VOTING Stimmregister

Authentication Model

Author	Abraxas Informatik AG
Classification	public
Version	1.1
Date	31. Jan 2024

Für die digitale Schweiz. Mit Sicherheit.

abraxas

Contents

1. VOTING IAM

The VOTING IAM software developed by Abraxas is used as the IAM solution. This includes user management, role management and the use of VOTING IAM as an identity provider.
The setup of applications, roles and clients is 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. 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 this kind of higher privileged actions.
VOTING IAM has full support for the OpenID Connect (OIDC) standard.

2. Procedure

2.1 Login flow

The user login flow in the frontend uses the authorization code flow with PKCE (Proof Key for Code Exchange - https://tools.ietf.org/html/rfc7636). After successful authentication, the frontend application (user agent) is provided with an access token - JSON Web Token (JWT) - that can be used to authenticate against the backend services. Among with the access token the user receives further identity or authorization information from the identity and permission services used by the frontend for authorization and visual representation purposes.

2.2 Logout flow

After a successful logout, all tokens and other user specific information is deleted from the client's user-agent (frontend) and revoked. VOTING IAM protects active sessions by invalidating session cookies and access tokens after the user has logged out.

2.3 Refresh flow

The frontend uses the token refresh flow to prevent long-lived access tokens.

2.4 Access Token Validation

The VOTING Stimmregister backend requires the following information as an HTTP header to be able to fully authenticate a user:

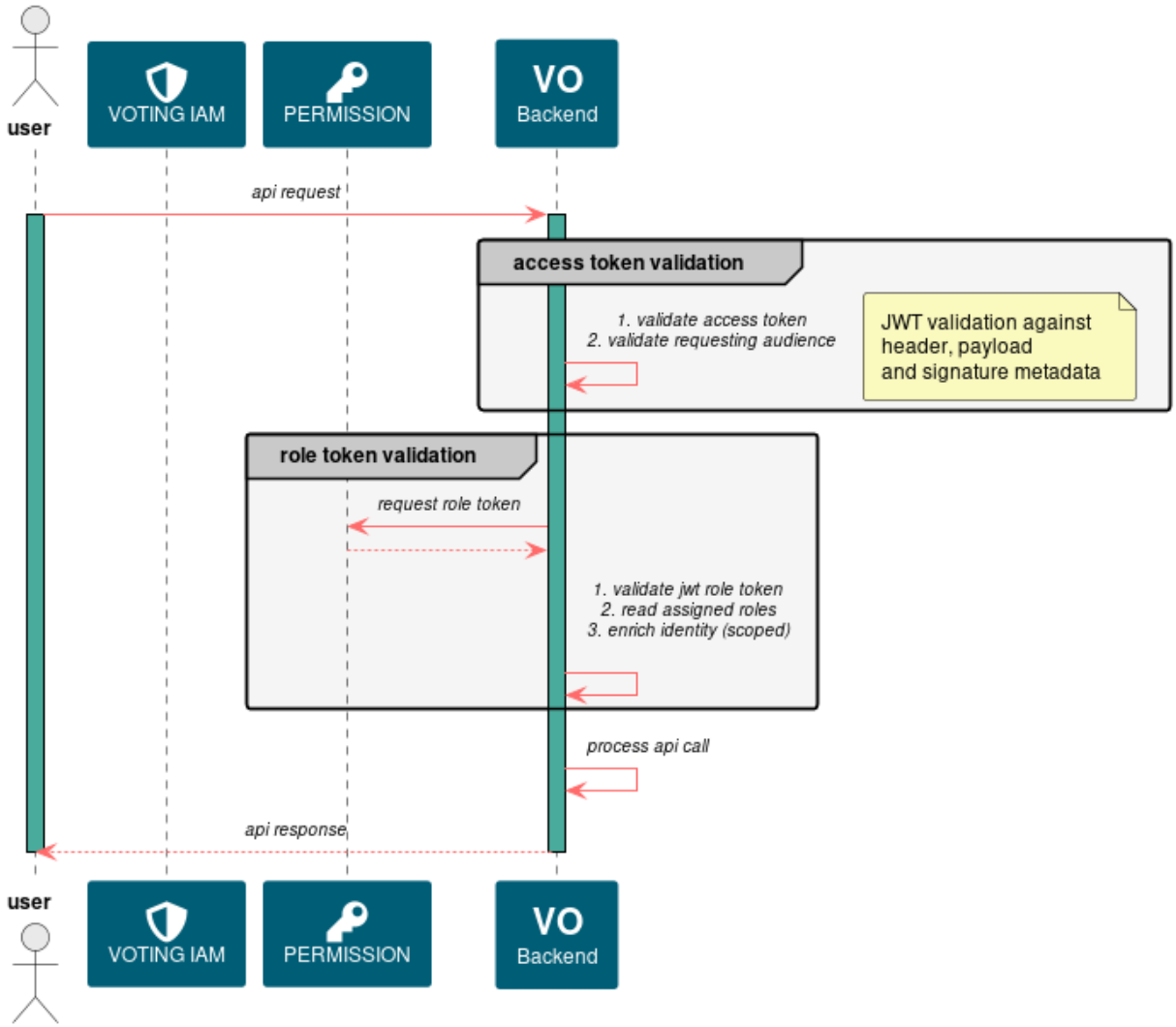
HTTP Header	Description	Example value
Authorization	Access token of the user	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc...

HTTP Header	Description	Example value
x-app	Application(s) to be accessed	VOTING-STIMMREGISTER
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 all validation succeeded, the user is considered authenticated.
8. Users, tenants and roles are saved for the current request to prove valid authentication against further evaluating logic.

2.4.1 User Flow



2.5 Audiences

The following audiences are used for VOTING Stimmregister:

Name	Authenticated Application
VOTING-STIMMREGISTER	VOTING Stimmregister Frontend
VOTING-STIMMREGISTER-ESERVICE-EGV-ZUS-GW	Zustellplattform Backend Service
VOTING-STIMMREGISTER-ESERIVCE	VOTING Stimmregister E-Service
VOTING-STIMMREGISTER-EVOTING	VOTING Stimmregister E-Voting
VO-SR-ESERVICE	VOTING Stimmregister Backend

2.5.1 E-Voting System Components