Token Exchange

Keycloak's Secret Weapon for Platforms
KeyConf25

Amsterdam, 28.08.2025

CONCISO.

About me

Sven-Torben Janus

Partner,
Principal Architect

sven-torben.janus@conciso.de

LinkedIn
@sventorben

CONCISO.

Basic Platform Security Model

CONCISO.

Understanding Platform Planes

Control Plane:
Orchestrates the Data Plane; manages configs, deployments, and routing of service traffic.

Management Plane:
Provides administrative control over the entire platform; used by operators, SREs, and security teams.

Data Plane:
Handles user-facing traffic and business logic; executes requests but doesn't manage infrastructure.

CONCISO.

Examples of Platform Planes

	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

Security Through Plane Separation

- **Isolated privileges**
  Each plane gets only the access it needs.

- **Reduced attack surface**
  Compromising a Data Plane service won't give control over the platform.

- **Clear trust boundaries**
  - ✅ Management → Control
  - ✅ Control → Data
  - ❌ Data → Management
  - ❌ Data → Control
  - ❌ Control → Management
  - …

- **Scoped tokens with Keycloak**
  Use Token Exchange to grant least-privilege access per plane.

CONCISO.

The Challenge of Scalable Auth

User tokens leak too far

Tokens issued to frontends often reach backend services uncontrolled

Services lack identity isolation

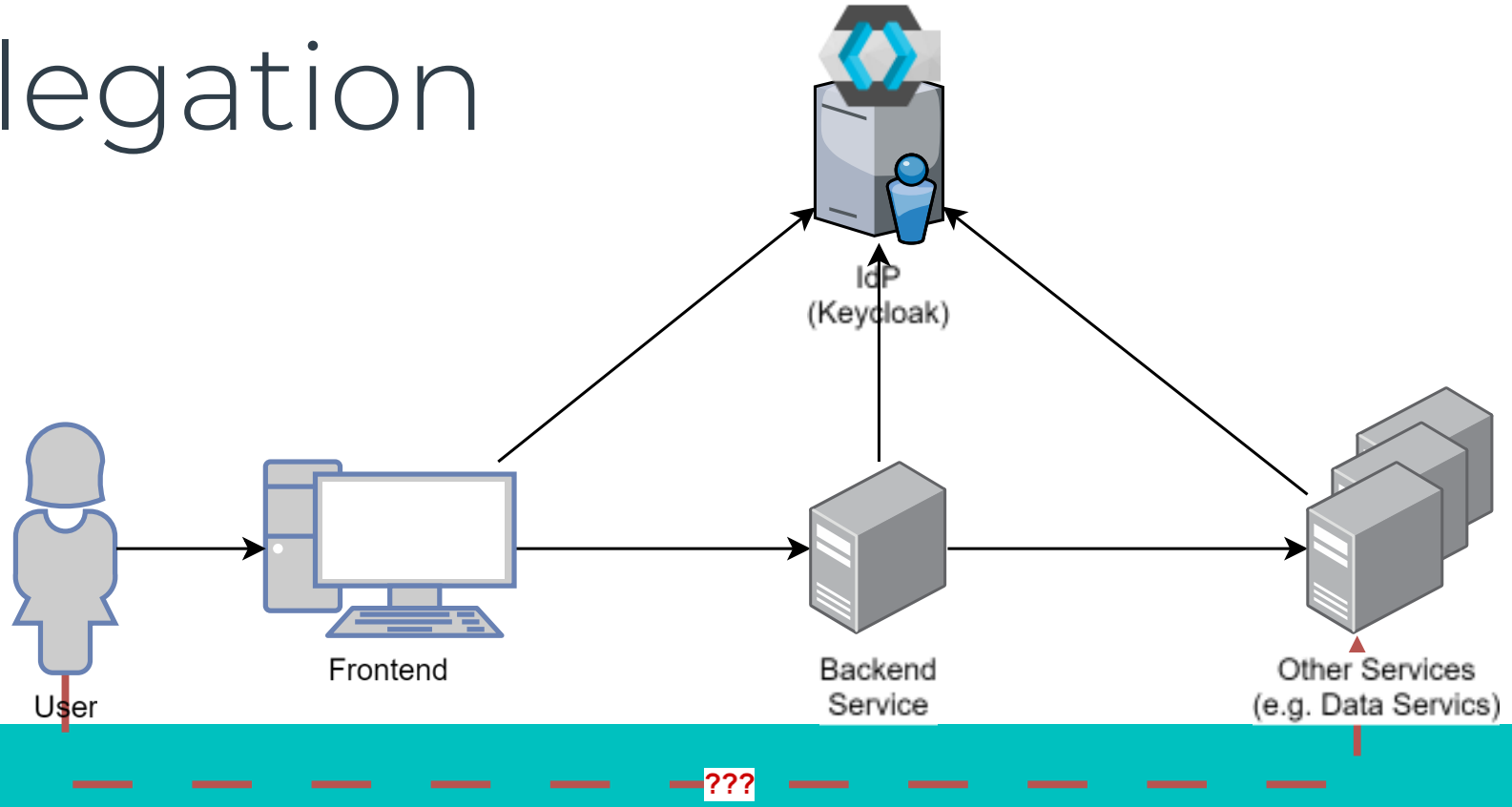
No clear separation of scopes and audiences

Uncontrolled client access

Static credentials and hardcoded secrets increase exposure

CONCISO.

Common Delegation Approaches



Stitching together methods manually

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

Using Client Credentials Grant (Service Account)

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

Forwarding the user's access token (Poor Man's Delegation)

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

CONCISO.

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","backend",
          "some-service"],
  "realm_access":{"roles":["user"]},
  "resource_access": {
    "frontend":{"roles":["user"]},
    "backend":{"roles":["admin"]},
    "some-service":{"roles":["viewer"]}
  }
}
```

Why is forwarding tokens a bad idea?

Over-privileged tokens

Token can be used across multiple backend services, even if they don't all need the same level of access.

No clear identity tracking

Difficult to distinguish by which service a request was made.

Security risks

If a single backend service is compromised, an attacker could use the forwarded token to access multiple other services.

Difficult auditing

Since the same token is passed around, logging mechanisms fail to capture true request origins.

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

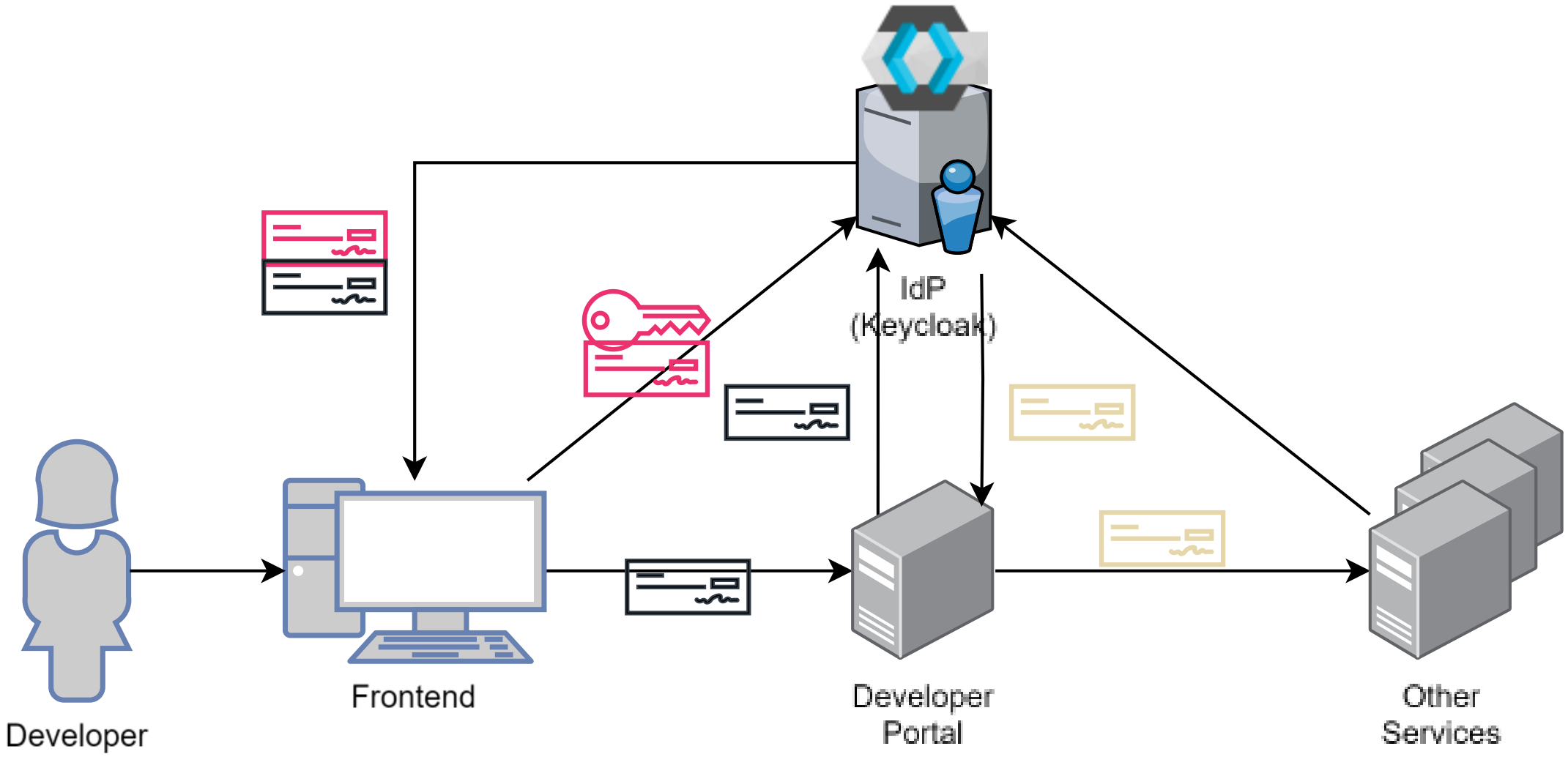
Use a centrally managed API token to call external APIs.

Impersonation & Delegation

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

CONCISO.

Token Exchange – How It Works

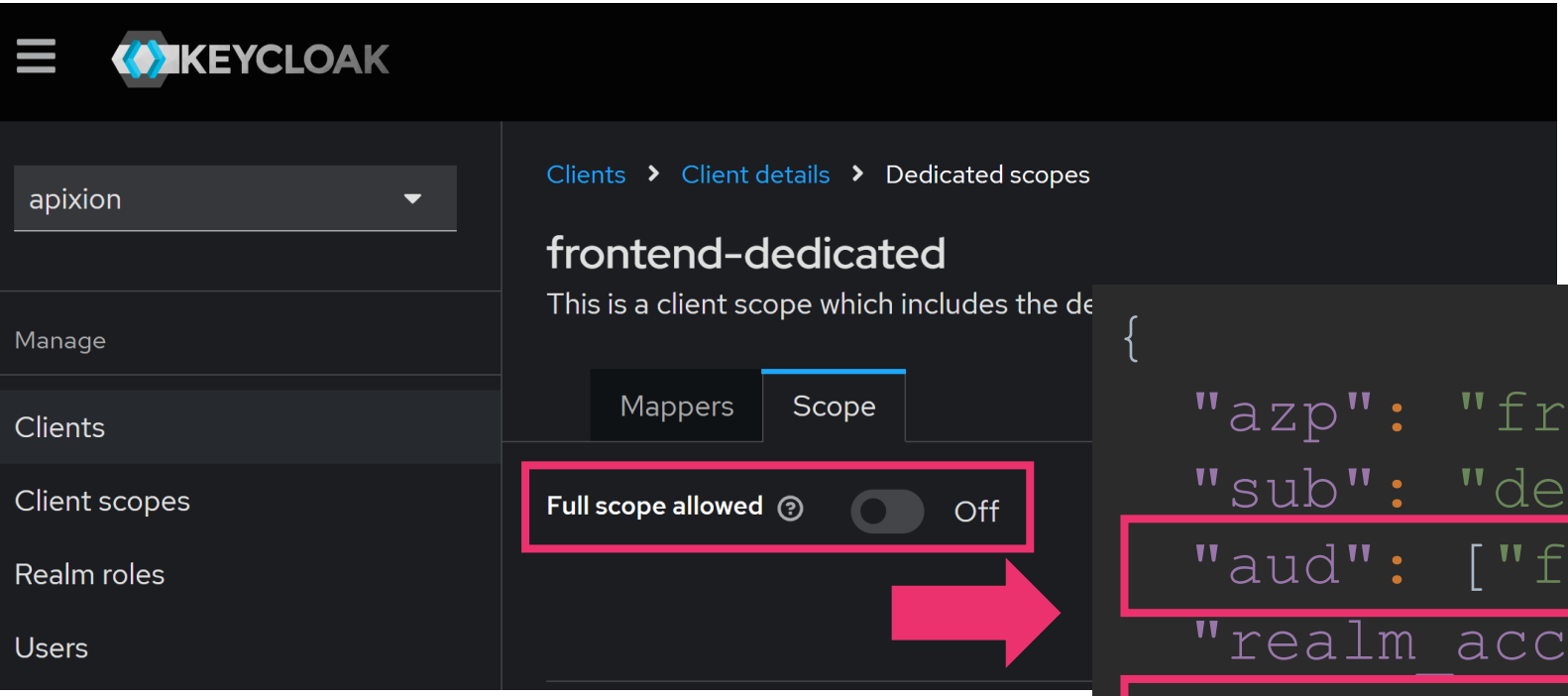


CONCISO.

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

Token Exchange Version	Configuration Requirements	Notes
Legacy Token Exchange (v1)	Must be enabled: - token-exchange - admin-fine-grained-authz (FGAP **v1**)	• Preview feature not enabled by default • quite complex to configure
Standard Token Exchange (v2)	Enabled by default since Keycloak 26.2; just enable the toggle in the client's settings	• Simplified configuration • compliant with RFC 8693 • lacks support for external tokens

Assigning Permissions (v1)

Clients > Client details

backend OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings | Keys | Credentials | Roles | Client scopes | Sessions | Permissions | Advanced | Events

Permissions

Fine grained permissions for administ… …hat want to manage this client or apply roles defined by this client.

Permissions enabled On

Permission list

Edit the permission list by clicking the scope-name. It then redirects to the permission details page of the client named **realm-management**

Scope-name	Description
configure	Reduced management permissions for administrator. Cannot set scope, template, or proto…
manage	Policies that decide if an administrator can manage this client
map-roles	Policies that decide if an administrator can map roles defined by this client
map-roles-client-scope	Policies that decide if an administrator can apply roles defined by this client to the client scope of another client
map-roles-composite	Policies that decide if an administrator can apply roles defined by this client as a composite to another role
token-exchange	Policies that decide which clients are allowed exchange tokens for a token that is targeted to this client.
view	Policies that decide if an administrator can view this client

map-roles-client-scope

map-roles-composite

token-exchange

view

CONCISO.

Assigning Permissions (v1)

Clients > Client details > Permission details

token-exchange.permission.client.31a1cf7b-0f31-4a8f-b1e3-ee24648f5f76

Name *: token-exchange.permission.client.31a1cf7b-0f31-4a8f-b1e3-ee24648f5f76

Description:

Apply to resource type: Off

Resource: 20a0916e-c0f9-4962-b403-e8c2186c5ced

Authorization scopes *: token-exchange

Policies: Control Plane Clients

Decision strategy: Affirmative / Unanimous / Consensus

Save Cancel

Resource: 20a0916e-c0f9-4962-b403-e8c

Authorization scopes *: token-exchange

Policies: Control Plane Clients

Decision strategy: Affirmative / Unanimous / Consensus

Save Cancel

CONCISO.

Assigning Permissions (v1)

Clients > Client details > Policy details

Control Plane Clients

Name *: Control Plane Clients

Description:

Client scopes *: Add client scopes

clientScopeTitle	Required field
plane:control	☑

Logic: Positive / Negative

Save Cancel

Client scopes * Add client scopes

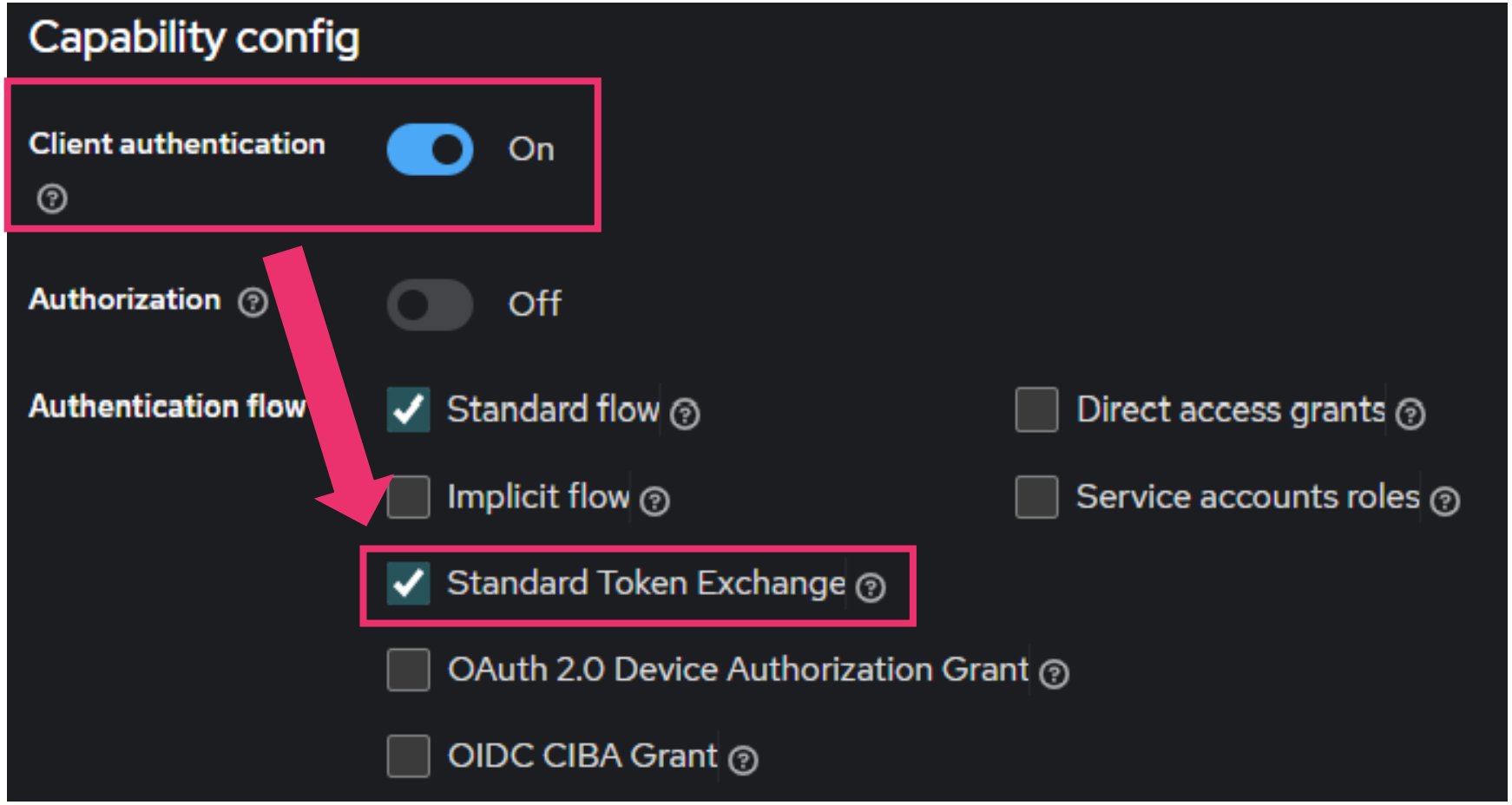
clientScopeTitle

plane:control

Logic: Positive / Negative

CONCISO.

Enable Standard Token Exchange (v2)



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"] },
  }
}
```

CONCISO.

Token Exchange v1 vs v2

Capability	**Token Exchange v1** (Preview, pre-26.2)	**Standard Token Exchange v2** (Default, 26.2+)
Feature flag required	✅ Must be enabled manually	❌ Enabled by default
Fine-grained permissions	✅ Required for security	✅ Built-in per client
Audience switching	✅ Fully flexible (internal & external)	⚠️ Limited (only **downscoping**)
Internal ↔ Internal	✅ Fully supported	✅ Fully supported
Internal → External	✅ Supported	❌ Not supported
External → Internal	✅ Supported	⚠️ Experimental, needs manual activation per feature flag
Complexity	🔹 High (AuthZ policies, mappings)	🟢 Simple (UI-based config)
Use cases	Advanced B2B, federation, SaaS	Microservices, platform-internal
Status	Deprecated	Recommended default

Observability

CONCISO.

Audit Logs

User events | Admin events

Search events

Event saved type: Token exchange × | Token exchange error ×

Refresh

1 - 14

Time	User ID	Event saved type	IP address	Client
February 28, 2025 at 11:48 PM	7c57ef4c-345c-4fab-b57f-25ddf2798d3d	TOKEN_EXCHANGE	172.18.0.1	frontend

auth_method	token_exchange
audience	backend
token_id	07d5a70e-fdd4-4f9c-b091-a50558ee13bb
grant_type	urn:ietf:params:oauth:grant-type:token-exchange
refresh_token_type	Refresh
scope	openid profile email
refresh_token_id	fa5bfec3-06c3-49c8-bc88-a35a4efda292
subject_issuer	external-partner
validation_method	signature
client_auth_method	client-secret

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 INFO [org.keycloak.events] (executor-thread-15)
    type="TOKEN_EXCHANGE" ...
```

Environment variables to visualize successful token exchanges:

```
KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_SUCCESS_LEVEL=info
KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_ERROR_LEVEL=warn
```

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.

Takeaways

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

CONCISO.

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

Q&A – Let’s Discuss

CONCISO.

Join Our Team