Check an access token

When another care organization calls your endpoint, it first requests an access token at your node (Nuts RFC003), see Request an access token. Your node checks the request and, if it's valid, issues the token. Your resource server then decides whether the token gives access to the requested data.

> **Use case design (spec v1)**
>
> The use case specifies an *access policy* for each service: which requests a token for that service allows. A request can be allowed by the token alone, or only when a NutsAuthorizationCredential in the token lists the resource. Your resource server applies this policy; the node doesn't.

Prerequisites

- Your node accepts calls on `/n2n` from other nodes, on its public interface (`:8080`). Your proxy checks the TLS client certificate; the node doesn't do TLS itself: see Recommended deployment in the Nuts node documentation. **v5:** the node can also do this itself.
- Your care organization's DID document has the service of the use case, with an `oauth` endpoint that is the full URL of your node's token endpoint, e.g. `https://<node domain>/n2n/auth/v1/accesstoken`: see Set up DIDs and services.
- Your node trusts the requester's vendor DID as issuer of NutsOrganizationCredentials: see Register, find and authorize care organizations.

1. Your node issues the token

Your node issues the token only when all of these hold:

- The JWT has a valid signature and isn't expired.
- The requester has a NutsOrganizationCredential from an issuer your node trusts.
- Your node manages the authorizer DID: your care organization.
- The JWT is meant for the `oauth` endpoint of the `service` in the authorizer's DID document.
- If there's an `identity`: the contract is valid, and its organization name and city match a NutsOrganizationCredential of the requester.
- Each NutsAuthorizationCredential is valid, not revoked, issued by the authorizer and issued to the requester.

It doesn't check:

- which data the request is for, or whether the credentials fit the `service`: that's up to your resource server;
- credentials of other types: they're passed on to the token unchecked.

2. Check the token at your resource server

For each incoming request, introspect the token (RFC7662):

```
POST <internal Nuts interface>/internal/auth/v1/accesstoken/introspect
Content-Type: application/x-www-form-urlencoded

token=<access token>
```

Deny the request if `active` is `false`. Otherwise, check:

Field	Contains	Check
`service`	The service the token was requested for	Matches the endpoint that is called
`iss`	The authorizer DID	Is the care organization whose data is requested
`sub`	The requester DID	Record in your audit log. Decide whether this care organization gets access to the requested data. A NutsAuthorizationCredential issued to it is one basis, but you may need more, such as the patient's explicit consent or a treatment relationship. What you check depends on the use case and on what the credential was issued for
`resolvedVCs`	The credentials of the request	The `purposeOfUse` fits the `service`. The requested path and operation are in `resources` of one of them, or allowed by the use case's access policy. If that resource has `userContext: true`, the token has a user
`username`, `initials`, `prefix`, `family_name`, `email`, `user_role`, `assurance_level`	The user, if the request had an `identity`	Record in your audit log; `username` identifies the user

The response can have more fields than these.

`vcs` lists the IDs of all credentials in the request. `resolvedVCs` only has those that are valid now: a credential that was revoked or expired after the token was issued is left out. So use `resolvedVCs`, not `vcs`.

A reverse proxy in front of your resource server can check a token with `HEAD <internal Nuts interface>/internal/auth/v1/accesstoken/verify` and the token in the `Authorization` header: `200` means valid, `403` invalid. The proxy calls the node's internal interface itself; never make `/internal` reachable from outside. This call doesn't return the token's content, so your resource server still introspects it for the checks above.

> **eOverdracht**
>
> A token for `eOverdracht-receiver` only allows a notification. A token for `eOverdracht-sender` only allows the resources in its credentials.
>
> The sender gets the requests for the Task and the patient data (`service` `eOverdracht-sender`). The receiver gets the notification (`service` `eOverdracht-receiver`, without credentials or user). See the table in Request an access token.
>
> All calls use mTLS, with the same kind of certificate as the nodes.

API reference: Auth (v1)

Revision #4
Created 2026-11-18 11:36:10 UTC by Rein Krul
Updated 2026-11-18 12:50:25 UTC by Rein Krul