Validate the access token

When another care organization queries your FHIR API (or other data API), the request carries an access token your Nuts node issued. Your API (or the API gateway in front of it) must check that token on every request, then decide whether to allow the query.

The Nuts node checks the token and tells you who is asking. Whether that party may see the data is decided by your application.

Before you start

- The node has the use case's policy definition (see chapter "Configure the node for a use case"). The policy says which credentials another party must present to get a token for a scope. Without it, the node doesn't issue tokens for that scope.
- Your care organization is registered on the Discovery Service, so others can find your authorization server (see "Register on the Discovery Service").

What an incoming request looks like

For a DPoP token, the request has two headers:

```
<HTTP method> <URL of the request>
Authorization: DPoP eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp...
DPoP: eyJhbGciOiJFUzI1NiIsInR5cCI6ImRwb3Arand0In0...
```

For a Bearer token, there's only `Authorization: Bearer <token>`.

1. Introspect the token

```
POST <internal Nuts interface>/internal/auth/v2/accesstoken/introspect
Content-Type: application/x-www-form-urlencoded

token=eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp...
```

The result has the standard OAuth2 fields, plus the fields from the use case's policy definition:

Field	What it is
`active`	Whether the token is valid
`scope`	The scope the token was issued for

Field	What it is
`iss`	The authorization server that issued the token: your node, for one of your care organizations
`client_id`	The client ID the requester gave in its token request. It isn't verified, so don't use it to identify the requester: use the fields from the credentials instead, e.g. `organization_ura`.
`exp`	When the token expires (seconds since 1970)
`cnf.jkt`	DPoP tokens only: the thumbprint of the requester's DPoP key
Other fields	Values from the credentials the requester presented, named by the field IDs in the policy definition

360-graden-clientbeeld

```
{
  "active": true,
  "iss": "https://nuts.example.com/oauth2/careorg-a-acc",
  "client_id": "https://nuts.other.example/oauth2/careorg-b-acc",
  "scope": "360-graden",
  "exp": 1793110000,
  "cnf": { "jkt": "jlkhnp87453slfhansdhf" },
  "organization_ura": "87654321",
  "organization_name": "Zorgorganisatie B",
  "organization_city": "Utrecht",
  "employee_name": "J. de Vries",
  "employee_role": "Verpleegkundige",
  "employee_identifier": "123456"
}
```

- If `active` is `false`, reject the request with `401 Unauthorized`. The token is expired, invalid or unknown.
- Check that `scope` is the scope of the use case the request belongs to.
- Check that `iss` is the authorization server of the care organization whose data is requested, e.g. `https://nuts.example.com/oauth2/careorg-a-acc`. Your node hosts several care organizations and introspects tokens for all of them, so without this check a token issued for one care organization could be used to query another.
- `cnf.jkt` is only there for DPoP tokens. You need it in the next step.

Need the presented credentials themselves, e.g. for your audit log? Use `POST <internal Nuts interface>/internal/auth/v2/accesstoken/introspect_extended`: same result, plus the Verifiable

Presentations.

2. Check the DPoP header (DPoP tokens only)

A DPoP token is only valid together with a DPoP header that the requester signed for this request:

```
POST <internal Nuts interface>/internal/auth/v2/dpop/validate
Content-Type: application/json

{
  "dpop_proof": "eyJhbGciOiJFUzI1NiIsInR5cCI6ImRwb3Arand0In0...",
  "method": "<HTTP method of the request>",
  "url": "<URL of the request>",
  "thumbprint": "jlkhnp87453slfhansdhf",
  "token": "eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp..."
}
```

Field	Value
`dpop_proof`	The `DPoP` header of the incoming request
`method`	The HTTP method of the incoming request
`url`	The URL of the incoming request, as the requester called it. Query parameters are ignored.
`thumbprint`	`cnf.jkt` from the introspection result
`token`	The access token from the `Authorization` header

The node answers with `200 OK` and whether the DPoP header is valid:

```
{
  "valid": false,
  "reason": "DPoP proof is expired"
}
```

- `valid: true`: the DPoP header belongs to this token and this request.
- `valid: false`: reject the request with `401 Unauthorized`. `reason` says why.

Use the URL as the requester called it, not as your API sees it after a reverse proxy rewrote it. Otherwise the check fails.

3. Decide whether to allow the query

The Nuts node doesn't decide whether the requester may see this data. Your application does, using the introspection result and the rules of the use case.

The use case's IG describes the rules, e.g. which legal basis is needed (such as consent registered in Mitz), and what the request must contain.

You don't have to build this from scratch: **Nuts Knooppunt** has a policy decision point (PDP). Your reverse proxy or API gateway sends it the introspection result and the request details, and gets back allow or deny. It checks things like the FHIR CapabilityStatement, required search parameters and Mitz consent. It has built-in policies per scope, e.g. for BgZ, eOverdracht and PZP. See using the PDP in the Knooppunt docs.

> **360-graden-clientbeeld**
>
> The rules are in volume 2a of the IG. In short:
>
> - The request must fit the use case's FHIR CapabilityStatement.
> - The requesting organization (`organization_ura`) must be identified, and the user (`employee_identifier`, `employee_role`) must be present for the audit log.
> - There must be a legal basis: consent registered in Mitz, the requester is in the patient's CareTeam, or another legal basis.
> - The request must be about one patient: a BSN in `/Patient/_search`, otherwise a reference to the Patient.
>
> The IG has an OPA policy that implements these rules. It isn't used in production yet, but can be a starting point. Knooppunt has no built-in 360-graden policy.

Log every request with the requester's organization and user, as the use case requires.

API reference: Auth (v2)

Revision #8
Created 2024-09-23 09:28:37 UTC by Wout Slakhorst
Updated 2026-10-27 15:27:43 UTC by Rein Krul