Token Exchange

Keycloak's Secret Weapon for Platforms
Keycloak Dev Day 2025

Darmstadt, 06.03.2025

CONCISO.

About me

Sven-Torben Janus

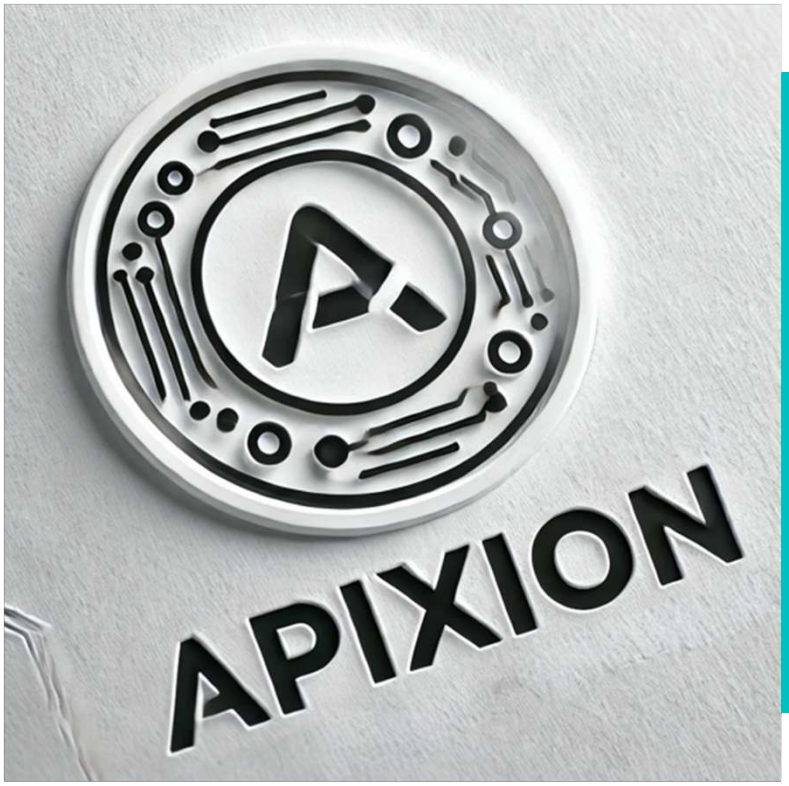
Partner,
Principal Architect

sven-torben.janus@conciso.de

LinkedIn
@sventorben

CONCISO.

What is APIxion?



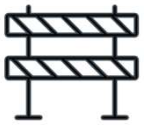
API Management
Providing Secure Access to internal and external APIs

Infrastructure as a Service
Enabling teams to provision and manage cloud-native resources.

CI/CD Automation
Streamlining deployment workflows for microservices and applications.

CONCISO.

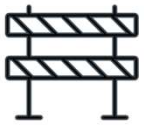
APIxion's Authentication Challenges



Microservices misused frontend user tokens

instead of having their own authentication

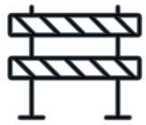
→ over-permissioned tokens



API keys were hardcoded into services

for external integrations

→ no clear delegation model



No clear access control

between internal services and across planes

→ increased the attack surface

CONCISO.

APIxion's Major Goals

Internal service-to-service authentication

without misusing user tokens

Secure external API access

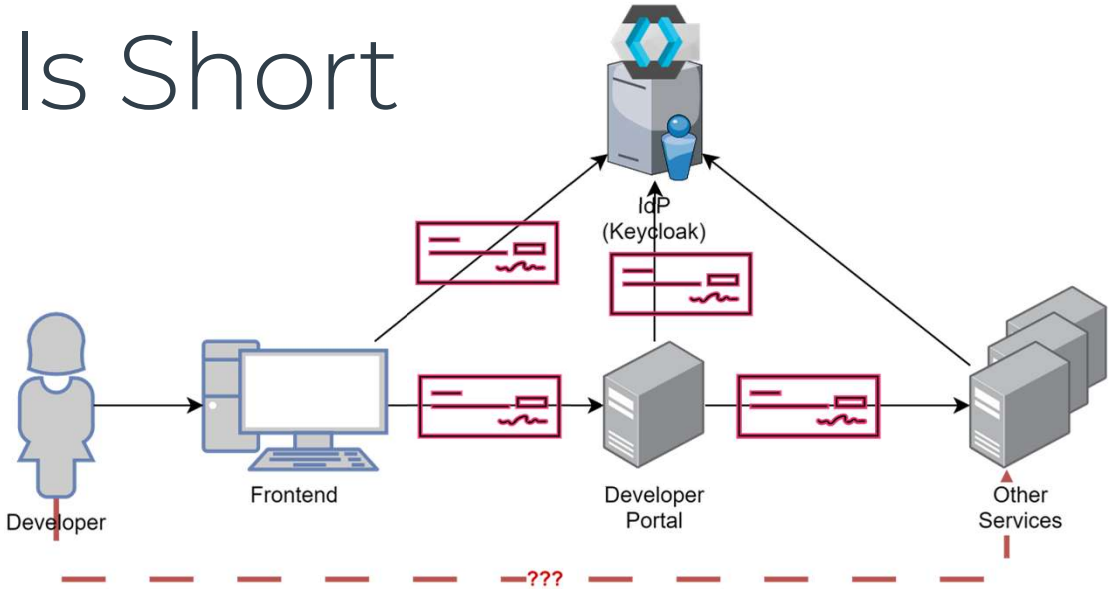
without hardcoded credentials

Fine-grained permission control

with audience-restricted tokens

CONCISO.

Where OAuth Falls Short (Delegation)



Forwarding the user's access token

This exposes scopes that backend services shouldn't have access to.

Using Client Credentials Grant (Service Account)

This issues a token for the backend service but doesn't retain the user's identity or permissions.

Stitching together both methods manually

This is cumbersome and error-prone, leading to security vulnerabilities and poor access control.

Poor-Man's Delegation & Why It Fails

🚨 This breaks security principles like least privilege and separation of concerns.


```
{
  "iss": "https://.../realms/apixion",
  "azp": "frontend",
  "sub": "developer-123@apixion",
  "aud": ["frontend","portal",
          "some-service"],
  "realm_access":{"roles":["user"]},
  "resource_access": {
    "frontend":{"roles":["user"]},
    "portal":{"roles":["admin"]},
    "some-service":{"roles":["viewer"]}
  }
}
```

01
Token Exchange

CONCISO.

Introducing Token Exchange RFC8693

User-to-Service Token Exchange

Securely exchange a user token for a backend service token.

Centralized API Token Management

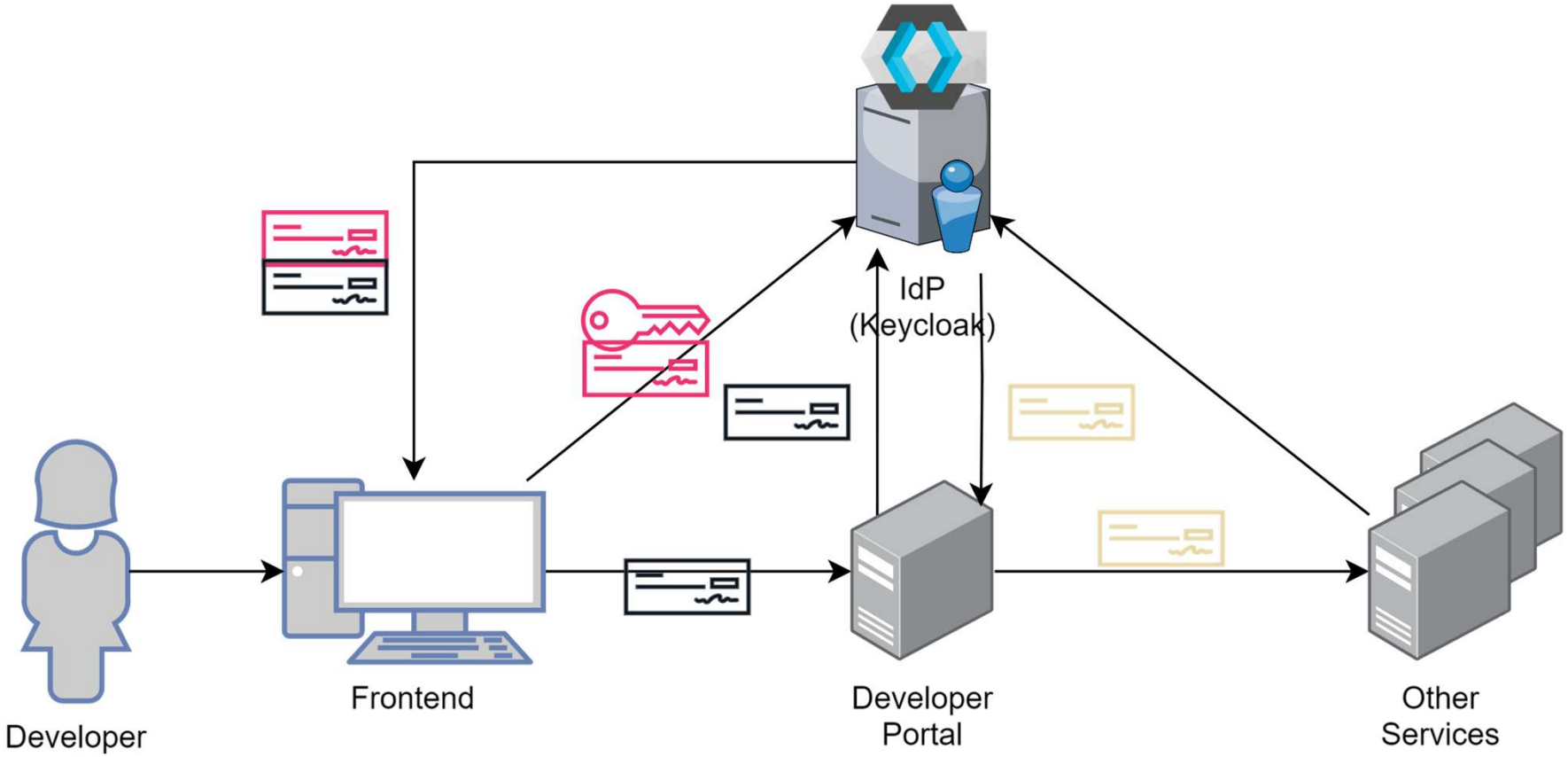
Use a centrally managed API token to call external APIs.

Impersonation & Delegation

Act on behalf of another user or service, if permitted.

CONCISO.

Token Exchange – How It Works

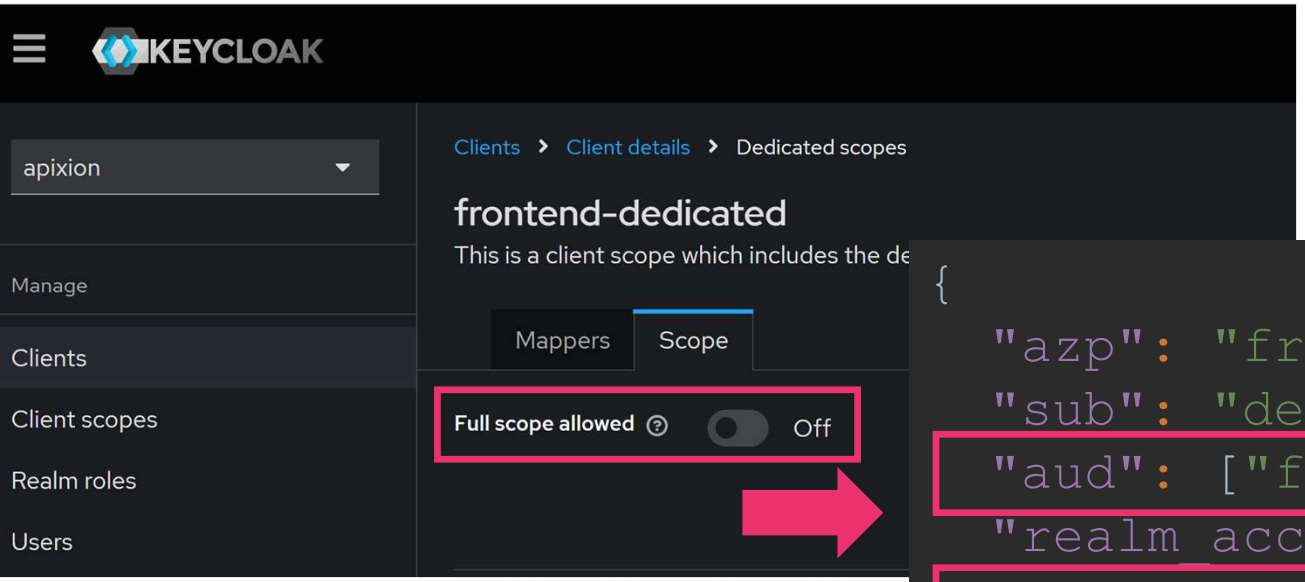


CONCISO.

02
Token Exchange in Keycloak

CONCISO.

Restricting Full Scope for Clients



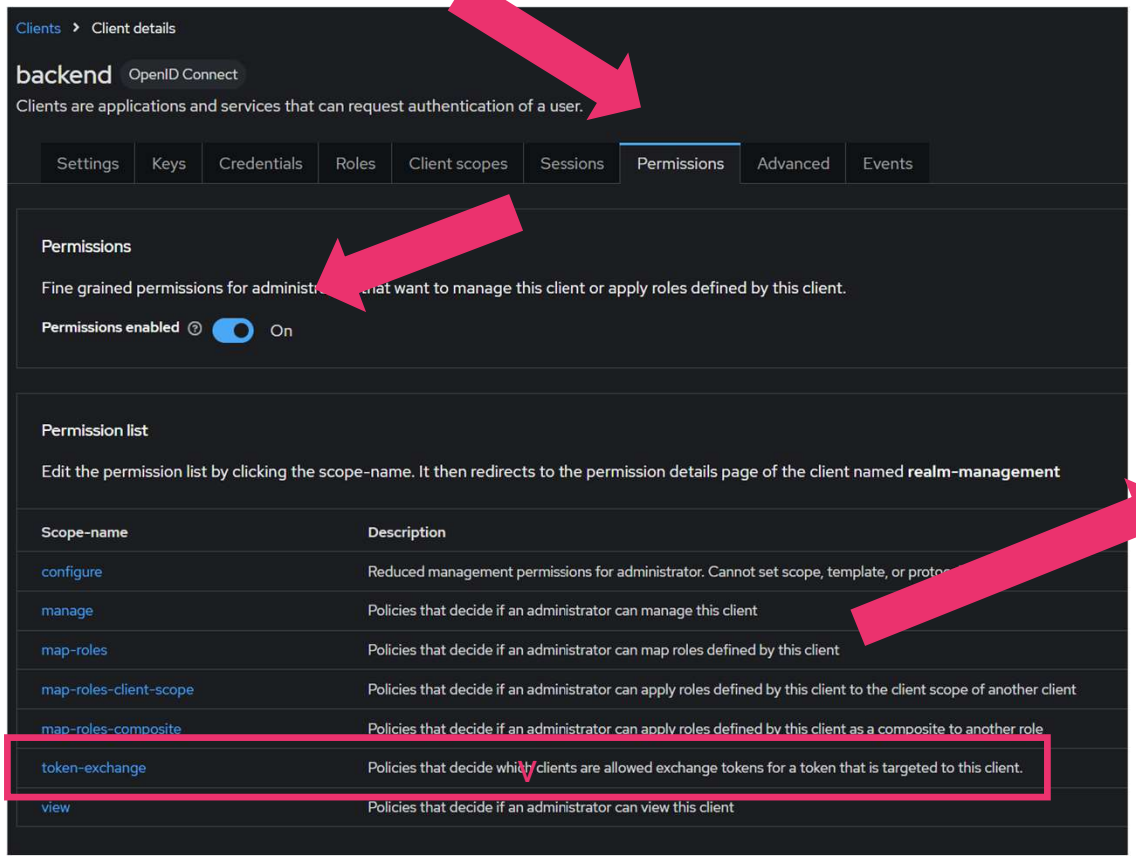

```
{
  "azp": "frontend",
  "sub": "developer-123@apixion",
  "aud": ["frontend"],
  "realm_access":{"roles":["user"]},
  "resource_access": {
    "frontend":{"roles":["user"]},
  },
  "iss": "https://.../realms/apixion"
}
```

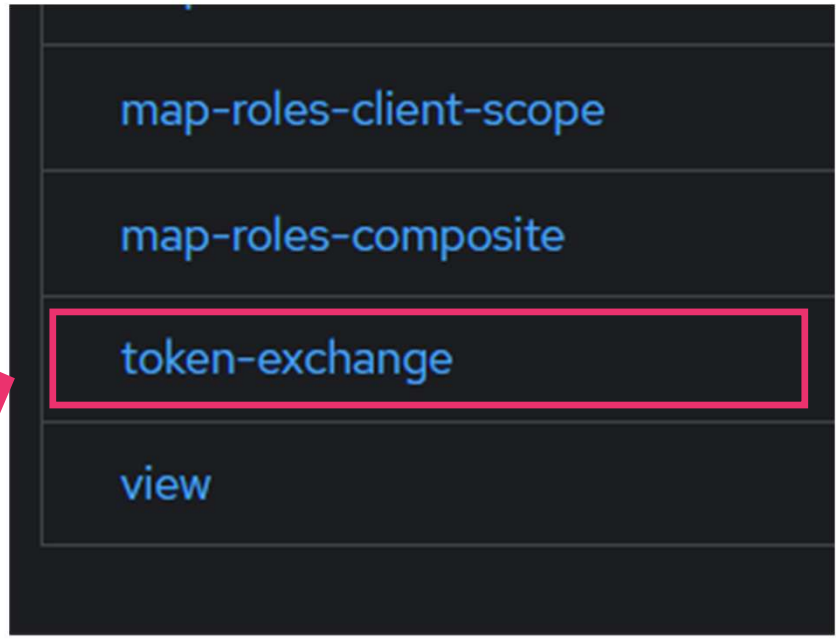
CONCISO.

Configuring Token Exchange

```
services:
  keycloak:
    container_name: keycloak
    hostname: keycloak
    image: quay.io/keycloak/keycloak:26.1.3
    environment:
      KC_BOOTSTRAP_ADMIN_USERNAME: admin
      KC_BOOTSTRAP_ADMIN_PASSWORD: admin
      KC_FEATURES: token-exchange,fine-grained-authz
    command: ['start-dev']
    ports:
      - 8080:8080
```

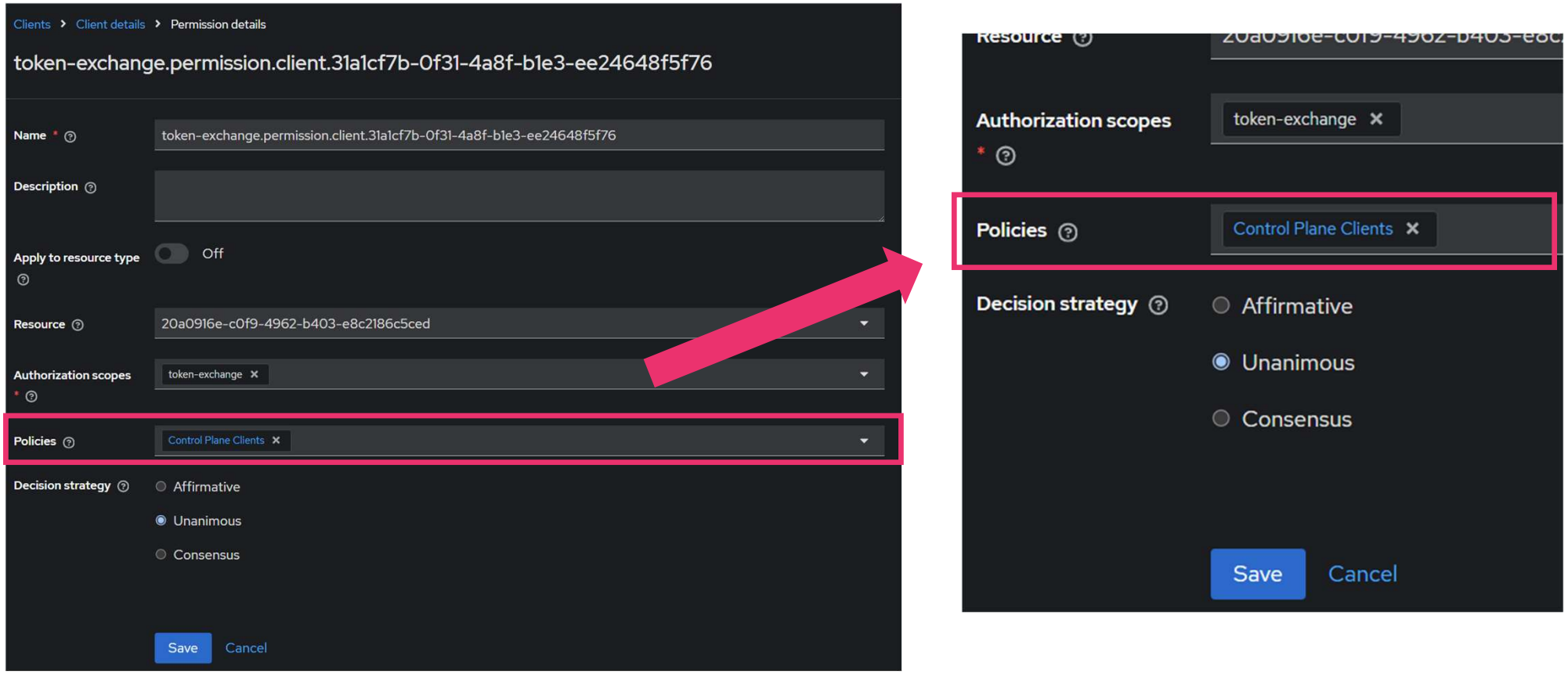
Assigning Permissions





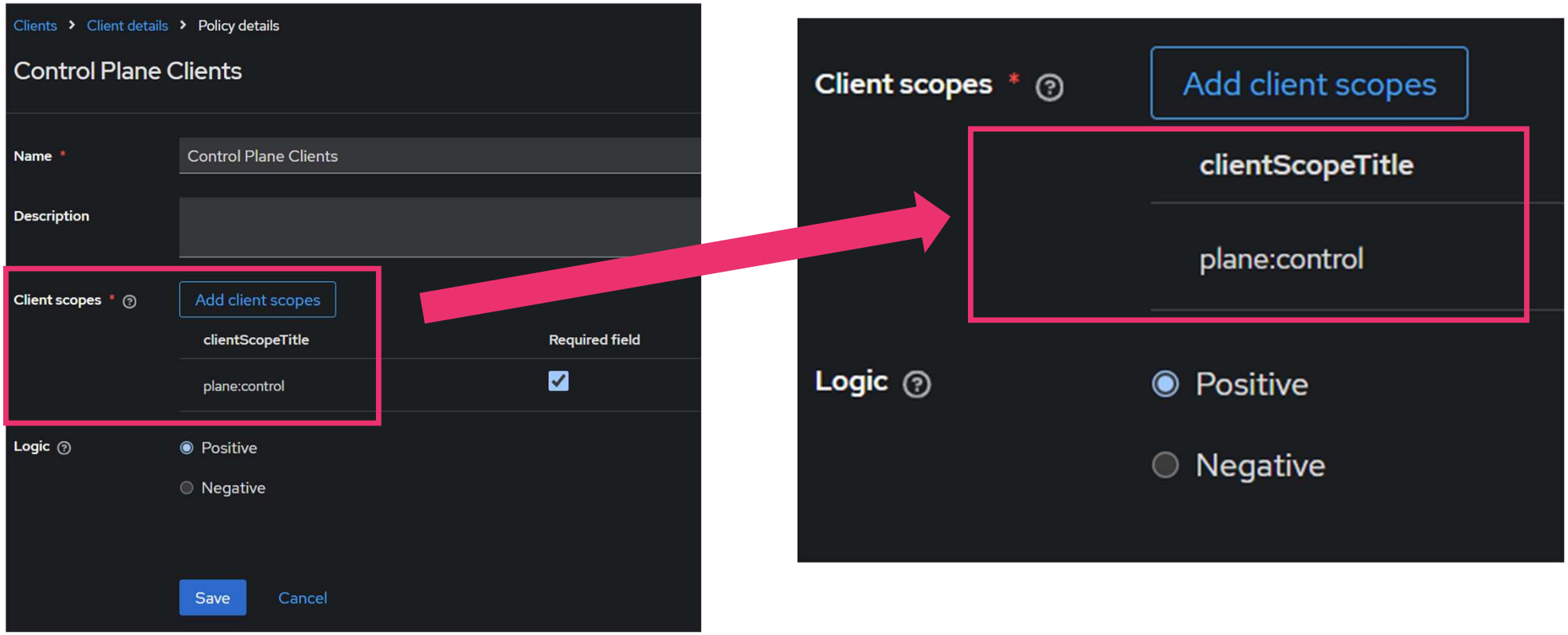
CONCISO.

Assigning Permissions



CONCISO.

Assigning Permissions



CONCISO.

Performing a Token Exchange

```
POST /realms/apixion/protocol/openid-connect/token
HTTP/1.1
Host: keycloak.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic BASE64(client_id:client_secret)

grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&subject_token=eyJhbGciOiJIUzI1NiIsInR5cCI...
&subject_token_type=urn:ietf:params:oauth:token-type:access_token
&requested_token_type=urn:ietf:params:oauth:token-type:access_token
&audience=some-backend-service
```

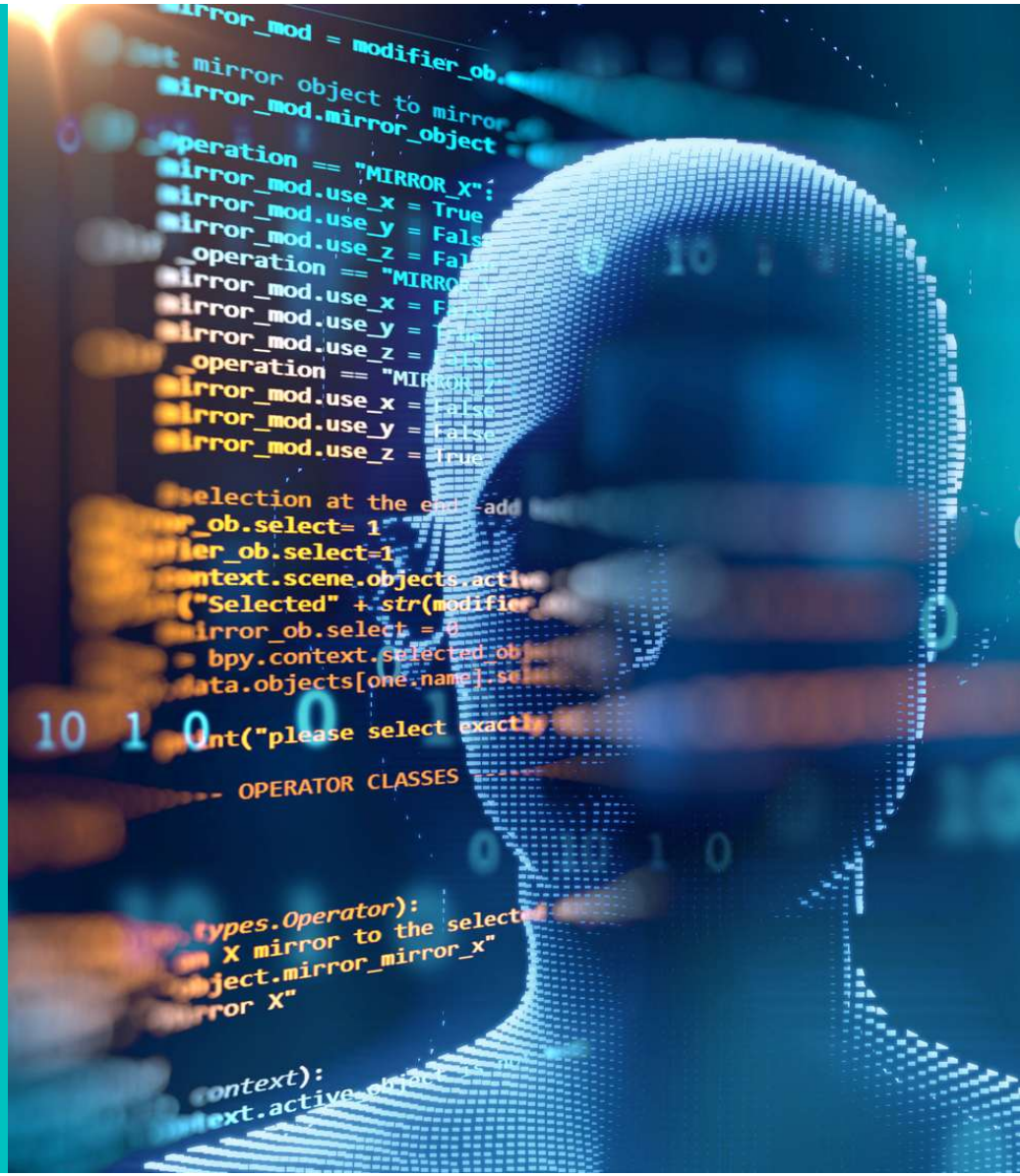
Token Exchange Response and Token Validation

```
{
  "iss": "https://.../realms/apixion",
  "azp": "portal",
  "sub": "developer-123@apixion",
  "aud": ["some-backend-service"],
  "realm_access": { "roles": ["user"] },
  "resource_access": {
    "some-backend-service": { "roles": ["viewer"] },
  }
}
```

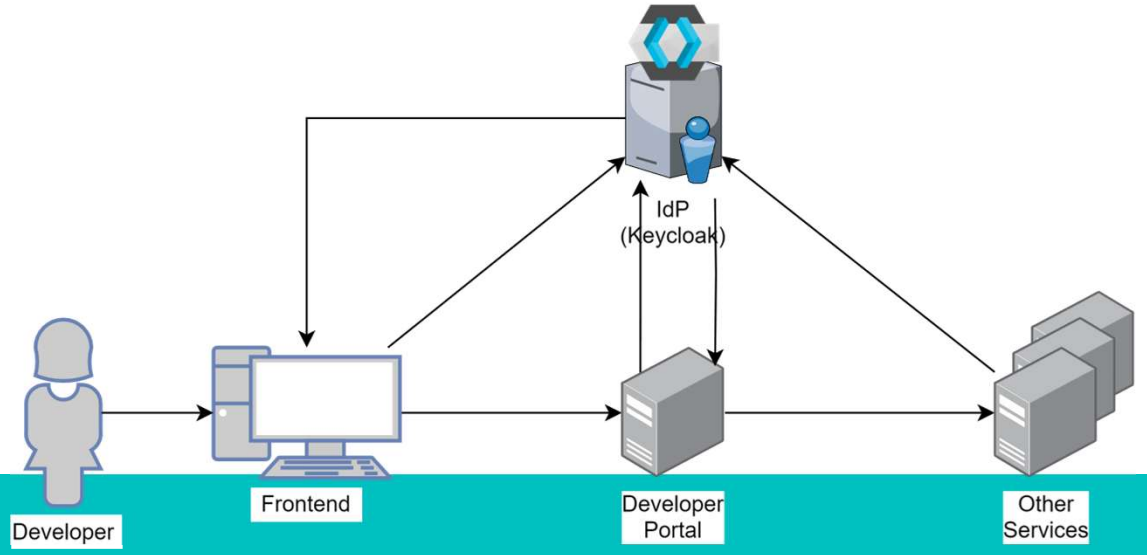
03

Types of Token Exchange

CONCISO.



Internal-to-Internal Token Exchange



Step 1

The Frontend sends a request with a user token to Developer Portal.

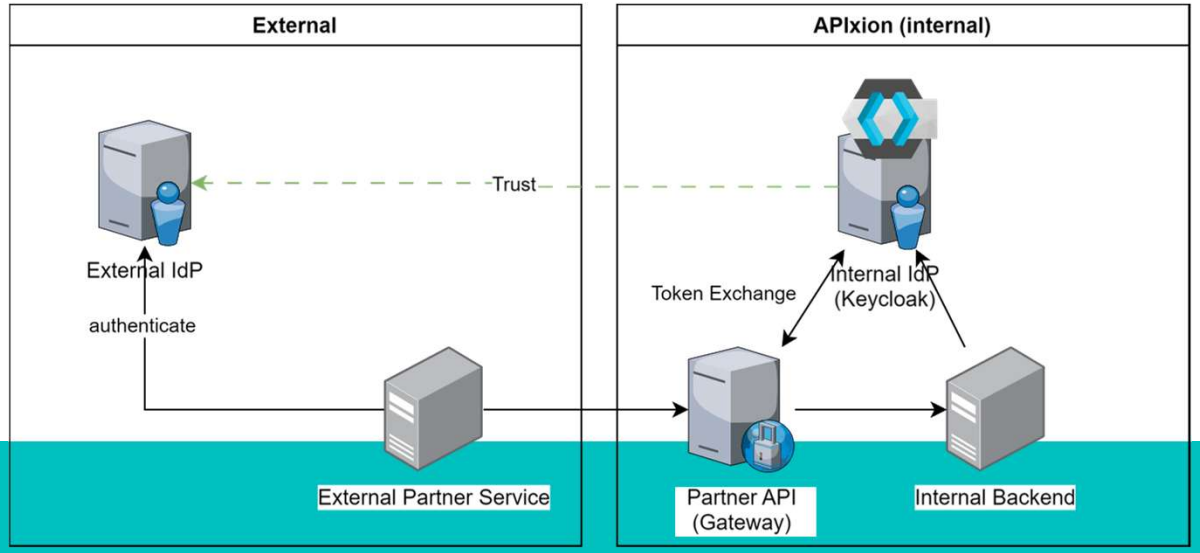
Step 2

Developer Portal exchanges the token for a new one meant for an Other Service.

Step 3

Developer Portal calls Other Service with the new token scoped for that service.

External-to-Internal Token Exchange



Step 1: Authentication

The external service authenticates with its **own IdP,** gets an external token, and calls an internal API Gateway.

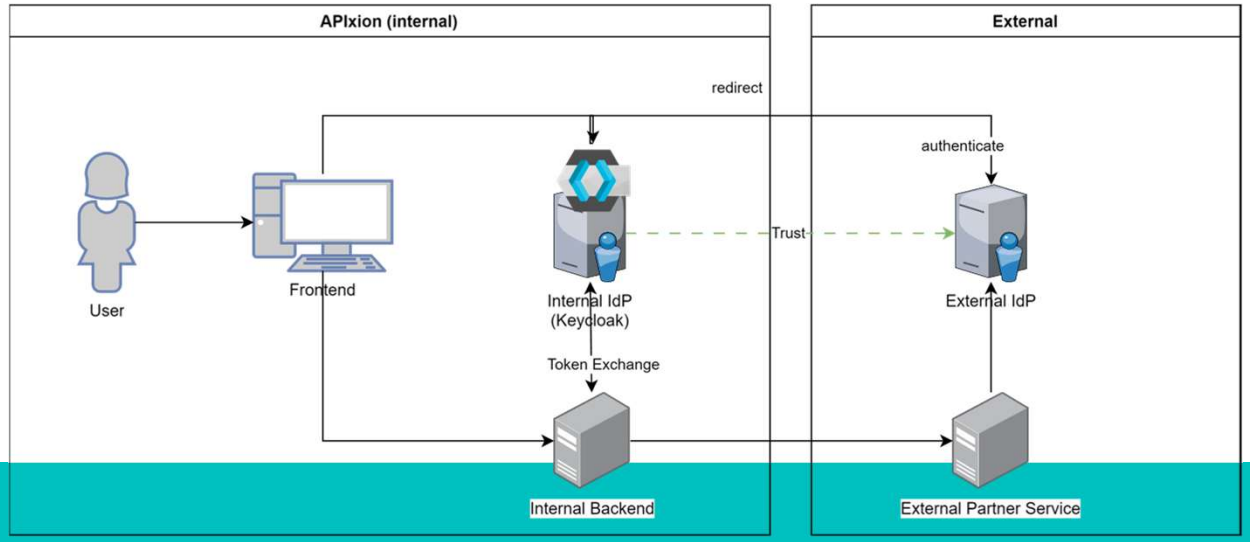
Step 2: Token Exchange

It exchanges the external token for a Keycloak token scoped for internal services.

Step 3: Service Request

It uses the new Keycloak-issued token to call the internal backend.

Internal-to-External Token Exchange



Step 1: Authentication

User authenticates to the internal and external IdP (identity provider federation)

Step 2: Forward Token

Internal backend receives a request from the frontend.

Step 3: Exchange Token

Backend exchanges it for a new token and calls the external API using the exchanged token.

04
Platforms

CONCISO.

Understanding Platform Planes

Control Plane: Manages orchestration, security policies, and infrastructure automation

Management Plane: Handles API gateways, service discovery, and centralized IAM.

Data Plane: Processes user traffic, application workloads, and storage operations.

CONCISO.

Examples of Platform Planes

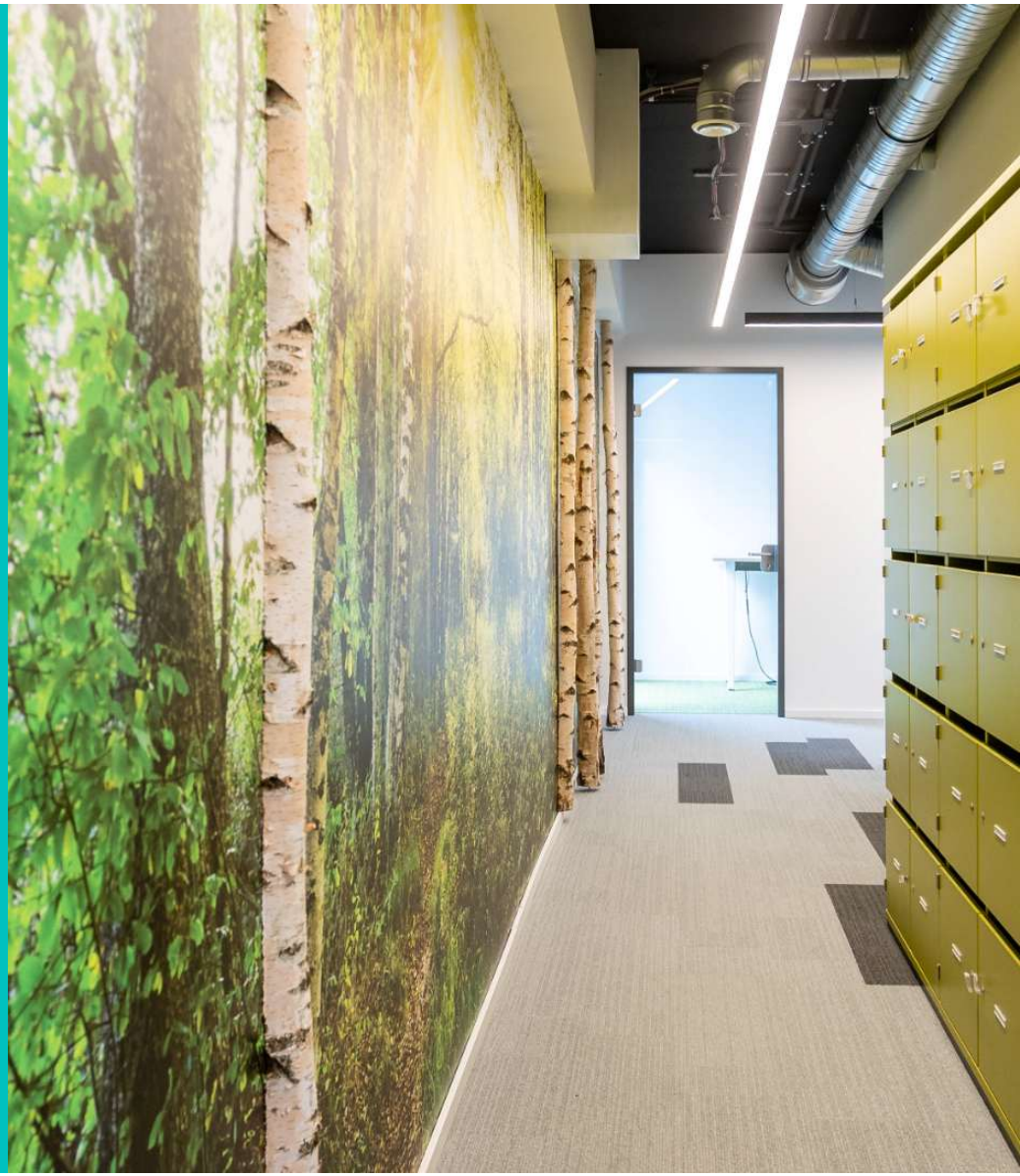
Platform/Plane Type	 Control Plane	 Management Plane	 Data Plane
Cloud Providers	IAM, Policy Enforcement	API Gateway, Service Mesh	Compute & Storage
Service Meshes	Istio Pilot, Consul Control Servers	Authentication & Service Discovery (Keycloak, SPIRE)	Sidecar Proxies, API Traffic (Envoy, Linkerd proxy)
Enterprise SaaS	Tenant Management, Admin APIs	Identity Federation, API Security Policies	Tenant-Specific Applications & Data Stores

Planes, Trust Relationships, and Token Exchange

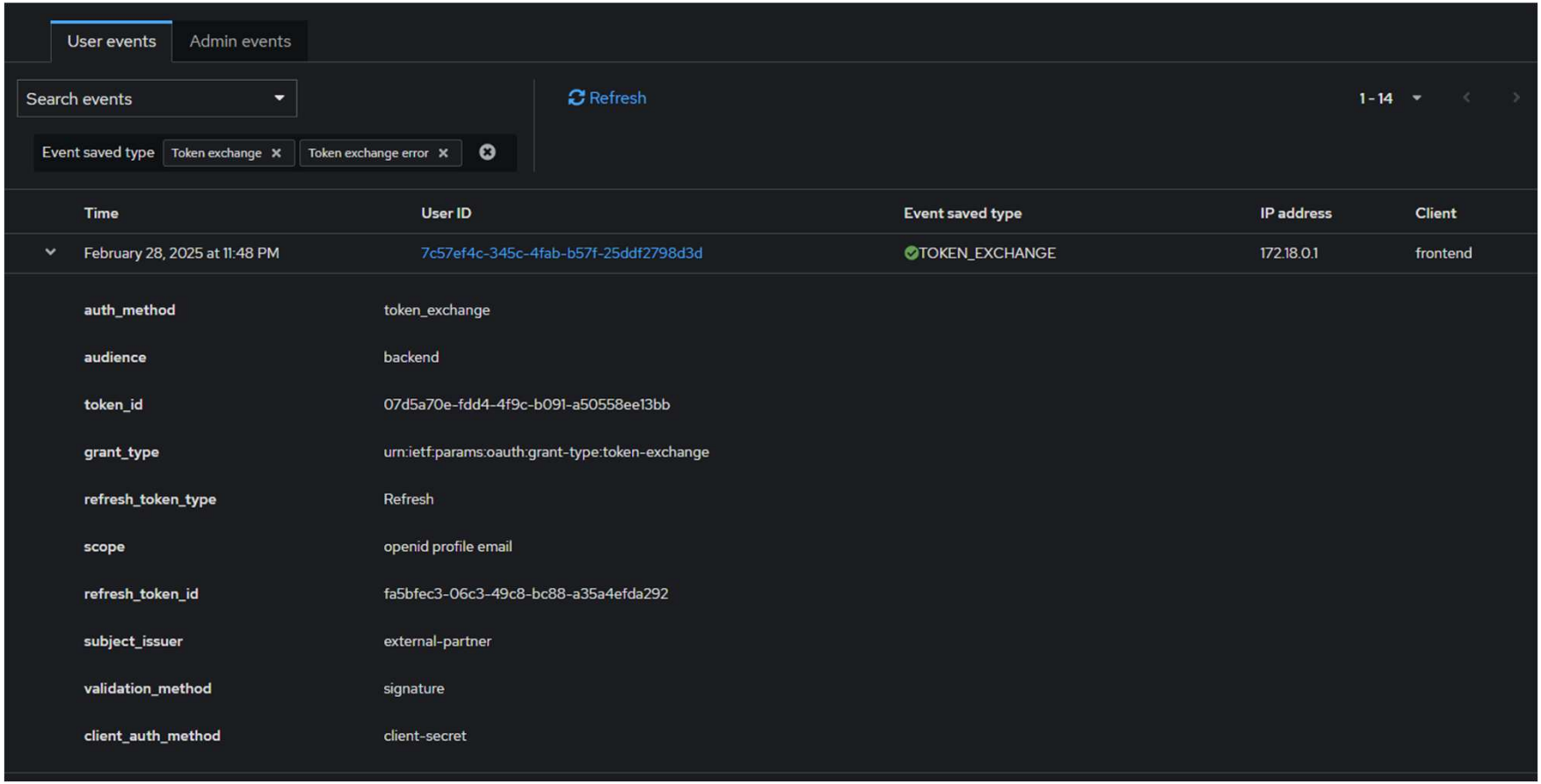
Trust Relationship	Example Use Case	Token Exchange Scenario
Control Plane → Management Plane	Multi-tenant SaaS platform managing identities across tenants.	A **Tenant Management Service** exchanges its control token for a **management token** to configure new identity providers in Keycloak.
Management Plane → Data Plane	API security enforcement for customer data.	An **API Gateway** exchanges its token for a **tenant-scoped token** before calling a customer's backend service.
Data Plane → Management Plane	Ensuring tenant microservices cannot escalate privileges.	A **tenant analytics dashboard** requests a **read-only token** to access metadata from APIxion's Service Registry.
Data Plane → Control Plane	Preventing tenant services from modifying platform configurations.	**Blocked by policy**—Tenant microservices are prevented from requesting control over platform-wide resources.

05
Observability

CONCISO.



Audit Logs



CONCISO.

Monitoring – Logging

```
2025-02-28 23:10:15,414 WARN  [org.keycloak.events] (executor-thread-3)
    type="TOKEN_EXCHANGE_ERROR",
    realmId="c6311f0b-e87a-423c-84e2-74f2a8618b40", realmName="apixion",
    clientId="frontend", userId="null", ipAddress="172.18.0.1",
    error="not_allowed", reason="client not allowed to exchange to audience",
    auth_method="token_exchange",
    audience="backend",
    grant_type="urn:ietf:params:oauth:grant-type:token-exchange",
    client_auth_method="client-secret"
2025-02-28 23:10:57,757 DEBUG [org.keycloak.events] (executor-thread-15)
    type="TOKEN_EXCHANGE„
    realmId="c6311f0b-e87a-423c-84e2-74f2a8618b40", realmName="apixion",
    clientId="frontend",
    userId="2bf85a60-3488-40e8-828b-2fbd25086834",
    sessionId="08e191c7-f3d4-4499-9c86-a2adf091ed03", ipAddress="172.18.0.1",
    auth_method="token_exchange",
    audience="backend",
    ...
```

Monitoring – Event Metrics

```
curl -s https://keycloak/metrics
| grep 'event="token_exchange"'

keycloak_user_events_total{
    client_id="portal",
    error="",
    event="token_exchange",
    idp="",
    realm="apixion"} 15422.0
keycloak_user_events_total{
    client_id="portal",
    error="not_allowed",
    event="token_exchange",
    idp="",
    realm="apixion"} 38.0
```

CONCISO.

06
Takeaways

CONCISO.

How APIxion Uses Token Exchange

Frontend User Authentication → Internal API Access

The frontend receives a **user token** but exchanges it for a **backend-scoped token** before making API calls

Internal Service-to-Service Authentication

Services use **Token Exchange to request scoped tokens** instead of forwarding user credentials.

External API Integrations

When an internal API needs to call an **external service**, it exchanges its Keycloak-issued token for an **API-specific access token**

Platform Segmentation

APIxion **segregates trust between its Control, Management, and Data Planes**, ensuring **least privilege** at every level

CONCISO.

Best Practices on Platform-Level

Enforce Fine-Grained Permissions

Always configure **strict Token Exchange policies** per client. Do not allow unrestricted token exchange.

Disable Full Scope for Clients

Ensure clients/services only get the **minimal scopes** they need, preventing token misuse.

Use Audience Restrictions

Tokens should always have **specific target audiences** to prevent cross-service misuse.

Establish Clear Trust Boundaries

If your platform has **Planes**, define **explicit trust relationships** and enforce separation of concerns.

Monitor and Audit Token Usage

Regularly inspect logs and metrics for **unexpected token exchange requests** to detect misconfigurations or security threats

Limit Token Exchange Availability

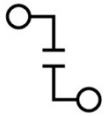
Not all clients should be able to exchange tokens—restrict it to **approved services only** via Keycloak permissions

Final Thoughts

Why Token Exchange is Key to Secure Platforms

Token Exchange is critical for modern platforms

It prevents Poor-Man's Delegation



It enforces trust and separation in a platform

It strengthens microservices, API security, and external integrations

CONCISO.

Q&A – Let’s Discuss

CONCISO.

Join Our Team

CONCISO.