Register, find and authorize care organizations

Spec v1 uses two kinds of credentials:

Credential	Says	Issued by	Visibility
NutsOrganizationCredential	This DID is care organization X in city Y	The vendor, with its vendor DID	Public: every node gets it
NutsAuthorizationCredential (RFC014)	Organization B may access these resources of organization A	The care organization that holds the data (A), to the organization that may access it (B)	Private: only A's and B's nodes get it

Both are sent over the Nuts network (gRPC). The examples below use eOverdracht, see the eOverdracht leveranciersspecificatie.

> **Use case design (spec v1)**
>
> The use case specifies, per role (e.g. eOverdracht sender and receiver):
>
> - the `purposeOfUse` value of its NutsAuthorizationCredentials, and the access policy that belongs to it;
> - what the credentials contain, how long they are valid and when they are revoked.
>
> The node doesn't check these rules: an authorization credential that doesn't follow the use case gets access to nothing, or breaks interoperability.
>
> Trust works differently per credential: a NutsOrganizationCredential is only accepted from issuers that the node trusts. A NutsAuthorizationCredential is issued by the care organization itself, so it is never from a trusted issuer. See Credential Trust.

1. Register your care organizations

Issue a NutsOrganizationCredential for each care organization, with your vendor DID as issuer:

```
POST <internal Nuts interface>/internal/vcr/v2/issuer/vc
```

```
{
  "issuer": "<vendor DID>",
  "type": "NutsOrganizationCredential",
  "credentialSubject": {
    "id": "<care organization DID>",
    "organization": {
      "name": "<name>",
      "city": "<city>"
    }
  },
  "visibility": "public"
}
```

Issue all of them with the same vendor DID: other vendors have to trust every issuer DID separately.

2. Trust other vendors

By default your node trusts no issuers, so a search finds no care organizations of other vendors. Trust each vendor's DID for NutsOrganizationCredential:

1. List the issuers you don't trust yet: `GET <internal Nuts interface>/internal/vcr/v2/verifier/NutsOrganizationCredential/untrusted`.
2. Check the DID with the other vendor, out-of-band, e.g. by phone. Ask them for their DID, rather than asking "is this your DID?". The contact information in a DID document isn't verified.
3. Trust it:

```
POST <internal Nuts interface>/internal/vcr/v2/verifier/trust
```

```
{
  "issuer": "<vendor DID>",
  "credentialType": "NutsOrganizationCredential"
}
```

3. Find the other care organization

Search for its NutsOrganizationCredential. The care organization's DID is in `credentialSubject.id`.

```
POST <internal Nuts interface>/internal/vcr/v2/search
```

```
{
  "query": {
    "@context": ["https://www.w3.org/2018/credentials/v1", "https://nuts.nl/credentials/v1"],
    "type": ["VerifiableCredential", "NutsOrganizationCredential"],
    "credentialSubject": {
      "organization": {
        "name": "<name>*"
      }
    }
  }
}
```

- `@context` and `type` are required, but don't filter: add at least one other field, or you get all credentials.
- A `*` at the end of a value matches everything that starts with it; `"*"` alone matches any value.

The endpoints of the organization are services in its DID document.

> **eOverdracht**
>
> The sender looks up the `eOverdracht-receiver` service in the receiver's DID document. Its `notification` endpoint is where the sender sends the notification.

4. Authorize the other organization

The care organization that holds the data issues a NutsAuthorizationCredential to the organization that may access it. Issue it with the care organization's DID, as `private`: it can contain personal data.

```
POST <internal Nuts interface>/internal/vcr/v2/issuer/vc
```

```
{
  "issuer": "<your care organization DID>",
  "type": "NutsAuthorizationCredential",
  "credentialSubject": {
    "id": "<other care organization DID>",
    "purposeOfUse": "<purpose of use>",
    "resources": [ ... ]
  },
```

```
  "expirationDate": "<date>",
  "visibility": "private"
}
```

What goes in `credentialSubject` (resources, patient) is set by the use case; see RFC014 for the fields.

> **eOverdracht**
>
> The sender issues two credentials to the receiver, both with `purposeOfUse` `eOverdracht-sender`: one for the Task (valid 1 year) and one for the patient data of the transfer. The leveranciersspecificatie lists their resources. The receiver needs no NutsAuthorizationCredential for its notification endpoint.

5. Use an authorization you received

Your node receives the credentials issued to your care organizations. Search for them with `allowUntrustedIssuer`: they're issued by care organizations, which you don't trust as issuers, so without it the search returns nothing.

```
POST <internal Nuts interface>/internal/vcr/v2/search
```

```
{
  "query": {
    "@context": ["https://www.w3.org/2018/credentials/v1", "https://nuts.nl/credentials/v1"],
    "type": ["VerifiableCredential", "NutsAuthorizationCredential"],
    "credentialSubject": {
      "id": "<your care organization DID>",
      "purposeOfUse": "<purpose of use>",
      "resources": {
        "path": "<path you want to call>"
      }
    }
  },
  "searchOptions": {
    "allowUntrustedIssuer": true
  }
}
```

Pass the credential in the access token request; see Connect your EHR in the Nuts node v5.4 documentation.

> **eOverdracht**
>
> After the notification, the receiver searches with `purposeOfUse` `eOverdracht-sender` and `resources.path` `/Task/<id>`, and uses the credential to fetch the Task.

6. Revoke an authorization

Revoke a NutsAuthorizationCredential when access must end. The revocation is sent over the Nuts network:

```
DELETE <internal Nuts interface>/internal/vcr/v2/issuer/vc/<URL-encoded credential ID>
```

> **eOverdracht**
>
> When the Task is `completed`, the sender revokes the patient data credential. The Task credential stays valid until it expires.

API reference: Verifiable Credential Registry (v2)

Revision #1
Created 2026-11-05 08:47:44 UTC by Rein Krul
Updated 2026-11-05 08:47:46 UTC by Rein Krul