Authorization

This chapter describes how authorization works and what decisions impact the design of a use case.

OAuth2 Scopes and Presentation Definition Mapping

Scope design

When designing a system that uses OAuth2, you have to decide how scopes map to resources that the client will attempt to access. "Resource access" is typically a specific REST-style HTTP operation on a specific URL, e.g. `POST /products/staplers/1`. Things to consider when designing scopes are discussed in this section.

Broad v.s. narrow scopes

Broad scopes are generally high level e.g., a scope that gives access to a certain use case or larger group of resources. Narrow scopes are often low-level e.g., a scope that gives read access to a specific resource, limited set of resources or operations. Examples scopes for an employee that's authorized to buy supplies for their employer:

- Very broad: `buyer`
- Broad: `buyer office` (office supplies only)
- Broad: `buyer lt-1000` (orders less than 1000 euros)
- Narrower: `buyer office:staplers` (staplers only)
- Very narrow: `buyer office:staplers:red lt-10` (red staplers only, less than 10 euros)

How scopes are mapped to operations on resources influences:

- How often clients need to request a new access token, if the previous token does not give access to a required resource.
- When access to a specific resource is authorized: when the access token is issued, or when it's used.

Broad, high-level Scopes

High-level, broad scopes typically give access to an entire use case, service, or group of resources. Checks that are executed before an access token is issued are limited to the Verifiable Credentials the client can present.

- Identification and authentication (user/client identity)
- General user access to the functionality (e.g. is admin, can buy supplies, etc.)

A real-life example of a broad scope is the Nuts eOverdracht use case, which specifies the following scopes:

- `eOverdracht-sender` which gives access to the receiver's services required by a care organization that wants to transfer a patient to another organization.
- `eOverdracht-receiver` which gives access to the sender's services to the transfer receiver.

However, when a resource is accessed, the system needs to verify that the scope gives access to the specific resource operation.

This type of scope is supported by the Nuts node.

Narrow, low-level Scopes

Narrow, low-level scopes typically give access to specific operations on specific resources, e.g., reading a specific patient's medical summary.

This type of scope is **not** supported by the Nuts node, because:

- narrow scopes often contain resource identifiers, which requires wildcards/regexes in policy mapping (more on that below), which is currently not supported by the Nuts node.
- this leads to more access tokens, since each access token has a more limited use. If user authentication involves manual input (e.g., presenting a credential using a mobile wallet), user experience will deteriorate.

Another consideration is that using low-level scopes, moves most authorization decisions to the access token issuance. This is viable and supported by the Nuts node, but complicated: it requires the vendor to implement a REST API that understands Presentation Definitions.

Policies: Mapping Scope to Authentication Subject

Due to the ongoing development of personal authentication methods and associated protocols, the Nuts node currently only supports the OAuth2 `vp_token` grant for production. User authentication via OpenID4VP is experimental and usable for production. It's still required to pass user claims within the token request if a data exchange contains PII (Personally Identifiable Information) and/or medical data (e.g., Social Security Number or EHR records)

Mapping document

This section contains an example of a presentation definition mapping document as it could be specified by a use case. The Presentation Definition is described more in detail in AuthN using Verifiable Credentials.

```
{
  "zorgtoepassing": {
    "organization": {
      "format": {
        "ldp_vc": {
```

```
    "proof_type": [
      "JsonWebSignature2020"
    ]
  },
  "ldp_vp": {
    "proof_type": [
      "JsonWebSignature2020"
    ]
  },
  "jwt_vc": {
    "alg": [
      "ES256"
    ]
  },
  "jwt_vp": {
    "alg": [
      "ES256"
    ]
  }
},
"id": "pd_any_care_organization",
"name": "Care organization",
"purpose": "Finding a care organization for authorizing access to medical metadata",
"input_descriptors": [
  {
    "id": "id_nuts_care_organization_cred",
    "constraints": {
      "fields": [
        {
          "path": [
            "$.type"
          ],
          "filter": {
            "type": "string",
            "const": "NutsOrganizationCredential"
          }
        },
        {
          "id": "organization_name",
          "path": [
```

```
            "$.credentialSubject.organization.name",
            "$.credentialSubject[0].organization.name"
          ],
          "filter": {
            "type": "string"
          }
        },
        {
          "id": "organization_city",
          "path": [
            "$.credentialSubject.organization.city",
            "$.credentialSubject[0].organization.city"
          ],
          "filter": {
            "type": "string"
          }
        }
      ]
    }
  }
 ]
}
}
```

AuthN using Verifiable Credentials

To successfully negotiate an OAuth2 access token, the token issuer (OAuth2 Authorization Server) will ask the client to present Verifiable Credentials. Nuts uses DIF Presentation Exchange for requesting and presenting credentials during authentication. It's used by the service-to-service (`vp_token bearer`) OAuth2 flow. It is also used by Discovery Services to restrict what can be registered on it.

Presentation Definition

The party requesting a presentation, typically during access token negotiation, provides a Presentation Definition to the credential wallet. The Presentation Definition specifies which credentials the wallet must provide. If the wallet can't fulfill the definition, access token negotiation will fail.

`NutsCareOrganization` example

Below is an example Presentation Definition specifying a `NutsOrganizationCredential`, not restricted to a specific issuer. It specifies the following:

- Only JSON-LD Verifiable Credentials are supported, which must be signed through `JsonWebSignature2020`
- No restrictions on the Verifiable Presentation format
- Credential type must be `NutsOrganizationCredential`
- `credentialSubject` of the credential must an object `organization` with string properties `name` and `city`.

```
{
 "format": {
  "ldp_vc": {
   "proof_type": [
    "JsonWebSignature2020"
   ]
  }
 },
 "id": "pd_any_care_organization",
 "name": "Care organization",
 "purpose": "Finding a care organization for authorizing access to medical metadata",
 "input_descriptors": [
  {
   "id": "id_nuts_care_organization_cred",
   "constraints": {
```

```
"fields": [
  {
    "path": [
      "$.type"
    ],
    "filter": {
      "type": "string",
      "const": "NutsOrganizationCredential"
    }
  },
  {
    "path": [
      "$.issuer"
    ],
    "filter": {
      "type": "string",
      "filter": {
        "type": "string"
      }
    }
  },
  {
    "id": "organization_name",
    "path": [
      "$.credentialSubject.organization.name"
    ],
    "filter": {
      "type": "string"
    }
  },
  {
    "id": "organization_city",
    "path": [
      "$.credentialSubject.organization.city"
    ],
    "filter": {
      "type": "string"
    }
  }
]
```

```
    }
   }
  ]
}
```

The identifiers used in the field constraints will be available in the token introspection result. The key will be the field `id` and the value will be the value in the credential that matches the `path`.

Authorizing Access Tokens through Presentation Exchange

The following example requires a

See the DIF Presentation Exchange specification for more information.

Credential Trust

Authentication on Nuts heavily depends on trusted credential issuers: any attribute, revelant to the security model of the use case should be verifiable. E.g., if a party claims to be a care organization, it should be able to present a Verifiable Credential to prove it. The same applies to a user presenting their name or claiming to be a care professional.

Who should be the trusted issuer for a specific Verifiable Credential depends on the context. But generally, issuers are authoritative registries (e.g. Dutch CIBG) or even state-issued (PID of natural persons).

In practice, there are the following credential issuers:

- **Governing body issuing for a specific use case**
  - In the KIK-v use case, governed by Zorginstituut Nederland, KIK-v Beheer issues to participating organizations:
    - A credential that identifies the party as participating (care?) organization, containing a Chamber of Commerce registration number.
    - Credentials that allow a participant to perform specific SPARQL queries at another participant.
- **Use case implementors issuing with explicit trust**
  - In the eOverdracht use case, implementing software vendors issue `NutsOrganizationCredential` for their clients. Software vendors explicitly trust each other.
- **Use case participant issuing with delegated trust**
  - In the eOverdracht use case, participating care organizations issue a `NutsEmployeeCredential` to their active user. It is trusted when the organization has a trusted `NutsEmployeeCredential`
  - In the Huisartsinzage, PZP, and Home monitoring use cases, parties self-issue the `X509Credential`, holding organization identity, using their CIBG UZI server certificate.
  - In the LSPxNuts use cases, CIBG UZI certificates are used as backing proof for the `HealthcareProfessionalDelegationCredential`, `HealthcareProviderCredential` and `PatientEnrollmentCredential`.

OAuth2 Flows and Wallets

Nuts supports a custom OAuth2 flows for acquiring an access token: the service-to-service flow.

Service-to-Service flow

Credentials that are presented during this flow are subject to legal organizations (e.g. registered care organizations).

This flow uses a custom grant type called `vp_token-bearer`. Presentation requests always and only target `organization` wallets. User claims can be passed as tokens. If and how the user claims correspond to the organization attestations is done by the authorization step.

The flow is secured with DPoP (optional). See "Security controls" for a detailed description.

Security controls

The following security controls are used by the OAuth2 flows:

- VP-Secured Authorization Request (Nuts RFC021) provides integrity protection and authenticity for the request.
- Demonstrating Proof of Possession (DPoP, RFC9449) provides authenticity of the client using the access token. This mitigates a MITM stealing access tokens. Usage is optional, to be enabled by the client.

Access Policy (TODO)

Anti-patterns

- **Bad**: "Clients can access /Observation, but the FHIR server has to limit it to /Observation?patient=XYZ" Requires transformation of the HTTP request at the Policy Enforcement Point.
  Better: TODO
- **Bad**: "Clients can update the FHIR resource at /Task/<XYZ> using an HTTP PUT, but only the status field. HTTP PUT is a replace operation, which requires the Policy Decision Point to verify whether delta of the update only updates the status field, which can't be performed atomically. Alternatively, it requires a use case-specific FHIR API, causing more implementation effort.
  Better: "Clients can update the status field of FHIR resource /Task/<XYZ> using an HTTP PATCH. Updates to other fields must be rejected"