Authenticatie

Hoe stellen we de identiteit van een gebruiker / persoon vast binnen het Nuts netwerk?

Voor het totaalbeeld van authenticatie, autorisatie en accountability, zie Toegangscontrole met Verifiable Credentials.

- Proposal for EmployeeIdentity means
- UZI certificates as a source of identity

Proposal for EmployeeIdentity means

> **Status: historical.** Kept for reference and no longer maintained. This proposal is specified in RFC019 Employee Identity Authentication Means and RFC020 Authorization credential extension, and implemented in the Nuts node as means `employeeid`, for spec v1 (did:nuts). This page is the original proposal. Where it differs from the RFCs (e.g. field names, request and response examples), the RFCs apply.

In this document we propose a new *means* to identify a user. The identity is issued by the employer about the employee, hence the name `EmployeeIdentity`. The goal of this means is to reuse the information form the current session the user has within the application. This will result is a smoother user experience with the cost of less trust in the correctness of the information.

The proposal was first discussed in a pull request on RFC002, which was closed; the final text is in RFC019 and RFC020.

Principles

The proposed solution is based on these principles:

1. Minimal impact on the development capacity of vendors. We know there is still a lot to do for eOverdracht, and that the deadline is close. We propose a solution that vendors can ideally build in one day.
2. Allow a growth path. We know that personal authentication means will soon become mandatory through laws such as the WDO and the Wegiz, combined with several NEN standards. The solution lets care organizations and vendors raise the level easily.
3. Comply with NEN 7513. To comply with this standard, it is important to know which user viewed which data. The proposed solution meets this requirement.

Proposed solution

Assurance level requirement

The care organization must be able to decide which trust level it requires of the means used. That is why we propose a new field `authenticationAssuranceLevel` in the `AuthorizationCredential`, which states the assurance level of the means used.

The Nuts agreement framework then defines a list of levels and means, for example:

means	level
EmployeeIdentity	low
Yivi(IRMA)	middle/substantial
Uzi	high

The data holder sets this level per resource. The requesting party requests the user identity at the required level.

Example

The object below is an example of a `credentialSubject` of an `AuthorizationCredential` that uses the `authenticationAssuranceLevel` field:

```
{
  "id": "did:nuts:SjkuVHVqZndMVVJwcnUzbjhuZklhODB1M1M0LW9LcWY0WUs5S2",
  "legalBase": {
    "consentType": "explicit",
    "evidence": {
      "path": "pdf/f2aeec97-fc0d-42bf-8ca7-0548192d4231",
      "type": "application/pdf"
    }
  },
  "localParameters": {...},
  "resources": [
    {
      "path": "/DocumentReference/f2aeec97-fc0d-42bf-8ca7-0548192d4231",
      "operations": ["read"],
      "userContext": true,
      "authenticationAssuranceLevel": "low"
    }
  ],
  "purposeOfUse": "test-service",
  "subject": "urn:oid:2.16.840.1.113883.2.4.6.3:123456780"
}
```

Introducing a new EmployeeIdentity means

The problem we try to solve is the implementation, acceptance and rollout of personal authentication means. We expect care organizations to need more time to set up internal policy, support and rollout for these means. The choice of means is also still limited, so they don't fit the work processes of care organizations yet. Care organizations say they want to fall back on existing agreements between them, and trust each other to identify their employees. That is why we introduce the `EmployeeIdentity` means. This means has level `low` and uses the identity of the logged-in user.

This means is a drop-in replacement for the Yivi (IRMA) means.

Create signing session

Current payload for the IRMA means:

```
POST /internal/auth/v1/signature/session
{
  "means": "irma",
  "payload": "EN:PractitionerLogin:v3 I hereby declare to act on behalf of CareBears located in CareTown. This declaration is valid from Monday, 2 January 2006 15:04:05 until Monday, 2 January 2006 17:04:05."
}
```

Proposed payload for the EmployeeIdentity means:

```
POST /internal/auth/v1/signature/session
{
  "means": "employeeIdentity",
  "params": {
    "employer":"did:123",
    "emplopyee": {
      "identifier": "481",
      "roleName": "Verpleegkundige niveau 2",
      "initials": "J",
      "familyName": "van Dijk",
      "email": "j.vandijk@example.com"
    }
  },
  "payload": "EN:PractitionerLogin:v3 I hereby declare to act on behalf of CareBears located in CareTown. This declaration is valid from Monday, 2 January 2006 15:04:05 until Monday, 2 January 2006 17:04:05."
}
```

Response:

IRMA:

```
{
  "sessionID": "123",
  "sessionPtr": {QRCode},
  "means": "irma"
}
```

The frontend renders the contents of the `sessionPtr` as a QRCode.

EmployeeIdentity:

```
{
  "sessionID": "123",
  "sessionPtr": { "url":"nuts.example.com/public/auth/employeeID/123" },
  "means": "irma"
}
```

The frontend uses the `url` from the `sessionPtr` object to open a new browser window or render the page in a iframe.

Polling for the session status

IRMA and EmployeeIdentity:

`GET /internal/auth/v1/signature/session/{sessionID}`

Pending, as long as the user is signing the contract:

```
{
    "status": "pending"
}
```

Completed:

```
{
  "status": "completed",
  "verifiablePresentation": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
```

```
      "https://nuts.nl/credentials/v1",
      "http://schema.org/",
      "https://w3c-ccg.github.io/lds-jws2020/contexts/lds-jws2020-v1.json"
    ],
    "type": ["VerifiablePresentation", "NutsSelfSignedPresentation"],
    "verifiableCredential": [
      {
      "issuer": "did:nuts:careOrg",
        "type": ["VerifiableCredential", "NutsEmployeeCredential"],
        "expirationDate": "2023-04-03T20:34:17.687862+01:00",
        "credentialSubject": {
          "type":"Organization",
          "id":"did:nuts:123456789",
          "member":{
            "type":"EmployeeRole",
            "identifier": "481",
            "roleName": "Verpleegkundige niveau 2",
            "member": {
              "type": "Person",
              "initials": "J",
              "familyName": "van Dijk",
              "email": "j.vandijk@example.com"
            }
          }
        }],
      "proof": {
        "type": "JsonWebSignature2020",
        "verificationMethod": "did:nuts:123456789#key-1",
        "created": "2023-04-03T16:34:17.687862+01:00",
        "jws":
"eyJhbGciOiJFUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..hKcboC8m6YnZPi6ReJAYs0J0Ztn5nxcx2Ea
voXdtrkWxmE1JZmImW89_8IIgjvfI8XtGeDlEnGywAuY2u7y9Bw"
      }
    },
    "proof": {
      "challenge": "LOGIN CONTRACT",
      "type": "JsonWebSignature2020",
      "verificationMethod": "did:nuts:123456789#key-1",
      "created": "2023-04-03T16:34:17.687862+01:00",
      "jws":
```

```
"eyJhbGciOiJFUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..hKcboC8m6YnZPi6ReJAYs0J0Ztn5nxcx2Ea
voXdtrkWxmE1JZmImW89_8IIgjvfI8XtGeDlEnGywAuY2u7y9Bw"
    }
  }
}
```

Usage of the identity token

The usage of the returned identity is the same for IRMA and EmployeeIdentity: the value of the `verifiablePresentation` field is base64 encoded and added to the access token request in the `usi` field.

Status

Implemented in Nuts node v5.3.0, as specified in RFC019 and RFC020. The open points of the proposal were settled as follows:

Item	Result
Remove evidence from the presentation	Done
Add the new types to the Nuts JSON-LD context	Done: `NutsEmployeeCredential` and `NutsSelfSignedPresentation`
RFC for `NutsSelfSignedPresentation`	Done: part of RFC019
Backwards compatibility of the JSON-LD context for the new field	Resolved: the field is `assuranceLevel` and is part of the Nuts JSON-LD context (RFC020)
Proof of ownership of the credential	The presentation is signed by the issuer (the care organization), which must also be the credential subject. A verifier trusts the `NutsEmployeeCredential` if its issuer has a trusted `NutsOrganizationCredential`; that credential is not added to the presentation.

UZI certificates as a source of identity

UZI certificates from CIBG can back Verifiable Credentials. The credential is signed with the UZI private key, and its issuer is a `did:x509` DID derived from the certificate. This reuses CIBG's identity checks instead of setting up new issuers. It breaks a chicken-and-egg problem: wallets have no value without credentials, and issuers wait until wallets are in use.

Credential definitions are in the credential catalog (normative). For the server-certificate credential used in production, including examples, see X509Credential with UZI server certificates.

Principles

- **Reuse CIBG's checks.** CIBG checks organizations (URA) and persons (UZI number, role) before it issues a certificate or card. A credential backed by that certificate needs no new issuer.
- **The key holder vouches for the claim.** Whoever holds the private key makes the claim: the care organization (or its vendor) for a server certificate, the person for a smartcard.
- **Binding.** Credentials are bound together by an identifier they share, such as the URA. Binding works within one Verifiable Presentation, or across presentations that belong together, e.g. one from the vendor and one from the care organization. All credentials are issued to the care organization's `did:web`; a server-certificate credential binds its URA to that `did:web`.
- **Trust the issuing UZI CA, check the pass type.** CIBG has a separate CA per pass type (UZI-register Zorgverlener CA, Medewerker op naam CA, Medewerker niet op naam CA, Server CA). A verifier trusts the CA that issues the pass type it accepts, and checks the pass type in `san.otherName`.

UZI types

UZI type	Pass type	Key held by	Proves	Credential	Status
Server certificate	`S`	Care organization, often managed by the vendor	The care organization and its URA	X509Credential; HealthcareProviderCredential (next version)	Production; draft
Zorgverlenerspas	`Z`	Care professional, on a smartcard	A named care professional, with UZI number and role	HealthcareProfessionalDelegationCredential, PatientEnrollmentCredential	Draft, LSPxNuts pilot

UZI type	Pass type	Key held by	Proves	Credential	Status
Medewerkerspas op naam	`N`	Named employee, on a smartcard	A named employee of the care organization	PatientEnrollmentCredential	Draft, LSPxNuts pilot
Medewerkerspas niet op naam	`M`	Employee, on a smartcard	Only that the holder works for the care organization	None: the catalog's credentials reject this pass type	-

Pass type codes as defined by CIBG, see e.g. the UZI reader of the Ministry of VWS.

How the credentials fit together

The URA binds the credentials together. Example from LSPxNuts, all presented by the care organization:

Credential	Signed with	States
HealthcareProviderCredential	Server certificate (`S`)	This URA belongs to this `did:web`
HealthcareProfessionalDelegationCredential	Zorgverlenerspas (`Z`)	This professional delegates these actions to this URA
PatientEnrollmentCredential	Zorgverlenerspas (`Z`) or medewerkerspas op naam (`N`)	This patient is enrolled at this URA

The verifier checks:

1. each credential's issuing CA and pass type;
2. that the `did:web` in the HealthcareProviderCredential is the presenter;
3. that all credentials name the same URA.

Issuance

UZI type	How the credential is created
Server certificate	With the certificate's key file, no smartcard needed: with the (Go) go-didx509-toolkit or the Java library. Done by someone at the care organization or its vendor, or automated.
Smartcard	The card holder signs with card and PIN, on a workstation with a card reader. In LSPxNuts this runs through the ZORG-ID SDK; see lspxnuts-pilots.

Governance

Decision	Who	Where
Which credential types exist, and which pass types and CAs they accept	Credential catalog	Credential catalog
Which credentials a use case requires	Use case owner	Discovery Service and policy definitions
Who holds the server certificate's private key	Care organization, or vendor by agreement	Contract between them
Who signs a person-backed credential	The card holder, personally	-
Who gets a certificate or card	CIBG	UZI register

Limits and dependencies

- **Server certificate key held by the vendor:** ideally only the care organization holds the private key of its server certificate. In practice the vendor often holds it, because the same certificate is used for TLS client authentication to other services. The vendor can then create the care organization's credential. This is why Nuts wants to move towards credentials that aren't backed by a TLS certificate (see Outlook).
- **CIBG and PKIoverheid:** a CA change affects every definition that trusts it, e.g. G1 to G4 for server certificates.
- **Workstations:** smartcards need a card reader and software on every workstation where people sign.
- **URA in a combined string:** credentials based on X.509 fields need a pattern to extract the URA from `san.otherName`. The HealthcareProviderCredential contains the URA as a separate field.
- **Personal data:** person-backed credentials contain a name and UZI number.

Outlook

- **Credentials not backed by a TLS certificate:** trust service providers (TSPs) could issue care organization credentials directly, so the organization's identity no longer depends on a key that is also used for TLS.
- **Dezi:** the credential catalog also has a DeziUserCredential, which identifies healthcare workers through Dezi. Whether Dezi will replace UZI cards for this is still open.
- **GIS-VN:** the catalog names future GIS-VN intermediate CAs as trust anchors for personal passes.