Data exchange using Nuts

This chapter describes how to perform inbound or outbound data exchanges using Nuts. It assumes you've deployed and configured your Nuts node in the preceding chapters.

Requesting access (outbound)

To access APIs secured through Nuts, callers need an access token issued by the OAuth2 Authorization Server of the API owner. This page describes how to acquire an access token.

Requesting Service Access Token

This section describes which value(s) need to be specified in the service access token request.

- In the request URL:
  - `subjectID`: the ID of the local requester, which was provided by the Nuts node when the subject and its DIDs was created.
- In the request body:
  - `authorization_server`: the OAuth2 issuer URL of the party that grants access, found in the service discovery search result.
  - `scope`: specifies what resources the access token will give access to. This is specified by the use case.
  - `credentials` (optional): one or more credentials to provide to the authorization server that are not in the requester's wallet. This is typically used to provide an `NutsEmployeeCredential` to the authorization server. See the section below for how to provide this.
  - `token_type` (optional): by default, tokens are of type DPoP that mitigate token theft. Alternatively, the `Bearer` token type can be specified, but you'll be more vulnerable to MITM attacks.

Example

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


{
  "authorization_server": "https://example.com/oauth2/hospital_x",
  "scope": "eOverdracht-sender"
}
```

Providing additional credentials

The service access token request allows you to supply credentials to the request, that are not in the subject's wallet but required for authentication. For instance, an `NutsEmployeeCredential` that contains information about the current logged-in user for logging purposes. These credential don't need to be signed: in that case they will be "self-attested" (e.g., the `NutsEmployeeCredential`); the Verifiable Presentation's signature will provide authenticity.

Example

The example below shows an example access token request with an `NutsEmployeeCredential`.

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

{
  "authorization_server": "https://example.com/oauth2/hospital_x",
  "scope": "eOverdracht-sender",
  "credentials": [
     {
       "@context": [
         "https://www.w3.org/2018/credentials/v1",
         "https://nuts.nl/credentials/v1"
       ],
       "type": ["VerifiableCredential", "NutsEmployeeCredential"],
       "credentialSubject": {
         "name": "John Doe",
         "roleName": "Nurse",
         "identifier": "123456"
       }
     }
   ]
}
```

Authorizing access (inbound)

TODO

Discovery of organizations and endpoints

Discovery of Organizations and Endpoints

Once organizations have activated themselves for a use case (covered in previous chapters), they become discoverable. This chapter focuses exclusively on how clients find organizations and their service endpoints using the Service Discovery API:

```
GET <internal Nuts interface>/internal/discovery/v1/{serviceID}
```

This API returns all organizations registered for a given service, optionally filtered by search parameters. It is the primary mechanism for discovering:

- Which organizations participate in your use case
- What endpoints they expose (FHIR URL, OAuth server, etc.)

Example

For instance, to search the "eOverdracht" service for a care organization that contains "Thuiszorg" in its name, perform the following HTTP query:

```
GET <internal Nuts
interface>/internal/discovery/v1/eOverdracht?credentialSubject.organization.name=*Thuiszorg*
```

Could yield (some fields omitted for brevity):

```
[
 {
  "credential_subject_id": "did:web:example.com",
  "fields": {
    "organization_name": "Thuiszorg de Zonnebloem"
  },
  "id": "did:web:example.com#1",
  "registrationParameters": {
   "fhirBaseURL": "https://example.com/fhir"
  },
  "vp": {
   // etc
```

```
    }
  }
]
```

What `fields` and `registrationParameters` are returned depends on the use case. Review the use case's Discovery Service's Presentation Definition for more information.