Authenticate users

In spec v1, the user signs a *login contract*: a statement that they act on behalf of a care organization, valid for a limited time. The signed contract is a Verifiable Presentation. You send it as `identity` in an access token request, see Request an access token, so the other party knows which user requests the data. The contract identifies the user to other parties, and says for which care organization they work. You can also use the signed contract as the user's session token in your EHR.

> **Use case design (spec v1)**
>
> The use case specifies whether an access token request needs a user identity. Requests that return personal or medical data normally need one.

The user signs with the *employee identity* means (`employeeid`): the care organization states who its employee is (identifier, name, role), and the user confirms the contract in a browser. In the access token, `username` is the identifier you pass and `assurance_level` is `low`.

The node also supports IRMA (Yivi) as means, but it's no longer used and not described here.

Prerequisites

The user's browser talks to the Nuts node directly. So:

- the APIs of the node that start with `/public`, on its public interface (`:8080`), must be reachable over HTTPS from the internet, without extra access restrictions;
- `url` must be set to the URL they're reachable on:

```
url: https://<public domain>
```

v5: the setting is `auth.publicurl`.

How it fits in your application

The user's browser can't call the node's internal API. Your backend does that, and offers your frontend two endpoints:

Your backend endpoint	Does
Start signing	Draws up the contract (step 1), starts the session (step 2), returns `sessionPtr.url` and the session ID

Your backend endpoint	Does
Get the result	Asks the node for the session (step 3), returns the status. When signing is done, it stores the signed contract in the user's session

Your frontend shows the page at `sessionPtr.url`, then asks for the result until signing is done. Protect both endpoints with the user's EHR session. Pass errors on to the frontend, so the user knows what went wrong.

1. Draw up a contract

```
PUT <internal Nuts interface>/internal/auth/v1/contract/drawup
```

```
{
  "type": "PractitionerLogin",
  "language": "EN",
  "version": "v3",
  "legalEntity": "<care organization DID>",
  "validFrom": "2026-01-02T15:04:05+01:00",
  "validDuration": "2h"
}
```

- `type` / `language`: `BehandelaarLogin` with `NL`, or `PractitionerLogin` with `EN`.
- `version`: `v3`.
- `legalEntity`: the care organization the user acts for. Your node must manage its DID, and it needs a NutsOrganizationCredential, see Register, find and authorize care organizations: its name and city are in the contract text. The user picks the care organization at login, or is already logged in for it.
- `validFrom`: RFC3339 time, optional (default: now).
- `validDuration`: e.g. `2h` or `60m`, optional (default: 60 minutes).

If the care organization has NutsOrganizationCredentials with different names or cities, the request fails. Then pass the one to use as `organizationCredential`.

The response has the `type`, `language` and `version` of the contract, and the contract text in `message`. The dates in the text are in Dutch time (`Europe/Amsterdam`). E.g.:

```
EN:PractitionerLogin:v3 I hereby declare to act on behalf of CareBears located in CareTown.
This declaration is valid from Monday, 2 January 2026 15:04:05 until Monday, 2 January 2026
17:04:05.
```

2. Sign the contract

The user needs a recent browser with JavaScript. Start a signing session with the contract text as `payload`:

```
POST <internal Nuts interface>/internal/auth/v1/signature/session
```

```
{
  "means": "employeeid",
  "payload": "<message>",
  "params": {
    "employer": "<care organization DID>",
    "employee": {
      "identifier": "<unique user ID, e.g. email address or employee number>",
      "initials": "<initials>",
      "familyName": "<family name>",
      "roleName": "<role>"
    }
  }
}
```

All fields are required, except `roleName`: the user's role in the care organization. That is a job title, or the role you use for access control if people can read it.

The response:

```
{
  "sessionID": "<session ID>",
  "means": "employeeid",
  "sessionPtr": {
    "sessionID": "<session ID>",
    "url": "https://<public domain>/public/auth/v1/means/employeeid/<session ID>"
  }
}
```

Show the page at `sessionPtr.url` to the user (iframe, pop-up, or a browser window inside a native app), where they confirm the contract. The user has 10 minutes to do so. Make sure the user sees the page, e.g. bring it to the front. The page can be opened once: after a reload it no longer shows the contract. Then start a new session.

3. Get the signed contract

The endpoint is internal, so your frontend can't call it: it asks your backend. Poll the session from your backend:

```
GET <internal Nuts interface>/internal/auth/v1/signature/session/<sessionID>
```

```
{
  "status": "completed",
  "verifiablePresentation": { ... }
}
```

- `status`: `created`, `in-progress`, `completed`, `cancelled`, `errored` or `expired`. Use it to show progress to the user.
- `verifiablePresentation`: the signed contract. It's only there when signing succeeded. Store it in the user's session.

You get the signed contract once: only the first response with `completed` contains it. After that, the node removes the session: later calls don't return the contract and end in `404`. A cancelled, errored or expired session is also removed once you've read its status.

4. Check the signed contract before using it

The contract expires (`validDuration`). Check it before you send it:

```
PUT <internal Nuts interface>/internal/auth/v1/signature/verify
```

```
{
  "VerifiablePresentation": <signed contract>
}
```

Add `checkTime` (RFC3339) to check at another moment than now. `validity` is `false` when the contract is invalid or expired. Then let the user sign a new one.

The response for a valid contract:

```
{
  "validity": true,
  "vpType": "NutsSelfSignedPresentation",
  "issuerAttributes": {
    "username": "<identifier>",
    "initials": "<initials>",
    "familyname": "<family name>",
    "user_role": "<role>",
    "assurance_level": "low"
  },
```

```
  "credentials": {
    "legal_entity": "<organization name>",
    "legal_entity_city": "<organization city>",
    "valid_from": "<start, as in the contract text>",
    "valid_to": "<end, as in the contract text>"
  }
}
```

API reference: Auth (v1)

Revision #4
Created 2026-11-17 13:07:03 UTC by Rein Krul
Updated 2026-11-18 12:50:15 UTC by Rein Krul