



# Abgesicherte OGC API Dienste

Praktische Umsetzung  
mit Keycloak bei IT.NRW

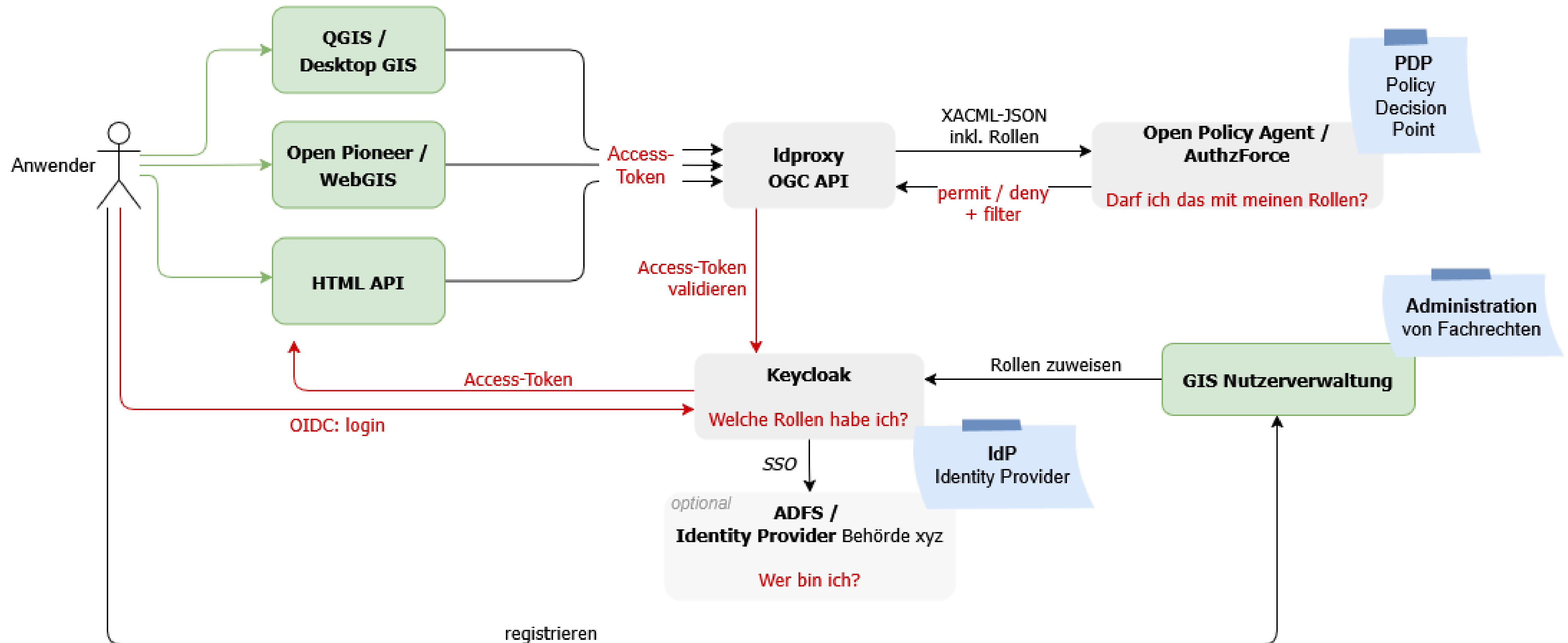
17. März 2026



# Agenda

1. Übersicht der beteiligten Komponenten
2. Grundregeln für Zugriffstoken
3. Konfiguration von Keycloak und Idproxy
4. Rollenbasierte Rechtesets im Policy Decision Point (PDP)
5. OAuth2/OIDC Konfiguration in QGIS
6. Fazit

# Infrastruktur



# Zugriffstoken

## Berechtigungen begrenzen

- **Grundregel:** Zugriffsrechte auf das notwendige Minimum beschränken.
- **Zielgruppe (Audience / aud):** Tokens explizit an definierte Ressourcenserver binden (Wo darf das Token genutzt werden?).
- **Berechtigungen (Scope / scope):** Zugriff auf spezifische Ressourcen und Aktionen (z. B. read vs. write) begrenzen.

**Fazit:** Die Sicherheit basiert auf der strikten Validierungspflicht durch den Ressourcenserver.

```
"typ": "Bearer",
"resource_access": {
  "denkmal": {
    "roles": [
      "ratingen_r",
      "ratingen_w",
      "duesseldorf_r",
      "duesseldorf_w",
      "read::denkmal",
      "data:write::denkmal"
    ]
  },
  "fiscalbo": {
    "roles": [
      "duesseldorf_w",
      "duesseldorf_r",
      "read::fiscalbo",
      "data:write::fiscalbo"
    ]
  }
},
"aud": [ "https://lv.gistest.nrw.de/ldproxy" ],
"scope": "openid email profile read", // > read::denkmal
"email_verified": true,
```

# Keycloak

## Client Scopes

- Client Scopes sind gruppierte Sammlungen von Protokoll- und Rollen-Mappings und dienen als wiederverwendbare Vorlage für die Konfiguration von Token-Inhalten.
- Ermöglicht eine einheitliche Konfiguration über die gesamte Infrastruktur.
- Jeder Client erhält präzise nur die Daten, die er für seinen Anwendungsfall benötigt.
- Änderungen am Scope wirken sich sofort auf alle verknüpften Clients aus.

### Client scopes

Client scopes are a common set of protocol mappers and roles that are shared between multiple clients. [Learn more](#)

1 - 25

| <input type="checkbox"/> | Name                 | Assigned type | Protocol       | Display order | Description  |
|--------------------------|----------------------|---------------|----------------|---------------|--|
| <input type="checkbox"/> | aud-ldproxy          | None          | OpenID Connect | -             | Zugriff auf OAPI   |
| <input type="checkbox"/> | basic                | Default       | OpenID Connect | -             | OpenID Connect scope for add all basic claims to the token                         |
| <input type="checkbox"/> | data:write::denkmal  | None          | OpenID Connect | -             | Schreibender Zugriff auf Daten von Denkmalliste NRW                                |
| <input type="checkbox"/> | data:write::fiscalbo | None          | OpenID Connect | -             | Schreibender Zugriff auf Daten von FIS Albo  |
| <input type="checkbox"/> | denkmal_r            | None          | OpenID Connect | -             | Auf Verwaltungseinheiten eingeschränkter lesender Zugriff auf Denkmalliste NRW     |
| <input type="checkbox"/> | denkmal_w            | None          | OpenID Connect | -             | Auf Verwaltungseinheiten eingeschränkter schreibender Zugriff auf Denkmalliste NRW |

# Keycloak

## Audience Mapper

- Der Audience-Mapper legt fest, für welche Empfänger (Ressourcenserver) ein Token gültig ist.
- **Statischer Wert:** Eine fest definierte URL oder Kennung
- **Client ID:** Automatische Übernahme der Client-ID als Audience

[Client scopes](#) › [Client scope details](#) › Mapper details

### Audience

e00b6f6e-81f3-4447-bc44-5815442fc64b

Mapper type

Audience

Name \* ⓘ

ldproxy-test

Included Client

Audience ⓘ

Included Custom

Audience ⓘ

https://lv.gistest.nrw.de/ldproxy

Add to ID token ⓘ

# Keycloak

## Role Scope Mapping

- Role Scope Mappings begrenzen die im Token enthaltenen Rollen.
- Das ausgestellte Token enthält nur die Rollen, die explizit für den Scope des Clients definiert wurden.
- Die Berechtigungen des Tokens können damit gezielt eingeschränkt werden.

**Achtung:** Um eine effektive Rollen-Limitierung zu erreichen, muss **Full Scope Allowed** deaktiviert werden.

The screenshot shows the 'denkmal\_r' client scope details in the Keycloak Admin Console. The 'Scope' tab is active, displaying a table of role mappings. A blue information banner at the top states: 'If there is no role scope mapping defined, each user is permitted to use this client scope. If there are role scope mappings defined, the user must be a member of at least one of the roles.' The table has columns for Name, Inherited, and Description. The 'Name' column includes a checkbox and a 'denkmal' label. The 'Inherited' column contains 'False' for all entries. The 'Description' column lists 'Radevormwald', 'Büren', 'Hilden', and 'Selfkant'. The interface includes a search bar, a 'Refresh' button, a 'Hide inherited roles' checkbox (checked), an 'Assign role' dropdown, and an 'Unassign' button. The page number '1-100' is visible at the bottom right of the table area.

| <input type="checkbox"/> | Name               | Inherited | Description  |
|--------------------------|--------------------|-----------|--------------|
| <input type="checkbox"/> | denkmal 05374036_r | False     | Radevormwald |
| <input type="checkbox"/> | denkmal 05774016_r | False     | Büren        |
| <input type="checkbox"/> | denkmal 05158016_r | False     | Hilden       |
| <input type="checkbox"/> | denkmal 05370024_r | False     | Selfkant     |

# Keycloak

## Full Scope Allowed

- Eine Konfiguration auf Client-Ebene, die steuert, welche Rollen eines Benutzers in dessen Token aufgenommen werden.
- *ON (Standardeinstellung)*: Das Token enthält alle verfügbaren Realm- und Client-Rollen des Benutzers ungeachtet der zugewiesenen Scopes.
- **OFF**: Das Token enthält nur die Rollen, die dem Client explizit über Scopes zugewiesen wurden.

[Clients](#) > [Client details](#) > Dedicated scopes

### myapp

This is a client scope which includes the dedicated mappers ar

Mappers

Scope

Full scope allowed ?



Off

```
{
  "typ": "Bearer",
  "resource_access": {
    "denkmal": {
      "roles": [
        "ratingen_r",
        "duesseldorf_r",
        "read::denkmal",
      ]
    }
  },
  "aud": [ "https://lv.gistest.nrw.de/ldproxy" ],
  "scope": "openid email profile read::denkmal denkmal_r",
  "email_verified": true,
}
```

# Idproxy

## Konfiguration

- **Öffentliche Berechtigungsgruppe:** Ohne explizite Authentifizierung bzw. entsprechende Berechtigungen ist kein Zugriff auf die API möglich (public: noaccess).
- **Audience:** Es werden nur Token akzeptiert, die spezifisch für die OAPI ausgestellt wurden.
- **Scopes:** Keine Prüfung auf spezifische Scope-Werte im Token. Die Einschränkung erfolgt über die im Token enthaltenen Rollen, deren Zuweisung im Keycloak über Client Scopes gesteuert wird.
- **Permissions:** Jede API liest nur die Rollen aus dem eigenen Namensraum aus.

```
# api.yml
accessControl:
  enabled: true
  groups:
    public: [noaccess]
  audience:
    - https://lv.gistest.nrw.de/ldproxy
  scopes: []
  policies:
    enabled: true
    attributes:
      gemeinde:
        property: gemeinde
    obligations:
      ldproxy:parameters:filter:
        parameter: filter

# cfg.yml
claims:
  permissions: resource_access.{{apiId}}.roles
```

# Policy Decision Point

## AuthzForce

- Bei lesenden Zugriffen auf Features liefert der PDP neben der Entscheidung eine Obligation (Auflage) zurück.
- Diese Auflage enthält einen spezifischen Query-Filter, um die Antwort der OAPI serverseitig auf erlaubte Features einzuschränken.

### Herausforderung: Rollen basierte Rechtesets

- Besitzt ein Nutzer mehrere Rollen, werden die einzelnen Filter-Auflagen nicht dynamisch zu einer Gesamt-Query kombiniert.
- Jede mögliche Rollen-Kombination müsste vorab statisch definiert werden.

```
<Rule RuleId="rule" Effect="Permit"><Condition><Apply FunctionId="urn:oasis:names:
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    ratingen_r
  </AttributeValue>
  <AttributeDesignator AttributeId="ldproxy:claim:permissions" Category="urn:oasis
</Apply></Condition></Rule>
<ObligationExpressions><ObligationExpression ObligationId="ldproxy:parameters" Ful
  <AttributeAssignmentExpression AttributeId="ldproxy:parameters:filter">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      gemeinde = 'Ratingen'
    </AttributeValue>
  </AttributeAssignmentExpression>
</ObligationExpression></ObligationExpressions>

<Rule RuleId="rule" Effect="Permit"><Condition><Apply FunctionId="urn:oasis:names:
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    duesseldorf_r
  </AttributeValue>
  <AttributeDesignator AttributeId="ldproxy:claim:permissions" Category="urn:oasis
</Apply></Condition></Rule>
<ObligationExpressions><ObligationExpression ObligationId="ldproxy:parameters" Ful
  <AttributeAssignmentExpression AttributeId="ldproxy:parameters:filter">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      gemeinde = 'Düsseldorf'
    </AttributeValue>
  </AttributeAssignmentExpression>
</ObligationExpression></ObligationExpressions>

→ <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  gemeinde IN ('Ratingen','Düsseldorf')
</AttributeValue>
```

# Policy Decision Point

## Open Policy Agent

- Alternative zur statischen Konfiguration der Rechtesets in XACML > Open Policy Agent.
- Rechtesets werden nicht statisch konfiguriert, sondern als Skripte programmiert.
- Filter für die OGC API werden basierend auf den aktuellen Rollen des Nutzers dynamisch erzeugt.

```
obligations := o if {  
  ldproxy.api == "denkmal"  
  ldproxy.operation in ["getItems", "getTile"]  
  
  gemeinde_cql := concat("'", "'", gemeinde_read)  
  
  filter_cql := concat("", ["gemeinde IN ('", gemeinde_cql, "')"])  
  
  o := {ldproxy.obligation_parameter("filter"): filter_cql}  
}
```

# QGIS

## Verfügbare OIDC Authentifizierungs-Methoden:

- Authorization Code Flow + PKCE
- Authorization Code Flow + Client Secret
- Implicit Flow (veraltet)
- Resource Owner Password Credentials Grant (veraltet)

Voraussetzung für alle Flows ist die Registrierung von QGIS als Client im Keycloak.

## Technischer Ablauf:

- QGIS startet für den Login kurzzeitig einen lokalen Webserver.
- Die Eingabe der Anmeldedaten erfolgt sicher im Standard-Browser direkt beim Identity Provider (IdP).
- Nach erfolgreichem Login sendet der IdP einen Authorization Code an den lokalen Webserver zurück.

The screenshot displays the configuration page for an OAuth2 client in Keycloak. The client name is 'qgis' with ID 'o102145'. The resource is set to 'Optionale URL-Quelle'. The authentication flow is 'Authorization Code Flow + PKCE'. The configuration includes the following fields:

- Beschreibung:** (empty)
- Anfrage-URL:** `http://127.0.0.1:7070/realms/GIS-TEST/protocol/openid-connect/auth`
- Token-URL:** `http://127.0.0.1:7070/realms/GIS-TEST/protocol/openid-connect/token`
- Tokenaktualisierungs-URL:** `http://127.0.0.1:7070/realms/GIS-TEST/protocol/openid-connect/token`
- Umgeleitete URL:** `127.0.0.1:7070/Optional`
- Client-ID:** `qgis`
- Bereich:** `read::denkmal denkmal_r`
- API-Schlüssel:** `Optional`

Additional settings include:

- Tokensitzung:**  Über eine Ausführung erhalten
- Zugriffsmethode:** `Kopfzeile`
- Tokenkopf:** `Voreinstellung`
- Abfragezeitüberschreitung:** `30 Sekunden`

At the bottom, there is a section for 'Zusätzliche initiale Abfrageparameter' and a note: 'Hinweise: Speicher schreibt direkt in den Authentifizierungsspeicher'.

# Fazit

- Durch etablierte Standards wie OIDC und XACML lässt sich die OAPI gut in unsere Infrastruktur integrieren.
- Ein wesentlicher Vorteil ist dabei die Unabhängigkeit von einzelnen Herstellern: Das System schreibt keine spezifische Zusatzsoftware vor, sondern setzt auf herstellerneutrale Schnittstellen.

# **Impressum**

## **Landesbetrieb Information und Technik Nordrhein-Westfalen**

Mauerstraße 51

40476 Düsseldorf

Telefon: 0211 9449-01

Telefax: 0211 9449-8000