However, both applications access the same data, which is why there is only one backend that is responsible for both audiences.
When API accesses are triggered from the frontends, the frontends send their respective audiences to the backend via x-app headers.