Outbound: querying another party

Find another care organization, request an access token, and query its FHIR API.

- Find care organizations and their endpoints
- Request an access token
- Make the FHIR call

Find care organizations and their endpoints

Before you can query another care organization, you need its API endpoint and its authorization server URL. Find both by searching the use case's Discovery Service through your own Nuts node.

Before you start

- The node has the use case's Discovery Service definition (see chapter "Configure the node for a use case").

Search

```
GET <internal Nuts interface>/internal/discovery/v1/<service ID>?<path in the credential>=<value>
```

- `<service ID>` is the `id` field of the Discovery Service definition.
- Without query parameters, you get all registered care organizations.

Query parameters are paths in the registered credentials:

- Write the path without `$.`, e.g. `credentialSubject.subject.O`.
- Field IDs from the definition, like `organization_name`, don't work as query parameters.
- Write the value with the same upper and lower case as in the credential. Use `*` at the start or end of a value as a wildcard.
- With several query parameters, all of them must match the same credential.
- Only text fields can be searched; arrays, numbers and true/false values can't.

Which paths you can search on depends on the credentials the use case requires.

360-graden-clientbeeld

Name and city are in the X509Credential, so you can combine them:

Search by	Query parameter
Name	`credentialSubject.subject.O=Zorgorganisatie*`
City	`credentialSubject.subject.L=Utrecht`

```
GET <internal Nuts interface>/internal/discovery/v1/360-graden?credentialSubject.subject.O=Zorgorganisatie*
```

Your node searches its own copy of the Discovery Service. It updates that copy every 10 minutes by default (`discovery.client.refreshinterval`), so a new registration can take that long to show up.

The result

Part	What it is
`credential_subject_id`	The care organization's DID
`fields`	Values taken from the credentials, named by the field IDs in the Discovery Service definition
`registrationParameters`	The values the care organization registered, e.g. its API endpoints
`registrationParameters.authServerURL`	The authorization server URL. You need it to request an access token (see "Request an access token")

Which fields and registration parameters you get depends on the use case.

360-graden-clientbeeld


```
[
  {
    "id": "did:web:nuts.other.example:iam:2f1b7c3e-0d1a-4a7e-9a55-4b8f3c2e9d10#7c4e...",
    "credential_subject_id": "did:web:nuts.other.example:iam:2f1b7c3e-0d1a-4a7e-9a55-4b8f3c2e9d10",
    "fields": {
      "organization_id": "did:web:nuts.other.example:iam:2f1b7c3e-0d1a-4a7e-9a55-4b8f3c2e9d10",
      "organization_name": "Zorgorganisatie B",
      "organization_city": "Utrecht",
      "organization_ura": "87654321",
      "authorization_server_url": "https://nuts.other.example/oauth2/careorg-b-acc",
      "fhir_base_url_r4": "https://fhir.other.example/r4"
    },
    "registrationParameters": {
      "authServerURL": "https://nuts.other.example/oauth2/careorg-b-acc",
      "fhirR4BaseURL": "https://fhir.other.example/r4"
```

```
    },
    "vp": { ... }
  }
]
```

`organization_ura` is the bare URA, because the definition's pattern has a capture group.

API reference: Discovery Service

Request an access token

To query another care organization's API, you need an access token from that organization's authorization server. Your Nuts node requests it for you: it shows the authorization server the credentials the use case requires.

Before you start

- You found the other care organization and its `authServerURL` (see "Find care organizations and their endpoints").
- Your care organization's wallet has the credentials the use case's policy requires (see "Load the credential into the wallet").
- You know the scope. The use case sets it.

> **360-graden-clientbeeld**
>
> - The wallet needs the X509Credential.
> - The scope is `360-graden`.
> - Each request needs a NutsEmployeeCredential for the logged-in user (see below).

Request the token

```
POST <internal Nuts interface>/internal/auth/v2/<subject ID>/request-service-access-token
Content-Type: application/json

{
  "authorization_server": "<authServerURL of the other care organization>",
  "scope": "<scope>",
  "credentials": [ <credentials that aren't in the wallet, if the use case requires them> ]
}
```

Field	Value
`<subject ID>` (in the URL)	The subject ID of your care organization: the one making the request, e.g. `careorg-a-acc`
`authorization_server`	The other care organization's `authServerURL`, from the Discovery Service search result
`scope`	Set by the use case

Field	Value
`credentials`	Optional. Credentials to present that aren't in the wallet, typically about the user. See below
`token_type`	Optional. `DPoP` (default) or `Bearer`. Use DPoP: a stolen DPoP token can't be used without your node's key

The user's identity

Some use cases require a credential about the logged-in user, e.g. a NutsEmployeeCredential. Pass it in `credentials`. You don't sign it, and you leave out `issuer` and `credentialSubject.id`: the node fills them in with your care organization's DID. The node's signature on the whole request protects it.

360-graden-clientbeeld

A NutsEmployeeCredential is required, for the other party's audit log:

Field	Value
`name`	The user's name
`roleName`	The user's role, e.g. "Verpleegkundige"
`identifier`	The user's ID in your system

```
POST <internal Nuts interface>/internal/auth/v2/careorg-a-acc/request-service-access-token
Content-Type: application/json

{
  "authorization_server": "https://nuts.other.example/oauth2/careorg-b-acc",
  "scope": "360-graden",
  "credentials": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://nuts.nl/credentials/v1"
      ],
      "type": ["VerifiableCredential", "NutsEmployeeCredential"],
      "credentialSubject": {
        "name": "J. de Vries",
        "roleName": "Verpleegkundige",
```

```
        "identifier": "123456"
      }
    }
  ]
}
```

The response

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp...",
  "token_type": "DPoP",
  "dpop_kid": "did:web:nuts.example.com:iam:657f064a-ebef-4f0f-aa87-88ed32db3142#<key id>",
  "scope": "<scope>",
  "expires_in": 900
}
```

- Use `access_token` in the `Authorization` header when you call the other party's FHIR API (or other data API).
- For a DPoP token, you also need a DPoP header for every request, signed with the key in `dpop_kid` (see "Make the FHIR call").
- The node caches tokens: calling this API again returns the same token while it's valid, so you don't have to cache it yourself.

Errors

Response	Meaning
`400 Bad Request`	A parameter is wrong, or the authorization server returned an OAuth error
`412 Precondition Failed`	Your wallet doesn't have the credentials the policy requires
`502 Bad Gateway`	The other party's authorization server returned an error
`503 Service Unavailable`	The other party's authorization server can't be reached

API reference: Auth (v2)

Make the FHIR call

With an access token, you can query the other care organization's FHIR API (or other data API).

The token is either a DPoP token (the default) or a Bearer token, depending on `token_type` in your token request. DPoP is optional, but more secure: every request also needs a DPoP header, which proves the request comes from you, the party the token was issued to. A stolen DPoP token is useless without your node's key. Your Nuts node creates the DPoP header for you.

Before you start

- You have an access token and its `dpop_kid` (see "Request an access token").
- You have the other care organization's API endpoint, from its `registrationParameters` (see "Find care organizations and their endpoints").
- Your client can connect to the other party's API. Data endpoints are protected with mTLS (see "Endpoints and TLS certificates").

1. Create the DPoP header (DPoP tokens only)

Create a new DPoP header for **every** request. It's only valid for one HTTP method and URL, and can only be used once.

```
POST <internal Nuts interface>/internal/auth/v2/dpop/<dpop_kid, URL-encoded>
Content-Type: application/json

{
  "htm": "<HTTP method>",
  "htu": "<URL of the request>",
  "token": "<access token>"
}
```

Field	Value
`<dpop_kid>` (in the URL)	`dpop_kid` from the token response, URL-encoded once, e.g. `#` becomes `%23`
`htm`	The HTTP method of the request you're about to make, e.g. `GET`
`htu`	The URL of the request you're about to make. Query parameters are ignored.
`token`	The access token

The response:

```
{
  "dpop": "eyJhbGciOiJFUzI1NiIsInR5cCI6ImRwb3Arand0In0..."
}
```

2. Make the request

Send the access token in the `Authorization` header with the type `DPoP`, and the DPoP header in `DPoP`:

```
<HTTP method> <URL of the request>
Authorization: DPoP <access token>
DPoP: <dpop from step 1>
```

For a Bearer token, skip step 1 and send `Authorization: Bearer <access token>`.

Errors

Response from the other party	What to do
`401 Unauthorized`	The token expired, or the DPoP header doesn't match the request. Check `htm` and `htu`. If the token expired, request a new one. If it hasn't expired, the other party may have lost it: request a new token with the header `Cache-Control: no-cache`, which skips your node's token cache.
`403 Forbidden`	The other party doesn't allow this query, e.g. no legal basis such as consent. Check the use case's rules.

360-graden-clientbeeld

Use `fhirR4BaseURL` or `fhirStu3BaseURL` from the search result, depending on which FHIR version the other party offers. The examples below use R4, with base URL `https://fhir.other.example/r4`.

1. Get the CapabilityStatement. It says which queries the other party supports. It doesn't need an access token, and you need it before any query other than Patient.

```
GET https://fhir.other.example/r4/metadata
```

2. Find the patient by BSN. Only POST-based search is allowed. Create the DPoP header with `htm` `POST` and `htu` `https://fhir.other.example/r4/Patient/_search`.

```
POST https://fhir.other.example/r4/Patient/_search
Content-Type: application/x-www-form-urlencoded
Authorization: DPoP <access token>
DPoP: <dpop>

identifier=http://fhir.nl/fhir/NamingSystem/bsn|123456789
```

The result has the patient's ID at the other party, e.g. `Patient/abc123`.

3. Query the data. For example, the patient's allergies. Create a new DPoP header with `htm` `GET` and `htu` `https://fhir.other.example/r4/AllergyIntolerance`.

```
GET
https://fhir.other.example/r4/AllergyIntolerance?patient=abc123&_profile=http://nictiz.nl/fhir/StructureDefinition/nl-core-AllergyIntolerance,http://nictiz.nl/fhir/StructureDefinition/zib-AllergyIntolerance
Authorization: DPoP <access token>
DPoP: <dpop>
```

- Queries select resources by FHIR profile (`_profile`). Volume 3 of the IG lists the query for each zib and FHIR version.
- Results are paged: use `_count` (maximum 100) and follow the bundle's `next` link.
- The order of requests is in volume 2b of the IG.

API reference: Auth (v2)