**System RBAC Zero-Polling z OPAL + OPA**

**w Symfonia**

**PRD – Dokument wymagań produktowych**

Metryka:

|  |  |  |
| --- | --- | --- |
| **Autor** | **Data utworzenia** | **Data ost. modyfikacji** |
| Jacek Paszek | 12.06.2025 |  |

# 

# 1. Przegląd

Ten dokument opisuje architekturę i komponenty systemu autoryzacji opartego na zdarzeniach (zero-polling) z użyciem:

• OPAL Server i Client do dystrybucji polityk i danych w czasie rzeczywistym  
• OPA (Open Policy Agent) jako silnika podejmowania decyzji lokalnie  
• API do provisioning’u i dostarczania danych o nowych tenantach i zmianach w uprawnieniach tenantów.  
• Repozytorium polityk GitHub jako źródła wersjonowanych polityk Rego

Obraz zawierający diagram, tekst, linia, Plan

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.System umożliwia skalowalną, multitenantową autoryzację typu RBAC z ACL dla dostępu do danych w kontekście tenantów i firm.

# 2. Streszczenie użycia

Środowisko SaaS obsługujące:

• Wiele aplikacji (np. fk, ksef, hr, edeklaracje, edokumenty)  
• Użytkownicy przypisani do wielu tenantów i firm w ich obrębie  
• Uprawnienia oparte na rolach per aplikacja  
• Dostęp do firm ograniczony przez ACL na poziomie tenantów

# 3. Kluczowe cechy architektury

## 3.1 Brak aktywnego odpytywania (Zero Polling)

• Brak cyklicznych zapytań do API źródłowego  
• Wszystkie aktualizacje danych są wyzwalane przez POST /data/config z publikacją topicu

## 3.2 Ładowanie danych per tenant

• Każdy tenant posiada swój topic: access.companies.<tenant\_id>  
• Provisioning dynamicznie dodaje źródła danych dla tenantów do OPAL  
• Nazewnictwo topiców odzwierciedla strukturę danych: access.companies.<tenant\_id> oznacza firmy (companies) należących do tenanta.

Uwaga: **Companies są encjami klientów końcowych tenantów**, a nie firmami płacącymi za usługę. Każda firma jest powiązana z dokładnie jednym tenantem.

## 3.3 Inline OPA

• OPAL Client uruchamia OPA wewnątrz tego samego kontenera  
• Nie trzeba konfigurować zewnętrznego serwera OPA

OPAL Client może uruchamiać OPA w trybie inline, jako współdzieloną instancję dla wielu aplikacji lub jako osobną instancję per aplikacja w fazie 2

**Faza 1:** OPAL Client współdzielony (centralny)

* OPAL Client uruchomiony w domenie Service Hub (OPA).
* Obsługuje polityki i dane dla wielu aplikacji równocześnie.
* OPA służy jako centralny PDP (Policy Decision Point) — odpytywany przez aplikacje przez REST/HTTP.
* Aplikacje nie mają własnego OPAL Clienta ani instancji OPA.

**Faza 2:** OPAL Client per aplikacja

* Aplikacje mają własnego OPAL Clienta + OPA (uruchamiane np. jako sidecar lub proces).
* Każdy OPAL Client subskrybuje polityki globalne + aplikacyjne (policies/<app>).
* Dane ACL są również ładowane tylko dla potrzeb danej aplikacji.
* Aplikacja odpytuje lokalną instancję OPA.

# 4. Usługi

## 4.1 OPAL Server

• Publikuje zdarzenia do klientów  
• Umożliwia dynamiczne dodawanie źródeł danych przez API /data/config  
• Klonuje repozytorium z politykami Rego z GitHub:  
• Wymaga konfiguracji zmiennej środowiskowej OPAL\_POLICY\_REPO\_URL  
• Każda aplikacja może umieszczać swoje polityki w podkatalogu, np. policies/fk, policies/ksef

OPAL Server może być skonfigurowany do automatycznego śledzenia repozytorium GitHub z politykami Rego. Każdy zespół aplikacji może utrzymywać własny podkatalog (policies/<app>) i publikować tam własne reguły. OPAL Server po wykryciu zmian publikuje odpowiedni topic (policy:<app>) do OPAL Clientów.

## 4.2 OPAL Client

• Subskrybuje topic’i tenantów i polityki z repozytorium  
• Po odebraniu zdarzenia pobiera dane i aktualizuje OPA  
• Działa z OPA w trybie inline

## 4.3 OPA

• Odbiera polityki i dane z OPAL  
• Wykonuje ewaluację przez data.policies.rbac.allow lub inne entrypointy (np. REBAC)

## 4.4 Data Provider API

• Endpoint: GET /access/  
• Zwraca pełną strukturę danych dla jednego tenanta:

{  
 "tenants": { "user42": ["tenant125"] },  
 "companies": { "user42": { "tenant125": ["company124"] } },  
 "roles": { "user42": { "fk": ["fk\_admin"] } },  
 "permissions": { "fk": { "fk\_admin": ["view\_entry", "edit\_entry"] } }  
}

## 4.5 Provisioning API

• Endpoint: POST /provision-tenant  
• Rejestruje nowego tenanta przez wywołanie OPAL:

{  
 "url": "http://data-provider-api:8100/access/tenant125",  
 "topics": ["access.companies.tenant125"],  
 "dst\_path": "access.companies.tenant125",  
 "polling\_interval\_seconds": 0  
}

# 5. Logika polityk (OPA)

## 5.1 Punkt wejścia RBAC

• data.policies.rbac.allow sprawdza:  
• Czy użytkownik należy do tenanta?  
• Czy użytkownik ma dostęp do firmy w ramach tenanta?  
• Czy użytkownik ma rolę zawierającą daną akcję w danej aplikacji?

## 5.2 Moduły RBAC

• roles.rego: weryfikacja ról i uprawnień  
• access.rego: weryfikacja dostępu do tenantów i firm  
• allow.rego: główna reguła decyzyjna łącząca oba powyższe

## 5.3 Moduły REBAC (Faza 2)

• Nowa polityka rebac.rego będzie sprawdzać dostęp do zasobów z relacyjnym ACL:

package rebac.authz  
  
default allow = false  
  
allow {  
 some acl in data.acls  
 acl.user == input.user  
 acl.organization == input.tenant\_id  
 acl.action == input.action  
 input.resource in [acl.resource, acl.project, acl.component]  
}

• Każda aplikacja będzie mogła umieścić takie reguły w swoim katalogu repozytorium polityk.   
• OPAL Client aplikacji będzie subskrybował zarówno policies/base, jak i policies/

# 6. Lokalny rozwój

## 6.1 Uruchomienie

git clone https://github.com/plduser/opa-zero-poll  
cd opa-zero-poll  
docker compose up

## 6.2 Provisionowanie tenant’a

curl -X POST http://localhost:8000/provision-tenant \  
 -H "Content-Type: application/json" \  
 -d '{"tenant\_id": "tenant125"}'

## 6.3 Synchronizacja danych ACL

curl -X POST http://localhost:7002/data/config \  
 -H "Content-Type: application/json" \  
 -d '{"topics": ["access.companies.tenant125"]}'

## 6.4 Zapytanie autoryzacyjne RBAC

curl -X POST http://localhost:8181/v1/data/policies/rbac/allow \  
 -H "Content-Type: application/json" \  
 -d @rbac\_full\_input.json

# 7. Możliwości rozszerzenia

• Dodanie nowych aplikacji: wystarczy rozbudować sekcję roles i permissions  
• Wsparcie dla nowych typów zasobów (np. dokumenty): poprzez nowe moduły i struktury ACL  
• Wsparcie dla REBAC (relacyjnego RBAC): poprzez rebac.rego i dane data.acls  
• Skalowanie klienta OPAL per aplikacja lub per tenant

# 8. Dalsze kroki

* Obsługa cold start i restartu systemu
* Pokrycie testami (opa test)
* Wydajność - benchmarki (opa bench)
* Optymalizacja czasu uruchomienia (np. preload danych tenantów)
* Konfiguracja GitHub jako repozytorium polityk z webhookiem do OPAL
* Audyty logów decyzyjnych i dostępu

# 9. Zasoby

• Dokumentacja OPA  
• Projekt OPAL  
• Przykłady RBAC w Rego