MICROSOFT ENTRA VERIFIED ID

Verifiable Credentials, Microsoft style
TF-DLT, Sept 19, 2022

Peter Clijsters, Martin van Es, Niels van Dijk

Use cases

- Credential exchange between institutions
- Company ID as Guest login
- Credential issuance towards services

Why investigate?

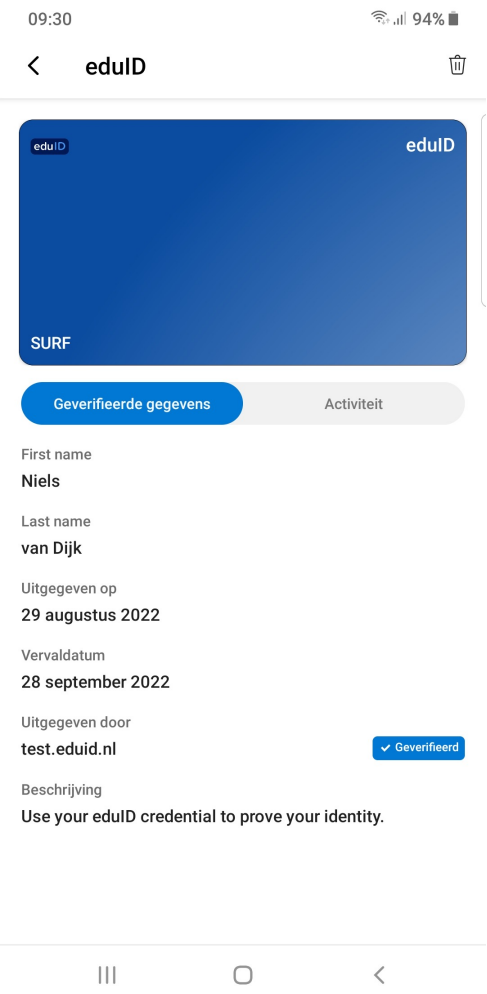
VC issuence & verification capabilities are now part of Azure/Entra

Many institutions have Azure in place already

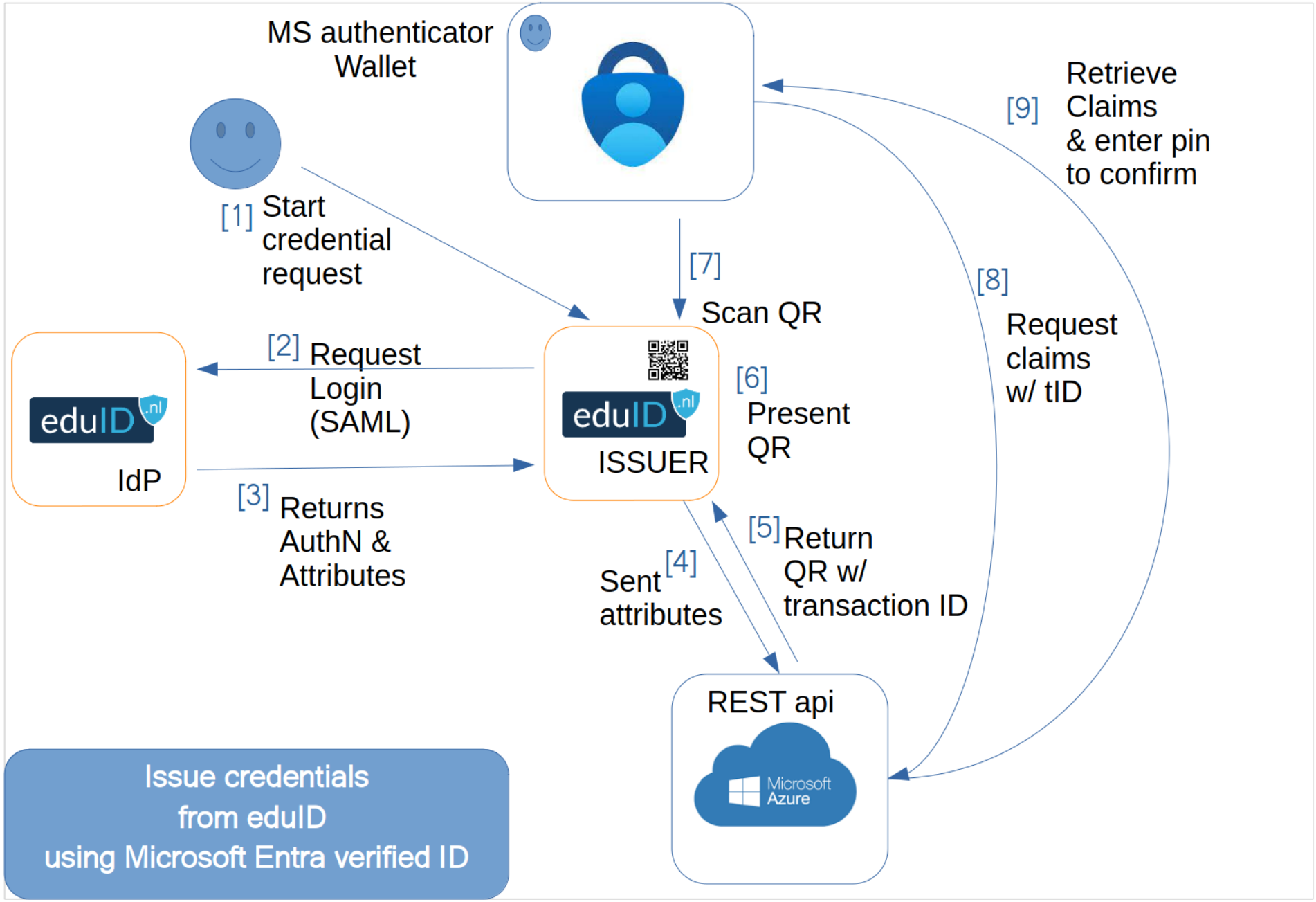
Can we leverage existing (Federated) AAI?

Can we integrate incoming credentials?

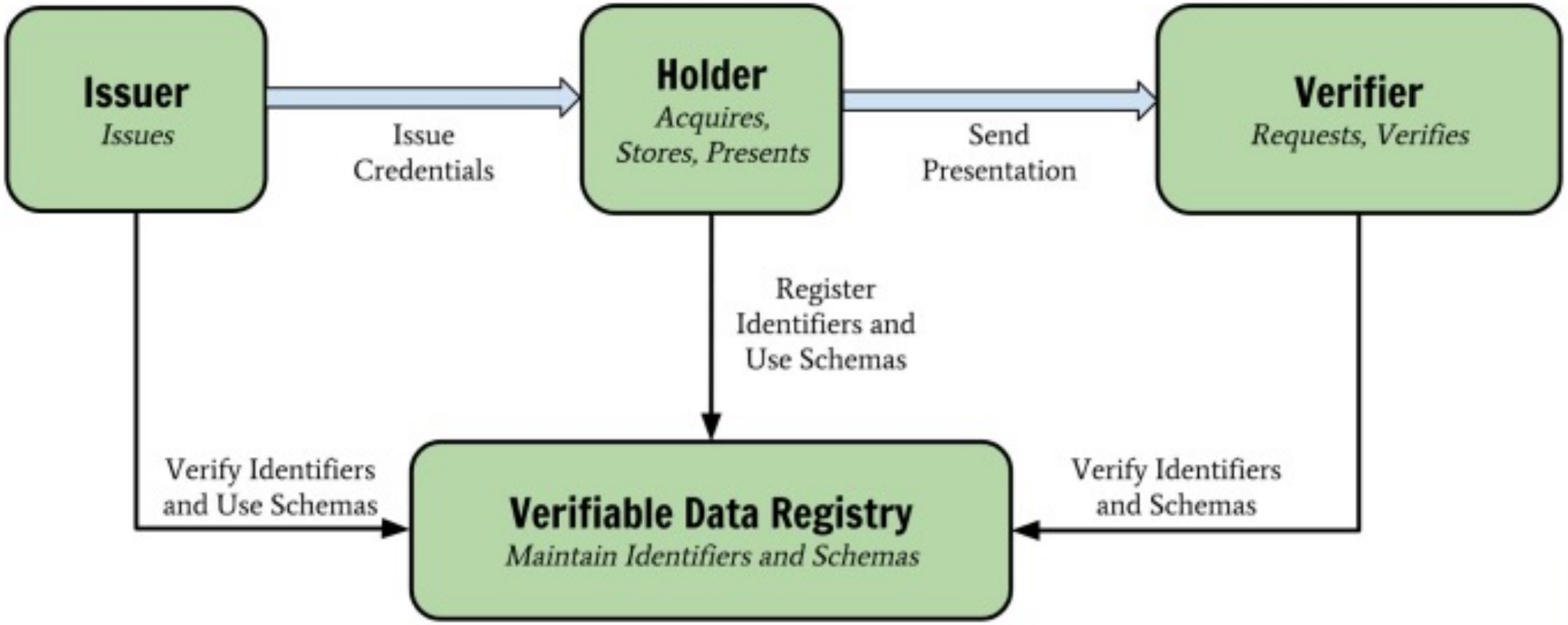
Demo

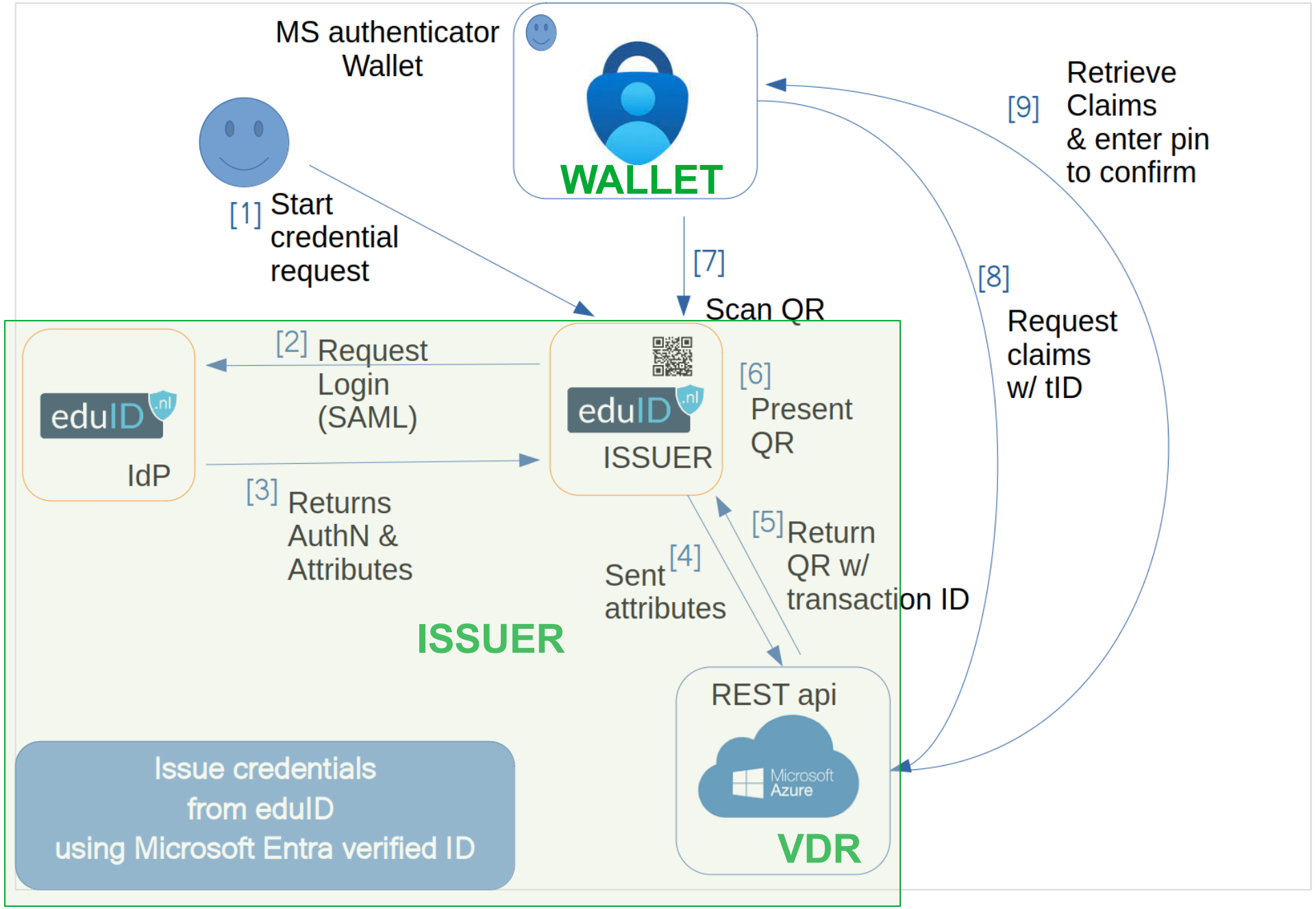

09:30
94%
eduID
eduID
eduID
SURF
Geverifieerde gegevens
Activiteit
First name
Niels
Last name
van Dijk
Uitgegeven op
29 augustus 2022
Vervaldatum
28 september 2022
Uitgegeven door
test.eduid.nl
Geverifieerd
Beschrijving
Use your eduID credential to prove your identity.

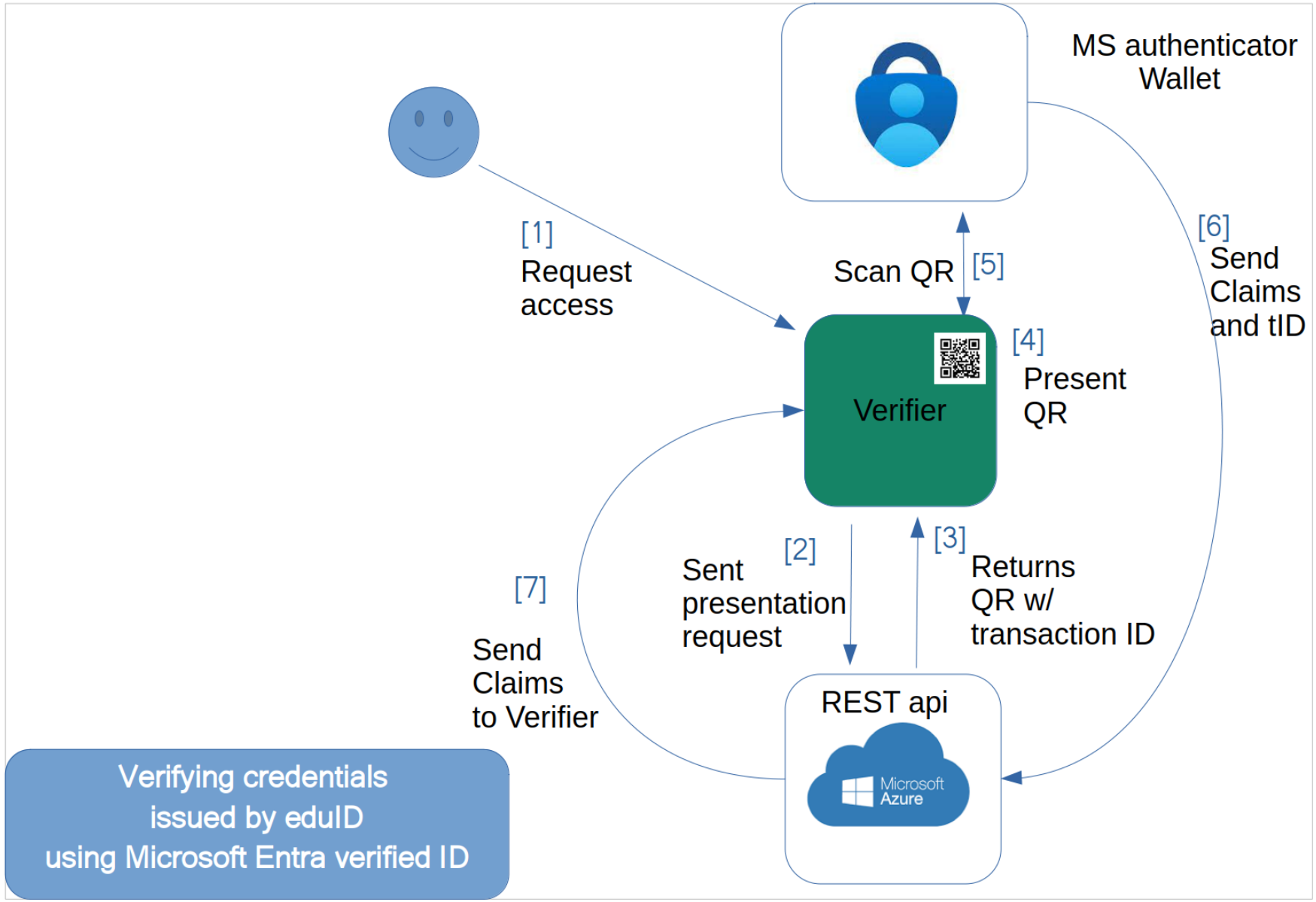

09:30
94%
eduID
eduID
eduID
SURF
Geverifieerde gegevens
Activiteit
VC
29 aug. 2022 15:49:17
U hebt deze geverifieerde id gedeeld met Verifiable Credential eduID Sample
VC
29 aug. 2022 15:26:19
U hebt deze geverifieerde id gedeeld met Verifiable Credential eduID Sample
VC
29 aug. 2022 15:09:22
U hebt deze geverifieerde id gedeeld met Verifiable Credential eduID Sample
S
29 aug. 2022 15:05:53
U hebt deze geverifieerde id ontvangen van van SURF


MS authenticator
Wallet
[1] Start
credential
request
[2] Request
Login
(SAML)
eduID .nl
IdP
[3] Returns
AuthN &
Attributes
[7]
Scan QR
eduID .nl
ISSUER
[6]
Present
QR
[5] Return
QR w/
transaction ID
Sent [4]
attributes
[8]
Request
claims
w/ tID
[9] Retrieve
Claims
& enter pin
to confirm
REST api
Microsoft
Azure
Issue credentials
from eduID
using Microsoft Entra verified ID

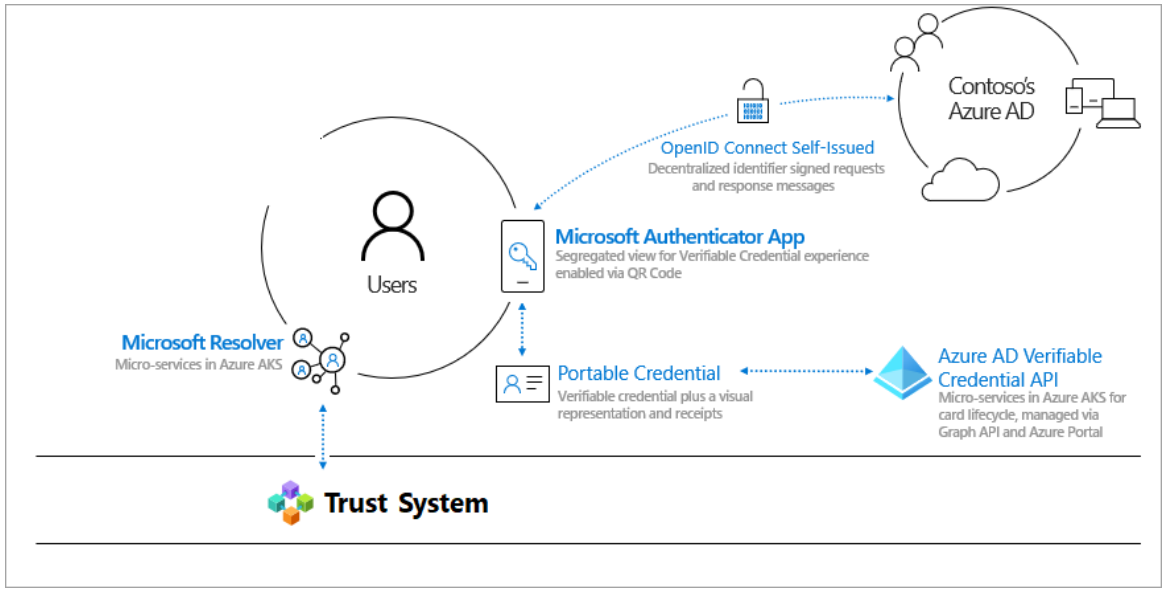
W3C verifiable Credentials




MS authenticator
Wallet
WALLET
[1] Start
credential
request
[7]
Scan QR
[8]
Request
claims
w/ tID
[9] Retrieve
Claims
& enter pin
to confirm
eduID .nl
IdP
[2] Request
Login
(SAML)
[3] Returns
AuthN &
Attributes
eduID .nl
ISSUER
[6]
Present
QR
[5] Return
QR w/
transaction ID
Sent [4]
attributes
ISSUER
REST api
Microsoft
Azure
VDR
Issue credentials
from eduID
using Microsoft Entra verified ID

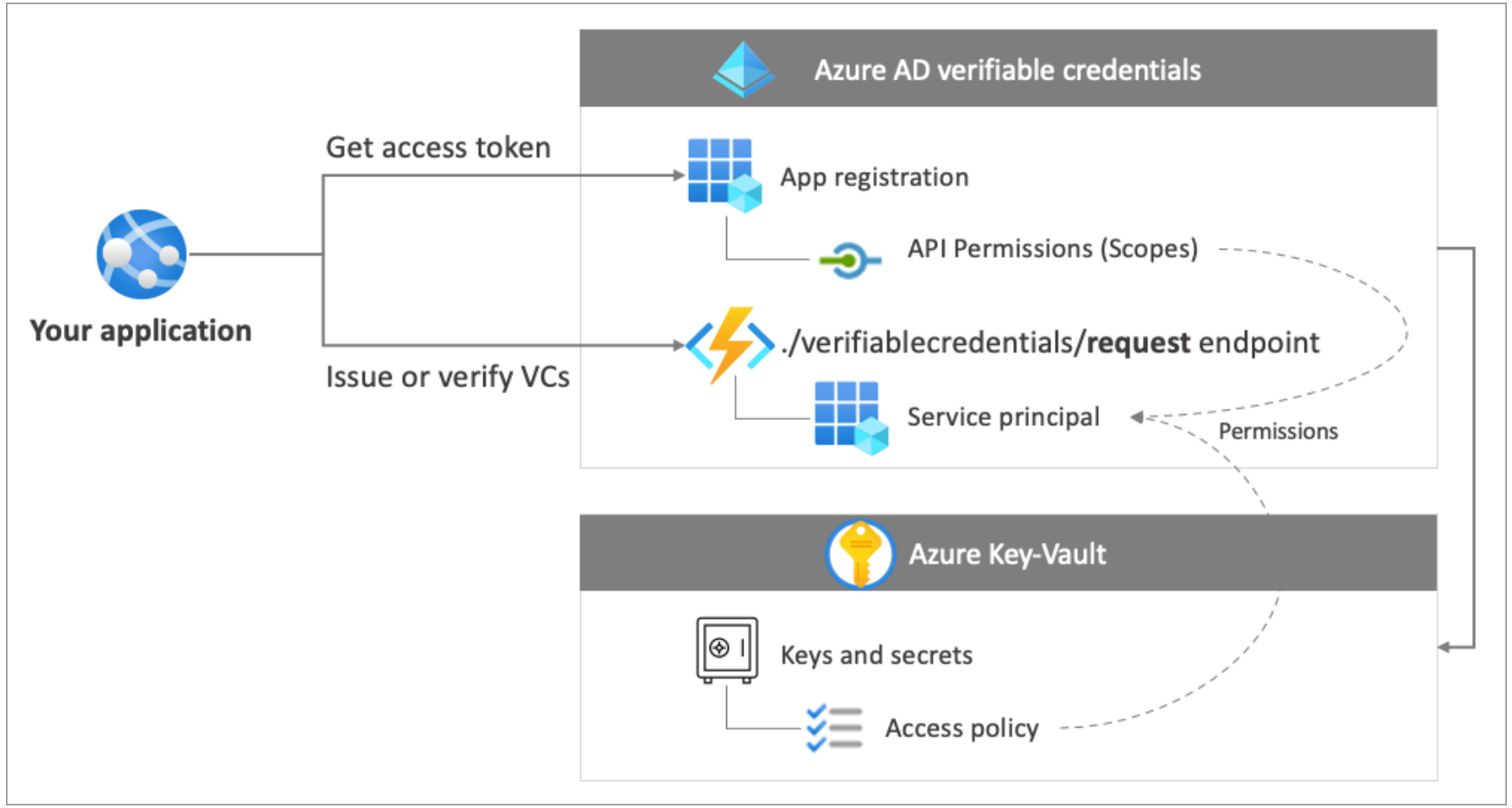

MS authenticator
Wallet
[1]
Request
access
Scan QR
[5]
[6]
Send
Claims
and tID
[4]
Present
QR
Verifier
[2]
Sent
presentation
request
[3]
Returns
QR w/
transaction ID
[7]
Send
Claims
to Verifier
REST api
Microsoft
Azure
Verifying credentials
issued by eduID
using Microsoft Entra verified ID

Requirements



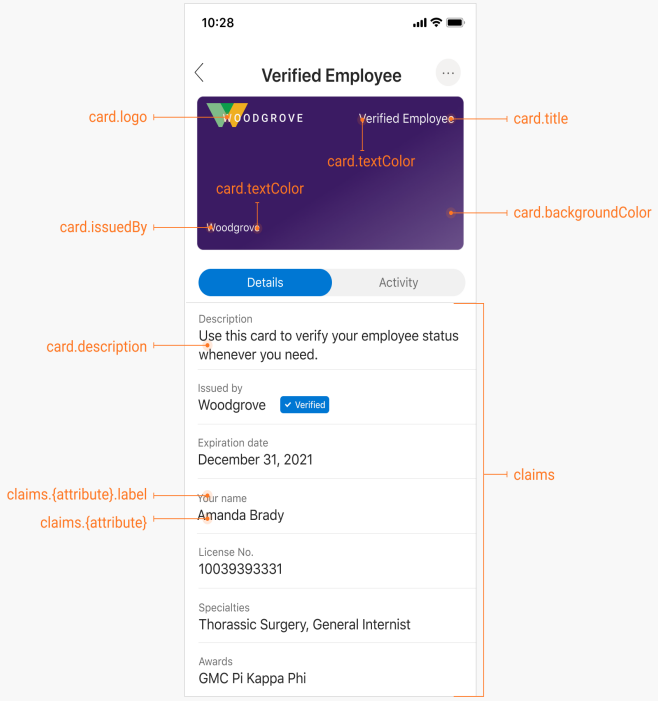
- Azure tennant both for Issuers as well as Verifier
- Activate various components in your tennant
- Issuer and/or verifier client lib (python/.net/java)
- Microsoft Authenticator

Configuration



Source: https://docs.microsoft.com/en-us/azure/active-directory/verifiable-credentials/verifiable-credentials-configure-tenant

Configuration -2



JSON Copy

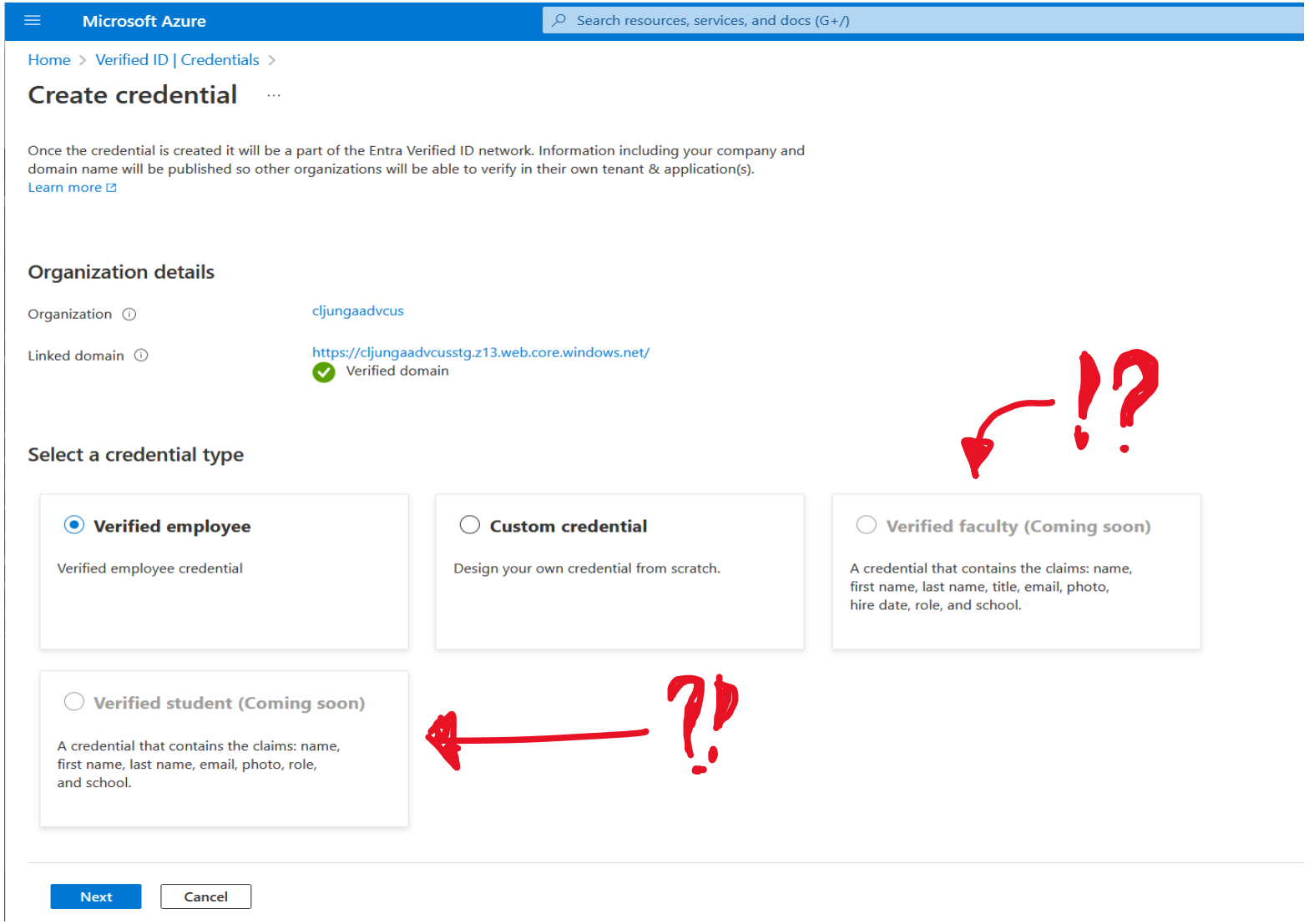
```
{
    "locale": "en-US",
    "card": {
      "title": "Verified Credential Expert",
      "issuedBy": "Microsoft",
      "backgroundColor": "#000000",
      "textColor": "#ffffff",
      "logo": {
        "uri": "https://didcustomerplayground.blob.core.windows.net/public/VerifiedCredentia
        "description": "Verified Credential Expert Logo"
      },
      "description": "Use your verified credential to prove to anyone that you know all abou
    },
    "consent": {
      "title": "Do you want to get your Verified Credential?",
      "instructions": "Sign in with your account to get your card."
    },
    "claims": [
      {
        "claim": "vc.credentialSubject.firstName",
        "label": "First name",
        "type": "String"
      },
      {
        "claim": "vc.credentialSubject.lastName",
        "label": "Last name",
        "type": "String"
      }
    ]
}
```

Source: https://docs.microsoft.com/en-us/azure/active-directory/verifiable-credentials/credential-design

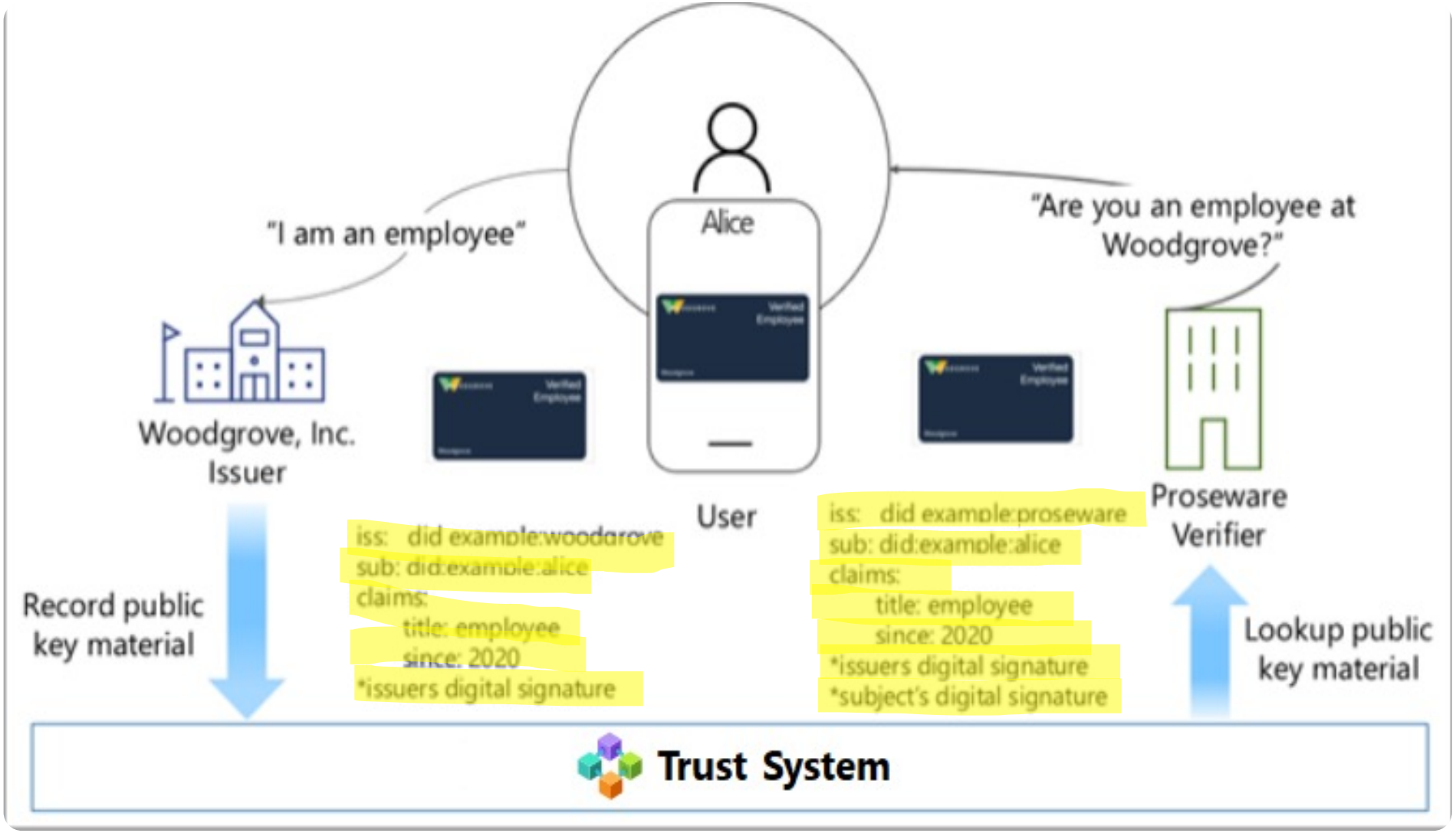
Attestation Types

- **ID token hint**
  → passes claim values after Issuer AuthN (demo'ed)
- **ID tokens**
  →issuance flow requires interactive sign-in to OIDC OP
  todo: will this also work w/ SSP or Shib?
- **Existing verifiable credential**
  → 'transfrom' using another VC
- **Self-asserted claims**
  → let the user type
- **Verifiable Credential for directory based claims**
  → not test (yet)

Directory Based Claims



Standards: OIDC / SIOP, DID:WEB , DID:ION



Trust

In the DID:Web implementation trust is established based on DNS, domain ownership and publishing .well-known file

Azure as well as MS authenticator validate the .well-know file

For DID:ION, Azure (and hence Microsoft) interacts with the VDR.
It is at this point unknown if one can independently valdidate transactions against the ION ledger

Relation between Issuer/Verifier client and Azure backend API is based on client key/secret

Additional trust beyond technical, e.g. federation membership, is not possible except via additional credential issuace

Findings

Setup of Issuer or Verifier ***is super easy*** (< 1 hour)

VC implementation does not make direct use of backend AAI storage like Azure AD; all credentials must be presented to the API. ***Azure is only used as a 'token translation' proxy***.

In DID:ION implementation Azure also interacts with the ledger (the wallet ***does not***)

No selective release of credentials by user, only the entire credential can be presented

Each issuer ***and verifier*** must have its own Azure tennant, no none-Azure implementations are known at this time

Conclusions

Implementation is ***very tightly locked*** into the MS ecosystem, not only technically, but also from the trust perspective

Integration is lightweight, AZURE takes care of heavy lifting

Privacy and scalability of VC release looks challenging

No selective controle over Credential usage by users

Given that Azure ***already has federation capabilities***, the only clear benefit of verifiedID seems to be that ***it does not need upfront trust establisment***.

No integrated provisioning path for Verifier

Verifier Azure tennant requirement is potential barrier