

# UZI server certificates with RFC023

## PKI overhead & UZI server certificates

The Dutch government has a Public Key Infrastructure (PKI) that is used to establish trust between parties. The PKI framework is currently in place and makes use of PKI Overhead Certificates issued by the root CAs of the Dutch government. In healthcare a specific instance of PKI overhead certificates are issued: the UZI certificates. These certificates are used to establish trust between parties in the healthcare sector. The UZI certificates are issued by the UZI register, which is a trusted party that is capable of verifying the identity of the holder of the certificate. The UZI register signs the certificate with its own private key. The holder of the certificate can then use the public key of the UZI register to verify the signature of the certificate. This way the holder of the certificate can prove that the certificate is valid and that the information in the certificate is correct. The UZI certificates are issued to:

- Individuals that work in healthcare, such as doctors, nurses, etc. They hold this certificate on a UZI card.
- Organisations that work in healthcare, such as hospitals, pharmacies, etc. They hold this certificate as server certificates.

## UZI certificate structure for organisations

The UZI certificate is used to identify the holder of the certificate. The UZI certificate contains information about the holder of the certificate. This information is used to identify the holder of the certificate. The UZI certificate contains the following information (of interest):

- The `subject.CN` The full FQN.
- The `subject.O` the name of the holder of the certificate.
- The `subject.serialNumber` The URI number
- The `subject.C` The subject country
- The `subject.ST` The subject state
- The `subject.L` The subject locality (city)
- The `subject.CN` the full FQN.
- The `san.otherName` a string containing `<OID CA>-<versie-nr>-<UZI-nr>-<pastype>-<Abonnee-nr>-<rol>-<AGB-code>`, where:
  - `<OID CA>` is the OID of the CA that issued the certificate, `2.16.528.1.1007.99.2110` for CIBG.
  - `<versie-nr>` is the version number of the certificate.
  - `<UZI-nr>` is the UZI number of the holder of the certificate, same as `subject.serialNumber`.
  - `<pastype>` is the type of the holder of the certificate, always `S`.
  - `<Abonnee-nr>` is the subscriber URA of the holder of the certificate.

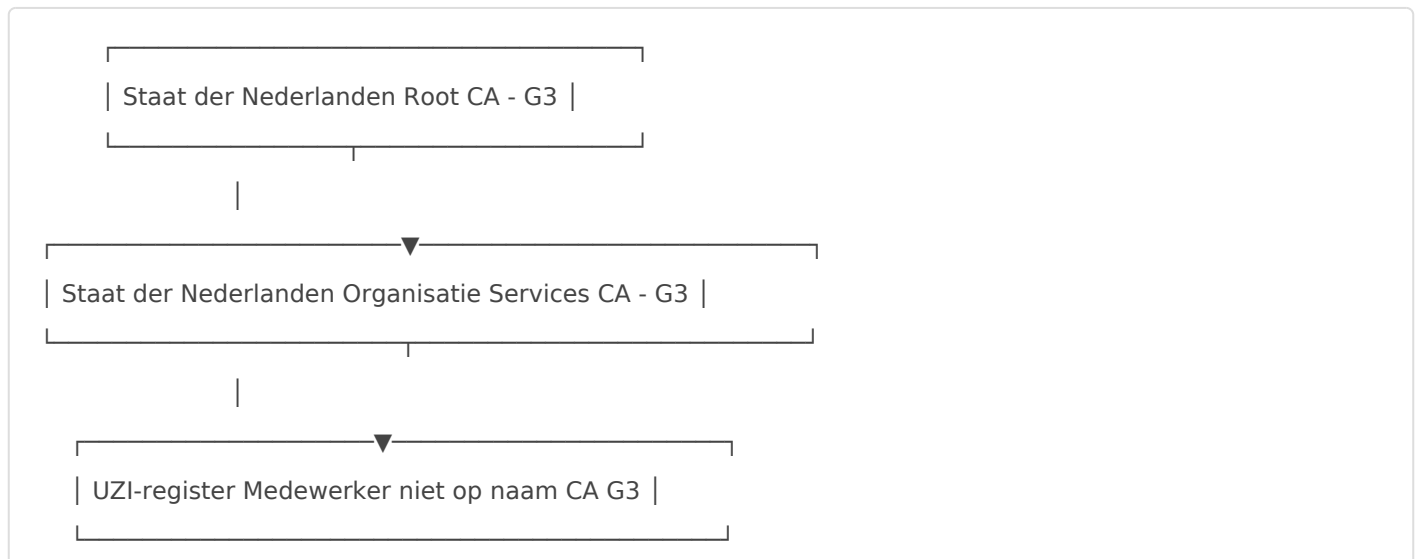
- `<rol>` is the role of the holder of the certificate, always "0.00"
- `<AGB-code>` is the AGB code of the holder of the certificate.

# Mapping UZI certificate to X509Credential

The mapping of certificates to did:x509 depends on the UZI X.509 Certificate structure.

## The ROOT G3

The `did:x509` specification dictates that the fingerprint of the Root CA is part of the did:x509. For mapping an UZI certificate to an X509Credential the ROOT CA MUST match one of the certificates in the UZI ROOT CA register hierarchy. For G3 this is:



## Field mapping of the UZI credential

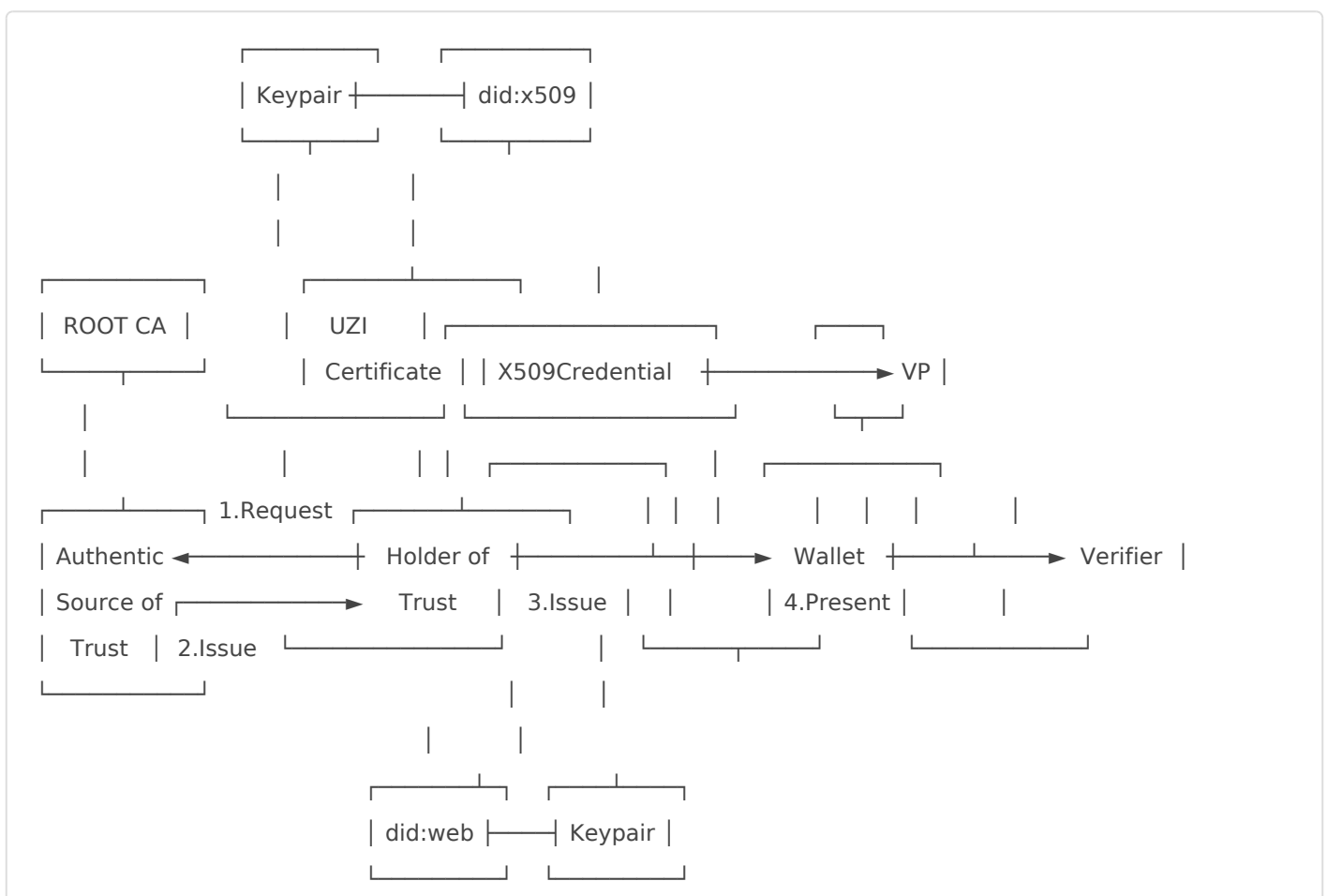
The following fields are commonly used for mapping UZI certificates to X509Credentials

- The `subject.O` the name of the holder of the certificate. Maps to `subject.O` in the X509Credential.
- The `subject.L` The subject locality (city)
- The `san.otherName` a string containing `<OID CA>-<versie-nr>-<UZI-nr>-<pastype>-<Abonnee-nr>-<rol>-<AGB-code>`, where:
  - `<OID CA>` is the OID of the CA that issued the certificate, `2.16.528.1.1007.99.2110` for CIBG.
  - `<versie-nr>` is the version number of the certificate.
  - `<UZI-nr>` is the UZI number of the holder of the certificate, same as `subject.serialNumber`.

- `<pastype>` is the type of the holder of the certificate, always `S`.
- `<Abonnee-nr>` is the subscriber URA of the holder of the certificate.
- `<rol>` is the role of the holder of the certificate, always "0.00"
- `<AGB-code>` is the AGB code of the holder of the certificate.

# The use of UZI server certificate in the Nuts network or identifying organizations

The focus on trust in the NUTS network for organizations lies primarily on the URA number identified as the `<Abonnee-nr>` on the UZI certificate. This number is used to identify the subject of the certificate within the Dutch healthcare ecosystem. The subject of the certificate can use the UZI certificate in combination with the private key to prove the ownership of the URA number. The diagram below shows how the UZI certificate can be used to transfer the trust from the UZI register acting as "authentieke bron" into the NUTS ecosystem using the `did:x509` method and the `X509Credential` Verifiable Credential.



This diagram represents the process of establishing trust, based on the use of X.509 certificates, the `X509Credential` and `did:x509` within a trust network. Below is a step-by-step explanation of the diagram:

## Key Components

### 1. Root CA:

- The starting point for trust. The Root Certificate Authority (CA) is a trusted source that issues and signs certificates to intermediate or end-user entities.

### 2. UZI Certificate:

- A specific X.509 certificate issued by the Root CA (or its intermediaries) to establish trust for the holder (e.g., an organization or an individual).

### 3. Keypair:

- Generated by the certificate holder, this is the private-public key pair required for signing and authentication processes.

### 4. `did:x509`:

- A Decentralized Identifier (DID) based on an X.509 certificate. It links decentralized systems with the trust of traditional X.509 certificates.

### 5. `X509Credential`:

- A Verifiable Credential (VC), such as a "X509Credential," which is issued by the certificate holder using its `did:x509` identifier and signed with the corresponding keypair. This credential is stored in the holder's wallet.

### 6. Wallet:

- A secure digital storage system for holding the `X509Credential`. It manages credentials and is used for presenting them to verifiers.

### 7. Verifier:

- An entity that validates the presented credential and establishes the holder's identity based on its associated trust components (e.g., `did:x509`, certificate chain, etc.).

### 8. `did:web`:

- Another Decentralized Identifier (DID) the holder may use to represent their identity and interact within the decentralized trust ecosystem.

# Process Steps

## Step 1: Keypair Generation and Request

The **holder** generates a keypair (private and public key) to represent their identity. They submit the public key as part of a **Certificate Signing Request (CSR)** to the Root CA (or intermediate CA). Within the CSR terminology, the **holder** is the **subject** of the CSR.

## Step 2: Certificate Issuance

The Root CA (or its intermediate CA) verifies the request and issues an **X.509 certificate** (e.g., a UZI certificate) to the subject. This certificate includes information about the subject (e.g., subject name and organization) and is signed by the CA. This guarantees the authenticity of the certificate. Note that the **holder** and **subject** are the same concepts but are named differently between the different terminologies.

## Step 3: X509Credential Issuance

The **holder** uses their X.509 certificate to create a **X509Credential or Verifiable Credential**. The process includes:

1. Using the certificate's `did:x509` identifier as the credential's **issuer**.
2. Signing this credential with the holder's private key (from the keypair).
3. Storing the credential securely in the **Wallet** for future use.

## Step 4: Credential Presentation

When the holder needs to prove their identity to a verifier (e.g., during authentication), they present the **X509Credential** from their wallet to the **Verifier**. This process includes:

1. The presentation of the digital credential as a Verifiable Presentation (VP).
2. Signing the presentation with the holder's private key to ensure it hasn't been tampered with.

## Step 5: Verification

The **Verifier** validates the credential and presentation. This includes:

1. Checking the integrity of the credential and presentation signature.
2. Confirming the certificate chain back to the **Root CA** to ensure the issuer of the X.509 certificate is authentic and trusted.
3. Validating the use case-specific attributes in the credential (e.g., fields like organization, UZI number, or other subject information).
4. Ensuring the credential has not been revoked using methods like CRL.

Trust is Established:

If all checks pass, the Verifier trusts the credential presented by the holder. The credential's trustworthiness is derived from:

1. The Root CA that anchors trust.
2. The validity of the X.509 certificate and the associated DID (`did:x509`).
3. Alignment of attributes between the X509Credential and the certificate.

---

Revision #3

Created 4 March 2025 10:13:06 by Roland Groen

Updated 4 March 2025 13:18:18 by Roland Groen