

Configuration of Nuts identity

This chapter provides a manual for the initial configuration of the Nuts-node for an organization. It covers:

- Identity creation (once per care organization)
- Registration for a use case (once per care organization, per use case)

- [1. Create organization identity](#)
- [2. Issue X509Credential](#)
- [3. Load X509Credential into organization wallet](#)
- [4. Discovery Service registration](#)

1. Create organization identity

First, a new identity, with associated DID(s), needs to be created for the tuple (organization, environment).

Using API

Perform the following REST API call. Note that `subject` is optional.

```
POST <internal Nuts interface>/internal/vdr/v2/subject
Content-Type: application/json

{"subject": "fooorg-acc"}
```

Using Nuts Admin

1. Log in to the **organization**'s Nuts Admin application. Click on "Identities".
2. Click on the plus sign on the right side.
3. Enter the subject: it consists of the tenant, dash, short form of the environment (e.g. `fooorg-acc`). This will act as key when interacting with the Nuts node's APIs for this instance. *Note: as deployments are isolated, theoretically, a static key could be used. But having the tenant and environment in the name reduces the possibility of leakage due to mixed-up environments.*
4. Click on "Create Identity".

2. Issue X509Credential

This document describes how to issue the X509Credential for a UZI server certificate, used to authenticate in several applications on Nuts/ use cases.

Introduction

Several applications on nuts specify that care organizations are authenticated using a Verifiable Credential containing their URA number. As the CIBG doesn't issue these credentials yet, it's derived from the X.509 UZI Server Certificate in the form of an X509Credential.

The credential will then be used for 2 purposes:

- Registering on the Discovery Service, so other participants can resolve key material and API endpoints of the care organization(s).
- Authenticating to other participants when enrolling patients and exchanging data.

The X509Credential will be issued to a DID (Decentralized Identifier) that resides in the Nuts node into which the credential will be loaded. Afterward, the Nuts node can authenticate as the care organization by presenting the credential.

The X509Credential will contain the UZI certificate itself, and the extracted (from the certificate) URA, name, and locality of the care organization. The X509Credential is signed with the private key backing the certificate. Note that the X509Credential **will not contain the private key** itself.

For non-production environments (test, acceptance), see the last section of this document.

Requirements

- DID (Decentralized Identifier) to which the X509Credential will be issued. Supplied by the Nuts node hoster (usually the vendor).
- CIBG UZI Server Certificate:
 - Full certificate chain (root CA certificate up until the leaf certificate) as PEM file
 - Decrypted private key of the leaf certificate as PEM file
- A system with Docker installed to run `nutsfoundation/go-didx509-toolkit` (<https://hub.docker.com/r/nutsfoundation/go-didx509-toolkit>)
 - The source code of the Docker image can be found here: <https://github.com/nutsfoundation/go-didx509-toolkit>
- For **non-production environments**: a checkout of <https://github.com/nutsfoundation/go-didx509-toolkit> that can run a Bash script, with `openssl` installed.

Procedure

1. Run the following command from a directory that contains a directory called `certs` with the certificate (`certificate.pem`) and private key (`privatekey.pem`) PEM files. Replace `<did>` with the proper DID.

```
docker run --rm -v "$(pwd)/certs:/certs" \  
  nutsfoundation/go-didx509-toolkit:1.1.0 \  
  vc /certs/certificate.pem /certs/privatekey.pem \  
  "CN=UZI-register Private Server CA G1,O=CIBG,C=NL,2.5.4.97=#130e4e54524e4c2d3530303030353335" \  
  <did>
```

Note: check <https://hub.docker.com/r/nutsfoundation/go-didx509-toolkit/tags> for the latest version.

This yields an X509Credential in JWT format which can then be loaded into the care organization's credential wallet in its Nuts Node. E.g.:

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImRpZDp4NTA5OjA6c2hhMjU2OnN6cU1hVHBuRDZHTjBhUnJUOThIVjRiaEFvT2d5SX  
RFWIZ5c2tZeUxfUWM6OnNhbJpvdGhlck5hbWU6Mi4xNi41MjguMS4xMDA3Ljk5LjlxMTAtMS0wLVMtMjE1MTQtMDAu  
MDAwLTA6OnN1YmplY3Q6TDpOaWV1d2VnZWluOk86QW50b25pdXMMjBaaWVrZW5odWlzlzAiLCJ0eXAiOiJKV1QiL  
Cj44NWMiOiSiTUljRDBqQ0NBcnFnQXJkFnSVVGvFBPK3BVazMyUVdzWXIMWWRsTFRtbFjXVmt3RFFZSkvWklodm  
NOQVFFTEJRQXdHekVaTUjJR0ExVUVBd3dRUM1GclpTQlZXa2tnVW05dmRDQkRRVEFIRncweU5UQXhNamd4TmPF  
eU1EUmFGdzB5TmpeBeE1qZ3hOakV5TURSUY1JR0RNVUI3UUFZRFZRUURERGXoYm5SdmjtbDFjM3BwWld0bGJtaDF  
hWE11WkdWMIpXeHjZjRzFsYm5RdWFXNTBaV2R5WVhScGlyNHVlbTI5WjJkKcGFtcHZkUzVqYjIweEhEQWFZCZ05WQkF  
vTUUwRnVkrZl1YVhWekIGChBaV3RsYm1oMWFYTXhFekFSQmdOVkYjY01DazVwWlhWM1pXZGxhVzR4Q2pBSUJnTI  
ZCQVVUQVRBd2dnRWlNQTBlHQ1Nxr1NjYjNEUUVVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUUR0RIRMBERPay9IU3NqN  
WgrdE9Cb2lvUURStloTEFTRUIOTk9ROEpldnhmaW5aMkNySUJMMm9QVUxOY1pyZHBGUXE4VXVGdnI0bVI0SVZM  
OE1HZVpDRFNiOVMvQnhpZ1RJM3RUenhyRVdCUTliR3paZVIKd29BbHRaejNUV01nVUcwaTVoek1vUFRtMnRxQW1P  
VXk3ZmUyZldMNfUyQ0IGWXhHbzRMVHZOcUphdHROZWRYN1lxTE5qVnNUSGE5eDjM3NIWDFyUTV4bll1Z1V4UIh  
4blFUVExQVUV6MmVyeFUxeTdDRk5LZDA1SGhoVVBqYUZSOUM5MXdCNDdRL2ZFekhuaeIiIUTBXT2dEUTjMmHNIW  
VdiS010Q05aZ3VCWIRJRG9Qc0RvSkZrczdpT3ozM0c0NW1vei9QN2NsWHV2YTIJMDZQcTdIRXV6K1dud2kxVjRLREF  
nTUJBQUdqZ2FRd2dhRXdiUVIEVIIwbEJCWXdGQVlJS3dZQkRVUUhBd0VHQ0NzR0FRVUZCd01DTUVBR0ExVWRFUVE  
1TURIZ05RWURWUUVGboM0TUxESXVNVF11TIRJNEExqRXVNVEF3Tnk0NU9TNHINVEV3TFRFdE1DMVRMVEI4TIRFME  
xUQXdMakF3TUMwd01CMEdBmVvkRGdRV0JCUxSU5xc3NldW1aTkROZmg5UDF6a3FteTBPakFmQmdOVkhTTU  
VHREFXZ0JTYnNjOUY4RGV6M1hqSWNpWDNINXU2Y3RRU3ZUQU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFOc3RY  
elpBdm9WUUpSNGdMMEtxcUICQ1RMBURsYktGdmVhemd3SFVsOGkvVStLTnNUQ2tscDJVM05FMXNuQmVXNVIVR3  
VXMzhvVS2QzL0p4Zm5pdk9XZXhyeFlwY0pCY2IjYVhY0ZMSDyYnMlUbxhtZnpEblZFUUhWUUVF4b1djShhVbmlMWV  
pxTDQzK2hYekFiMWRxKzImaHN4djViSURDemw0SHFCRy9oMIA3N0tsUVZOMzJML0ZTZ1c2TUINTFJDK0R6RXR5T25  
kd0pYN2pStmxpNGVPOWpXOHh2b25NelZwZmdHaGdjVArLzlaY0Njb2c1QW95dEJDTE54a2tibmR0ZnRZRktXQjZ  
3S2xzeG5JRWNJTDFET2FPalArQnUwZStFRzZuSVdYK1MxakxVKzZjYjk0VXNrRHJhS0VmUk9GQzIOWVILQkF4cDdwT
```

nVnUWNCVUE9PSIsIk1JSUM5akNDQWQ2Z0F3SUJBZ0IVUkZDcVByTDNRUWRCTk9xa3dtWFdOZ3g5cGRRd0RRWUp
Lb1pJaHJzTkFRRUxCUUF3R3pFWk1CY0dBMVVFQXd3UVJtRnJaU0JWV2trZ1VtOXZkQ0JEUVRBZUZ3MHIORREV4TVRF
eE5ERTFNVGhhRncwek5ERXhNRGt4TkRFMU1UaGFNqN4R1RBWEJnTIZCQU1NRUVaaGEyVWdWVnBKSUZKdmlzU
WdRMEV3Z2dFaU1BMEduDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRFQ1SjhnS2R5TUpOaTnjdUft
SitNSUxyTXV3ckt5VFJZaGpVVUZISG41cmNwYUUhOMGh6QjZ2NXQ3NE50NDB4VhSTmFvbURjY2xCSU9sd3Q4ZjYy
SkEycC9qODNFTmZkTHJYdIV1OU5NVGhrcVp3WjJkelJ3Szdsm1VaQnE4TIRRVU83Nfc0TTJxeDhuclhxMzFIV29neFVV
SUZjMVhPUmg1ZWNIYmVMNW1VYjJFNIVsbURtTmdtMmZHZVNtbWlZOHppZUkrS0tZT2hpL2hZdHllaXhyZzdyeFA0d
jBWUnjFc3RjV0FIdFJnWFdRWDBFbEF4czBWcnN5Ni92djNwRXRYaHg4d2lyd2kyeFkxNGQ5SWg4SGRITkkrKzN3SWJ
aejZXVk0zZkQ1UUZIVjJFWkJK3NvbzBwZktqMnRlc2FEejNGUE11TXpjTHQ2VTZQVDRBTEIkQWdNqkFBR2pNakF3TU
E4R0ExVWRfD1FJTUFZQkFmOENBUUF3SFFZRFZSME9CQIIFRkp1eHowWHdON1BkZU1oeUpmY2ZtN3B5MUJLOU1B
MEduDU3FHU0liM0RRRUJDD1VBQTRJQkFRQWhscGt6Njh4MmRHcE9MWDNGekFiOEVIK1kyT1YrUldGcHNNRTIaVkrV
MDZKRVRQZIBDajAyUEg4MmxnVW5jNGplUjgxcIBTc0I0MnNzcW0yUzR6YjAyTmlwNtK1Y0FqxQ0t2bUjMmRWM5aFBQ
Vzj1Z3BOeFQ4WlJVNEXLcnFwVjRuSjZuQnZEcW1HdUg1dXE5Tmc5bDITbk0zZUttZFp0SktjK1pOQVBLEfZBaXVITFR
kcjZXMIVibUtvWkFSUVEwSkxRm5aT3huVWtyOHBRZnhVekVJVWtIzZjKv2FhSS80d280UG5pN3hYZ2dGb1BEcFZ6d
HUvaVAzM1hCTHFYSnd4eEhYaHE5bmM5SIUva0VYRHQ3ajhFZ295Sm83SmpTS2NqcFjmcEdrRTVncXFCNFhOHdB
c0FQVUszalJyZXV5dGxsQXRRVpSYkN0SGJ4Y2xjOXIBlI0sIng1dCl6IkI5UjBhb1dHWDc2WEVtd3FCVlpfQkM4T2Exdy
J9.eyjleHAiOjE3Njk2MTY3MjQsmlzcyI6ImRpZDp4NTA5OjA6c2hhMjU2OnN6cU1hVHBuRDZHTjBhUnJUOTHIVjRiaEFv
T2d5SXRfWIZ5c2tZeUxfUWM6OnNhbJpvdGhlcK5hbWU6Mi4xNi41MjguMS4xMDA3Ljk5LjlxMTAtMS0wLVMtMjE1MT
QtMDAuMDAwLTA6OnN1YmplY3Q6TDpOaWV1d2VnZWluOk86QW50b25pdXMIMjBaaWVrZW5odWlZliwianRpljoiZ
GikOng1MDk6MDpzaGEyNTY6c3pxTWFUcG5ENkdOMGFScIQ5OGVWNGJoQW9PZ3JldEvaVnlza1I5TF9RYzo6c2FuO
m90aGVyTmFtZToyLjE2LjUyOC4xLjEwMDcuOTkuMjExMC0xLTAtUy0yMTUxNC0wMC4wMDAtMD06c3ViamVjdDpM
Ok5pZXV3ZWdlaw46TzpbBnRvbmI1cyUyMFppZWtlbmh1aXMjZjcwODVhZDEtYTQ4Mi00MjU4LWE5NDgtYWQ5YTd
iMjRmZDZkIiwibmJmljoxNzM5NDQyODQ1LClzdWliOijkaWQ6d2ViOmFudG9uaXVzemlla2VuaHVpcy5kZXZlbG9wb
WVudC5pbnRIZ3JhdGlvi56b3JnYmlqam91LmNvbTpubXRzOmIhbTozYmYxY2NmMS0wMWI5LTRmNDUtYTA2Yy05
NTdkMGIwZWJjOTkiLClj2YyI6eyJAY29udGv4dCl6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJW50aWFscy9
2MSJdLCljcmVkJW50aWFsU3ViamVjdCl6W3siaWQiOijkaWQ6d2ViOmFudG9uaXVzemlla2VuaHVpcy5kZXZlbG9wb
WVudC5pbnRIZ3JhdGlvi56b3JnYmlqam91LmNvbTpubXRzOmIhbTozYmYxY2NmMS0wMWI5LTRmNDUtYTA2Yy05
NTdkMGIwZWJjOTkiLClj2Yy4iOmsib3RoZXJOYW11IjoiMi4xNi41MjguMS4xMDA3Ljk5LjlxMTAtMS0wLVMtMjE1MTQtMD
AuMDAwLTAifSwic3ViamVjdCl6eyJMIjoiTmlldXdlZ2VpbilSk8iOijBbnRvbmI1cyBaaWVrZW5odWlZln19XSwidHlwZSI
6WyJWZjZpZmlhYmxiQ3JlZGVudGhlcGlzIlg1MDIDcmVkJW50aWFsIi19fQ.lijO1udVtlnFfEN6qs3ZjTmCyUyCEPSyJHa
BApZeMdhPJ8mtoqSQk6xsr8c4vmQhYfHCJLkibR0y2DvuzX13ksts3vDeidosIRf_yTxChUUa1b2FQMLBc-
t2i01PGIj2B7WMhbbRE1PmDrVAUn2fww9n0V5XMVjstzzNKR9DdFoZ3dr4f0oG-
aaNe2xzS4IM_NpckGpprXqzROh1iavZ6zjnd1GhDH3tq1RMXGBmNQBwlon2Bk2bfb-3oeGkEzhDvt5Mo8sBIDI-E-
4htqZx17UR6uC5J_bVf4fklULxZzeUQ7-y7qeEz5bcyYOin3hKlecWDjRa4yQvCxAjz2ICQ

Additional procedure for non- production environments

For non-production (e.g., test or acceptance) environments, CIBG issues test-grade certificates that don't match the actual care organization's name, URA and locality. In those cases, a self-signed test CA is used to self-issue a "fake" UZI server certificate. The self-signed test CA can be found in the `test_ca` directory in <https://github.com/nuts-foundation/go-didx509-toolkit>. Use the following command (in the `test_ca` directory) to issue a certificate:

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

Note that the values for domain, UZI and AGB are arbitrary and not used in the Shared Care Planning ecosystem.

It outputs the generated certificate chain PEM file and private key PEM file in the `out` directory. Use these in the procedure documented above.

Also, make sure to replace the DN of the certificate CA in the issuance command (`CN=UZI-register Private Server CA G1...`) with `"CN=Fake UZI Root CA"`.

The command should then look as follows:

```
docker run --rm -v "$(pwd)/certs:/certs" \
  nutsfoundation/go-didx509-toolkit:1.1.0 \
  vc /certs/somedomain-chain.pem /certs/somedomain.key \
  "CN=Fake UZI Root CA" \
  <did>
```

3. Load X509Credential into organization wallet

Using API

To load the credential into the Nuts node via its REST API, perform the following HTTP request:

```
POST <internal Nuts interface>/internal/vcr/v2/holder/<subjectID>/vc
Content-Type: application/json

"<JWT>"
```

Replace `<subjectID>` with the subject, and `<JWT>` with the credential in JWT format (a JSON string enclosed in double quotes)

Using Nuts Admin

1. Log in to the **care organization's** Nuts Admin application.
2. Click on "Identities".
3. Click on the wallet to load the credential into (there should be only one).
4. Click on "Load Credential".
5. Enter/paste the credential in the input field.
6. Click on "Load credential".

4. Discovery Service registration

Before another party can interact with you, they will first need to find your endpoints. The discovery service in the Nuts node helps you. You do this once per organization, per use case.

Using API

```
POST <internal Nuts interface>/internal/discovery/v1/<servicedefinition-id>/<subject-id>
```

```
Content-Type: application/json
```

```
{
  "registrationParameters": {
    "abc": "xyz",
    "someparameter": "somevalue"
  }
}
```

Replace `<servicedefinition-id>` with the servicedefinition-id and replace `<subject-id>` with the subject-id of the organization.

“ For each use case a `discovery service definition` is specified that specifies the servicedefinition-id and the registrationParameters.

Using Nuts Admin

to do