Get an X509Credential from a UZI certificate

Many use cases identify care organizations by their URA number. The CIBG doesn't issue Verifiable Credentials with the URA yet, so you create one yourself from the care organization's UZI server certificate: an X509Credential.

The node uses the X509Credential to:

- register the care organization on the Discovery Service, so other parties can find it;
- prove the care organization's identity when it requests an access token from another party.

The X509Credential is:

- **issued by** the UZI certificate: the issuer is a did:x509 DID derived from the certificate, and the credential is signed with the certificate's private key;
- **issued to** the care organization's did:web DID, which belongs to its subject on your Nuts node. That subject holds the credential in its wallet.

It contains the UZI certificate, and the URA, name and city taken from it. It does **not** contain the private key.

For background, such as how the URA ends up in the credential and which certificate authority a use case trusts, see X509Credential with UZI server certificates. The credential's definition: X509Credential in the credential catalog.

Do this once per care organization, per environment.

Who does what

The UZI certificate and its private key belong to the care organization. Ideally, the private key never leaves the care organization:

Step	Who
Create the subject and give its did:web DID to the care organization	Vendor
Create the X509Credential with the UZI certificate and private key	Care organization
Send the X509Credential (a JWT, without the private key) to the vendor	Care organization

Step	Who
Load the X509Credential into the wallet	Vendor

In practice, the vendor often already manages the care organization's UZI certificate, e.g. for client TLS connections. The vendor then has the private key anyway, and can create the X509Credential itself. The steps below work for the care organization and the vendor alike.

Before you start

- The care organization has a subject, and you have its did:web DID from the vendor (see "Create a subject").
- The care organization's UZI server certificate from the CIBG:
  - the full certificate chain (root CA up to the certificate itself) as a PEM file;
  - the unencrypted private key of the certificate as a PEM file.
- A tool to create the credential. Choose one:
  - The `nutsfoundation/go-didx509-toolkit` Docker image (source), to create the credential by hand. The steps below use this tool.
  - The Java library uzi-did-x509-issuer-java, to create the credential from your EHR or another Java application, without Docker. See its README for usage.

Create the credential

Put the certificate chain (`certificate.pem`) and private key (`privatekey.pem`) in a directory called `certs`. From the directory that contains `certs`, run:

```
docker run --rm -v "$(pwd)/certs:/certs" \
  nutsfoundation/go-didx509-toolkit:1.2.0 \
  vc /certs/certificate.pem /certs/privatekey.pem \
  "<CA name>" \
  <did>
```

- `<did>` is the care organization's did:web DID, from "Create a subject".
- `<CA name>` is the full name (subject DN) of the CIBG certificate authority that issued the UZI server certificate. It depends on the type of certificate:

Certificate	CA (`CN` part of the name)
Production, G1	`UZI-register Private Server CA G1`
Production, G4 (from 12 November 2026)	`UZI Server - G4 PKIo Priv G-TLS SYS - 2025`
CIBG test, G1	`TEST UZI-register Private Server CA G1`
CIBG acceptance, G4	`ACCEPTATIE UZI Server - G4 Priv G-TLS SYS - 2025`

The easiest way to get the exact name: run the command with `"<CA name>"` as it is. The tool then fails with `ca-fingerprint certificate not found in chain` and lists the names of all certificates in your chain under `Candidates:`. Copy the one for the CA and run the command again.

Check https://hub.docker.com/r/nutsfoundation/go-didx509-toolkit/tags for the latest version.

The output is the X509Credential as a JWT (a long string starting with `eyJ`). Send it to the vendor, who loads it into the care organization's wallet (see "Load the credential into the wallet").

Non-production environments

For non-production environments, the CIBG issues test certificates whose name, URA and city don't match the real care organization. Where the use case allows it, you can instead create a fake UZI server certificate with a test CA:

1. Check out https://github.com/nuts-foundation/go-didx509-toolkit. You need Bash and `openssl`.
2. In the `test_ca` directory, run:

   ```
   ./issue-cert.sh <domain (CN)> <name (O)> <city (L)> <uzi> <ura> <agb>
   ```

   The domain, UZI and AGB values can be anything; use cases don't use them.
3. The certificate chain and private key are written to the `out` directory. Copy them to your `certs` directory.
4. Create the credential as above, but with `"CN=Fake UZI Root CA"` as the CA:

   ```
   docker run --rm -v "$(pwd)/certs:/certs" \
     nutsfoundation/go-didx509-toolkit:1.2.0 \
     vc /certs/<domain>-chain.pem /certs/<domain>.key \
     "CN=Fake UZI Root CA" \
     <did>
   ```

Which CAs an environment accepts is set in the use case's Discovery Service definition.

360-graden-clientbeeld

The X509Credential is the only credential a care organization needs to obtain. Accepted certificates per environment:

Environment	Accepted certificates

Development, test	Fake UZI certificates and CIBG test certificates (G1 and acceptance G4)
Acceptance	CIBG test certificates (G1 and acceptance G4) and real G1 certificates. Not fake ones, and not real G4 ones.
Production	Real CIBG certificates only (G1 and G4)

Lifecycle

The X509Credential is only valid as long as the UZI certificate it was created from. Create and load a new X509Credential:

- before the UZI certificate expires, with the renewed certificate;
- when the UZI certificate is replaced or revoked;
- when the care organization's details in the certificate change (name, city).

Load the new credential as described in "Load the credential into the wallet", then remove the old one (see "Renew credentials").

Revision #13
Created 2025-08-12 11:56:24 UTC by Jorrit Spee
Updated 2026-10-28 10:11:28 UTC by Rein Krul