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 applicatopns-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/nuts-foundation/go-didx509-toolkit
- For **non-production environments**: a checkout of https://github.com/nuts-foundation/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.:


```
eyJhbGciOiJQUzI1NiIsImtpZCI6ImRpZDp4NTA5OjA6c2hhMjU2OnN6cU1hVHBuRDZHTjBhUnJUOThlVjRiaEFvT2d5SX
RFWlZ5c2tZeUxfUWM6OnNhbjpvdGhlck5hbWU6Mi4xNi41MjguMS4xMDA3Ljk5LjIxMTAtMS0wLVMtMjE1MTQtMDAu
MDAwLTA6OnN1YmplY3Q6TDpOaWV1d2VnZWluOk86QW50b25pdXMlMjBaaWVrZW5odWlzIzAiLCJ0eXAiOiJKV1QiL
CJ4NWMiOlsiTUlJRDBqQ0NBcnFnQXdJQkFnSVVGVFBPK3BVazMyUVdzWXlMWWRsTFRtbFJXVmt3RFFZSktvWklodm
NOQVFFTEJRQXdHekVaTUJjR0ExVUVBd3dRUm1GclpTQlZXa2tnVW05dmRDQkRRVEFlRncweU5UQXhNamd4TmpF
eU1EUmFGdzB5TmpBeE1qZ3hOakV5TURSYU1JR0RNVUl3UUFZRFZRUURERGxoYm5SdmJtbDFjM3BwWld0bGJtaDF
hWE11WkdWMlpXeHZjRzFsYm5RdWFXNTBaV2R5WVhScGIyNHVlbTl5WjJKcGFtcHZkUzVqYjIweEhEQWFCZ05WQkF
vTUUwRnVkRzl1YVhWeklGcHBaV3RsYm1oMWFYTXhFekFSQmdOVkJBY01DazVwWlhWM1pXZGxhVzR4Q2pBSUJnTl
ZCQVVUQVRBd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUUR0RlRMbERPay9lU3NqN
WgrdE9Cb2lvUURrSTloTEFTRUlOTk9rOEpldnhmaW5aMkNySUJMMm9QVUxOY1pyZHBGUXE4VXVGdnl0bVI0SVZM
OE1HZVpDRFNiOVMvQnhpZ1RJM3RUenhyRVdCUTliR3paZVlKd29BbHRaejNUV01nVUcwaTVoek1vUFRtMnRxQW1P
VXk3ZmUyZldMNFUyQ0lGWXhHbzRMVHZOcUphdHROZWRYN1lxTE5qVnNUSGE5eDJjM3NlWDFyUTV4bll1Z1V4Ulh
4blFUVExQVUV6MmVyeFUxeTdDRk5LZDA1SGhoVVBqYUZSOUM5MXdCNDdRL2ZFekhuaEllUTBXT2dEUTJmMHNlW
VdiS010Q05aZ3VCWlRJRG9Qc0RvSkZrczdpT3ozM0c0NW1vei9QN2NsWHV2YTlJMDZQcTdlRXV6K1dud2kxVjRLREF
nTUJBQUdqZ2FRd2dhRXdlUVlEVllwbEJCWXdGQVlJS3dZQkJRVUhBd0VHQ0NzR0FRVUZCd01DTUVBR0ExVWRFUVE
1TURlZ05RWURWUVVGb0M0TUxESXVNVFl1TlRJNExqRXVNVEF3Tnk0NU9TNHlNVEV3TFRFdE1DMVRMVEl4TlRFME
xUQXdMakF3TUMwd01CMEdBMVVkRGdRV0JCUnQxSU5xc3NldW1aTkROZmg5UDF6a3FteTBPakFmQmdOVkhTTU
VHREFXZ0JTYnNjOUY4RGV6M1hqSWNpWDNINXU2Y3RRU3ZUQU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFoc3RY
elpBdm9WUUpsNGdMMEtxcUlCQ1RMbURsYktGdmVhemd3SFVsOGkvVStLTnNUQ2tscDJVM05FMXNuQmVXNVlVR3
VXMzhVS2QzL0p4Zm5pdk9XZXhyeFlwY0pCY2lJYVhYN0ZMSDYyNmlUbXhtZnpEblZFUUhWUVF4b1djSHhVbmlMWV
pxTDQzK2hYekFiMWRxKzlmaHN4djViSURDemw0SHFCRy9oMlA3N0tsUVZOMzJML0ZTZ1c2TUlNTFJDK0R6RXR5T25
kd0pYN2pSTmxpNGVPOWpXOHh2b25NelZwZmdHaGdjdVArLzlaY0NJb2c1QW95dEJDTE54a2tibmR0ZnRZRktXQjZ
3S2xzeG5JRWNJTDFET2FPalArQnUwZStFRzZuSVdYK1MxakxVKzZjYjk0VXNrRHJhS0VmUk9GQzlOWVlLQkF4cDdwT
```

nVnUWNCVUE9PSIsIk1JSUM5akNDQWQ2Z0F3SUJBZ0lVUkZDcVByTDNRUWRCTk9xa3dtWFdOZ3g5cGRRd0RRWUp
Lb1pJaHZjTkFRRUxCUUF3R3pFWk1CY0dBMVVFQXd3UVJtRnJaU0JWV2trZ1VtOXZkQ0JEUVRBZUZ3MHlOREV4TVRF
eE5ERTFNVGhhRncwek5ERXhNRGt4TkRFMU1UaGFNQnN4R1RBWEJnTlZCQU1NRUVaaGEyVWdWVnBKSUZKdmIzU
WdRMEV3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRFQ1SjhnS2R5TUpOaTNjdUFt
SitNSUxyTXV3ckt5VFJZaGpVVUZISG41cmNWYUhOMGh6QjZ2NXQ3NE50NDB4VVhSTmFvbURjY2xCSU9sd3Q4ZjYy
SkEycC9qODNFTmZkTHJYdlV1OU5NVGhrcVp3WjlkelJ3SzdsM1VaQnE4TlRRVU83NFc0TTJxeDhuclhxMzFlV29neFVV
SUZjMVhPUmg1ZWNlYmVMNW1VYjJFNlVsbURtTmdtMmZHZVNtbWlzOHppZUkrS0tZT2hpL2hZdHllaXhyZzdyeFA0d
jBWUnJFc3RjV0FldFJnWFdRWDBFbEF4czBWcnN5Ni92djNwRXRYaHg4d2Iyd2kyeFkxNGQ5SWg4SGRlTkkrKzN3SWJ
aejZXVk0zZkQ1UUZlVjJFWkJlK3NvbzBwZktqMnRIc2FEejNGUE11TXpJTHQ2VTZQVDRBTElkQWdNQkFBR2pNakF3TU
E4R0ExVWRFd1FJTUFZQkFmOENBUUF3SFFZRFZSME9CQllFRkp1eHowWHdON1BkZU1oeUpmY2ZtN3B5MUJLOU1B
MEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQWhscGt6Njh4MmRHcE9MWDNGekFiOEVlK1kyT1YrUldGcHNNRTlaVkRV
MDZKRVRQZlBDajAyUEg4MmxnVW5jNGplUjgxclBTc0l0MnNzcW0yUzR6YjAyTmlwNTk1Y0FxQ0t2bUJmRWM5aFBQ
VzJ1Z3BOeFQ4WlJVNExLcnFwVjRuSjZuQnZEcW1HdUg1dXE5Tmc5bDlTbk0zZUttZFp0SktjK1pOQVBLeFZBaXVlTFR
kcjZXMlVibUtvWkFSUVEwSkxrRm5aT3huVWtyOHBRZnhVekVJVWtlZzJkV2FhSS80d280UG5pN3hYZ2dGb1BEcFZ6d
HUvaVAzM1hCTHFYSnd4eEhYaHE5bmM5SlUva0VYRHQ3ajhFZ295Sm83SmpTS2NqcFJmcEdrRTVncXFCNFNhOHdB
c0FQVUszalJyZXV5dGxsQXRRVVpSYkN0SGJ4Y2xjOXlBll0sIng1dCI6Ikl5UjBhb1dHWDc2WEVTd3FCVlpfQkM4T2Exdy
J9.eyJleHAiOjE3Njk2MTY3MjQsImlzcyI6ImRpZDp4NTA5OjA6c2hhMjU2OnN6cU1hVHBuRDZHTjBhUnJUOThlVjRiaEFv
T2d5SXRFWlZ5c2tZeUxfUWM6OnNhbjpvdGhlck5hbWU6Mi4xNi41MjguMS4xMDA3Ljk5LjIxMTAtMS0wLVMtMjE1MT
QtMDAuMDAwLTA6OnN1YmplY3Q6TDpOaWV1d2VnZWluOk86QW50b25pdXMlMjBaaWVrZW5odWlzIiwianRpIjoiZ
GlkOng1MDk6MDpzaGEyNTY6c3pxTWFUcG5ENkdOMGFSclQ5OGVWNGJoQW9PZ3lJdEVaVnlza1l5TF9RYzo6c2FuO
m90aGVyTmFtZToyLjE2LjUyOC4xLjEwMDcuOTkuMjExMC0xLTAtUy0yMTUxNC0wMC4wMDAtMDo6c3ViamVjdDpM
Ok5pZXV3ZWdlaW46TzpBbnRvbml1cyUyMFppZWtlbmh1aXMjZjcwODVhZDEtYTQ4Mi00MjU4LWE5NDgtYWQ5YTd
iMjRmZDZkIiwibmJmIjoxNzM5NDQyODQ1LCJzdWIiOiJkaWQ6d2ViOmFudG9uaXVzemlla2VuaHVpcy5kZXZlbG9wb
WVudC5pbnRlZ3JhdGlvbi56b3JnYmlqam91LmNvbTpudXRzOmlhbTozYmYxY2NmMS0wMWI5LTRmNDUtYTA2Yy05
NTdkMGIwZWJjOTkiLCJ2YyI6eyJAY29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy9
2MSJdLCJjcmVkZW50aWFsU3ViamVjdCI6W3siaWQiOiJkaWQ6d2ViOmFudG9uaXVzemlla2VuaHVpcy5kZXZlbG9wb
WVudC5pbnRlZ3JhdGlvbi56b3JnYmlqam91LmNvbTpudXRzOmlhbTozYmYxY2NmMS0wMWI5LTRmNDUtYTA2Yy05
NTdkMGIwZWJjOTkiLCJzYW4iOnsib3RoZXJOYW1lIjoiMi4xNi41MjguMS4xMDA3Ljk5LjIxMTAtMS0wLVMtMjE1MTQtMD
AuMDAwLTAifSwic3ViamVjdCI6eyJMIjoiTmlldXdlZ2VpbiIsIk8iOiJBbnRvbml1cyBaaWVrZW5odWlzIn19XSwidHlwZSI
6WyJWZXJpZmlhYmxlQ3JlZGVudGlhbCIsIlg1MDlDcmVkZW50aWFsIl19fQ.IjO1udVtlnFfEN6qs3ZjTmCyUyCEPSyJHa
BApZeMdhPJ8mtoqSQk6xsr8c4vmQhYfHCJLkibR0y2DvuzX13kstsn3vDeidosIRf_yTxChUUa1b2FQMLBc-
t2i01PGIj2B7WMhbbRE1PmDrVAUn2fwv9n0V5XMVJstzzNKR9DdFoZ3dr4f0oG-
aaNe2xzS4IM_NpchKGpprXqzROh1iavZ6zjnd1GhDH3tq1RMXGBmNQBwlon2Bk2bfB-3oeGkEzhDvt5Mo8sBlDI-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 looks 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>
```

Revision #4
Created 12 August 2025 11:56:24 by Jorrit Spee
Updated 30 September 2025 13:39:48 by Rein Krul