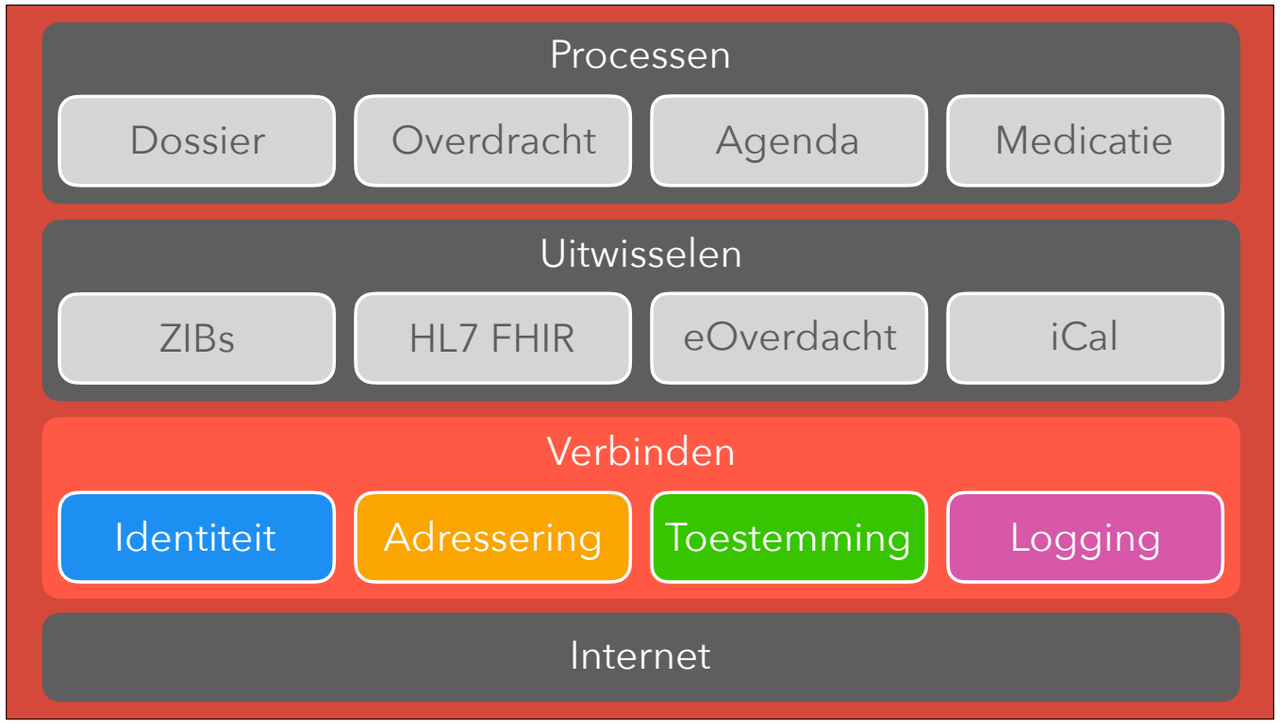
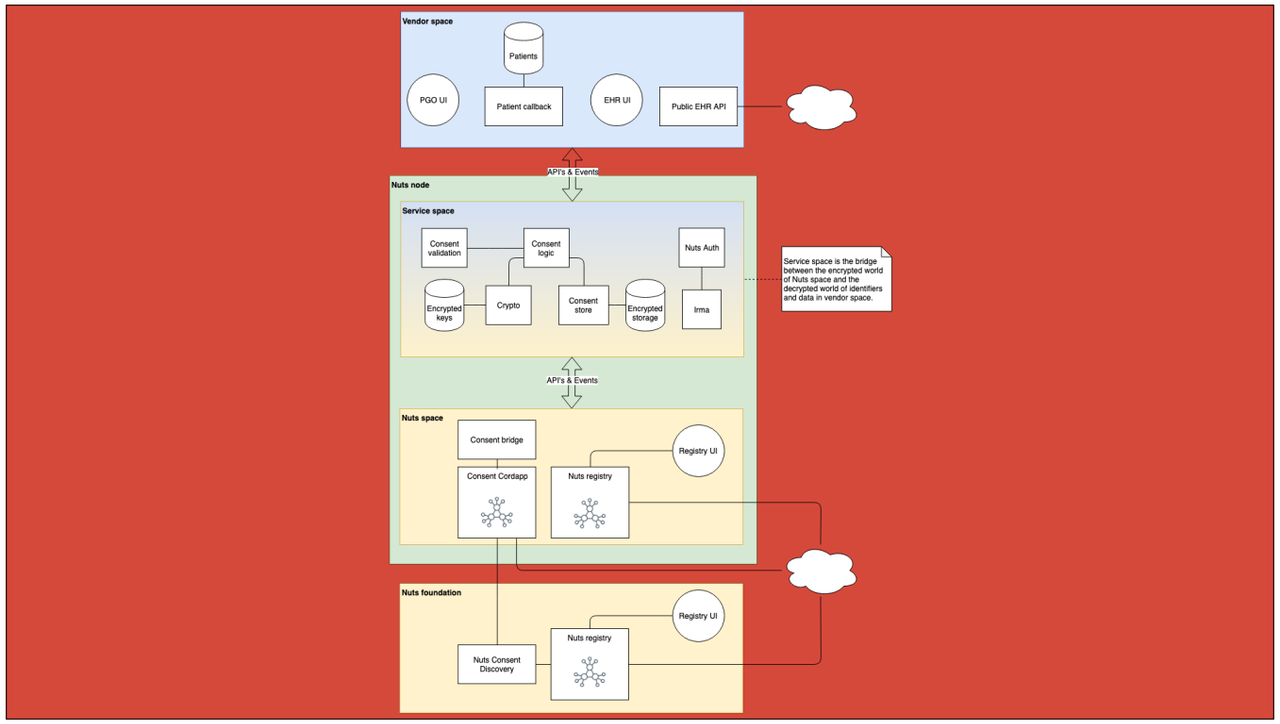
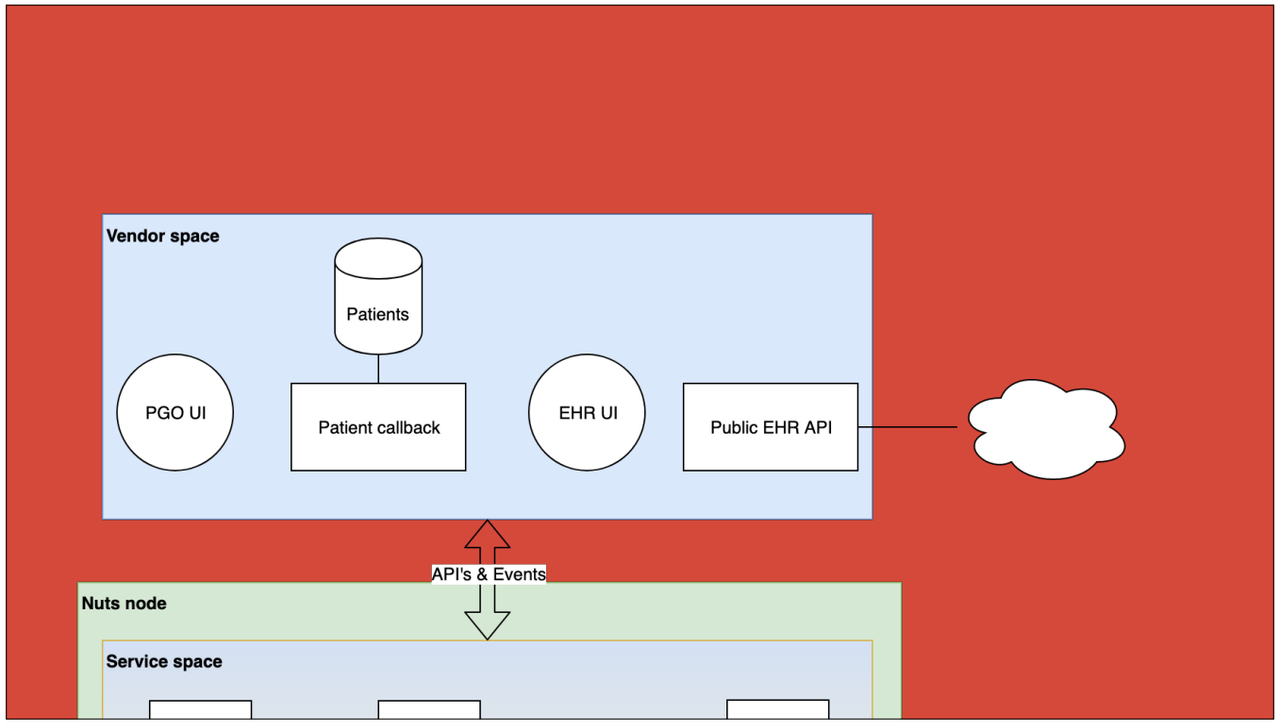

NUTS
Network-in-a-day



Nuts focusses on Identity, Addressing, Consent and logging. Not on data standards or processes.



The nuts architecture from https://nuts-documentation.readthedocs.io.
Read the docs is automatically updated with all code updates.

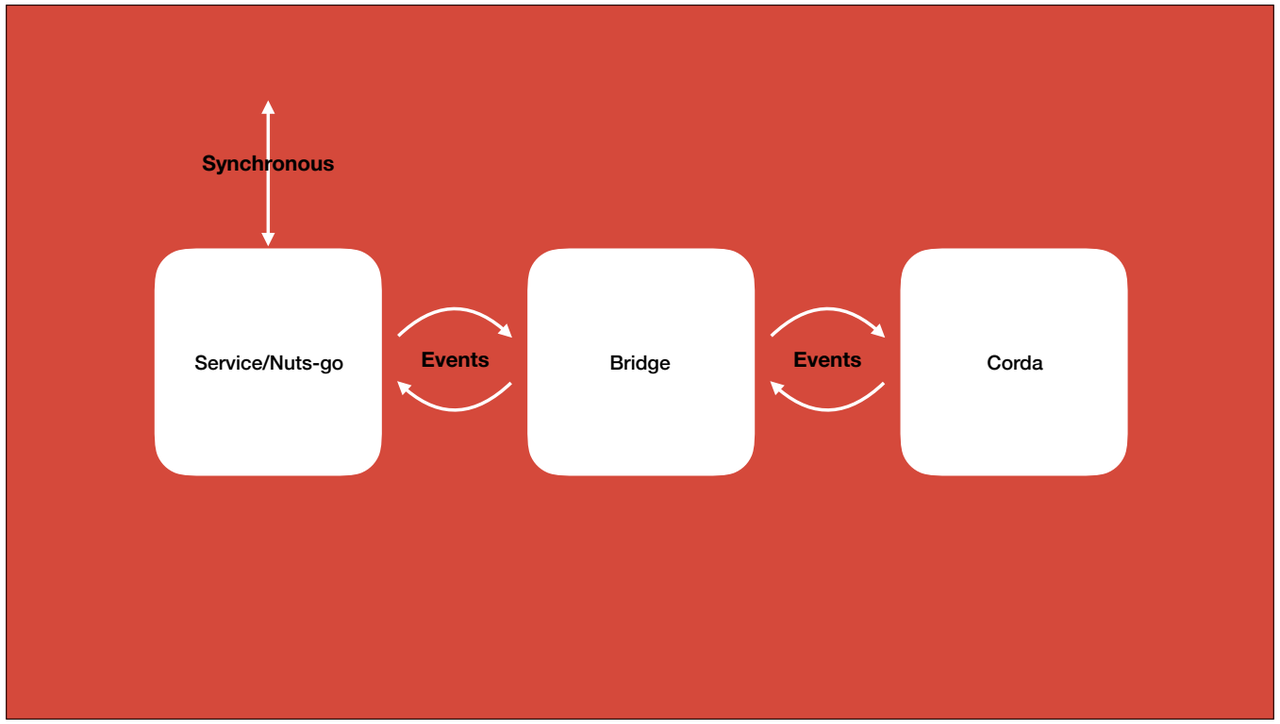

Vendor space
Patients
PGO UI
Patient callback
EHR UI
Public EHR API
API's & Events
Nuts node
Service space

Event based

- Other node may be down or slow
- Pause/Resume
- Retry
- Synchronous to asynchronous transformation
- Event sourcing
- Track progress via Event Store

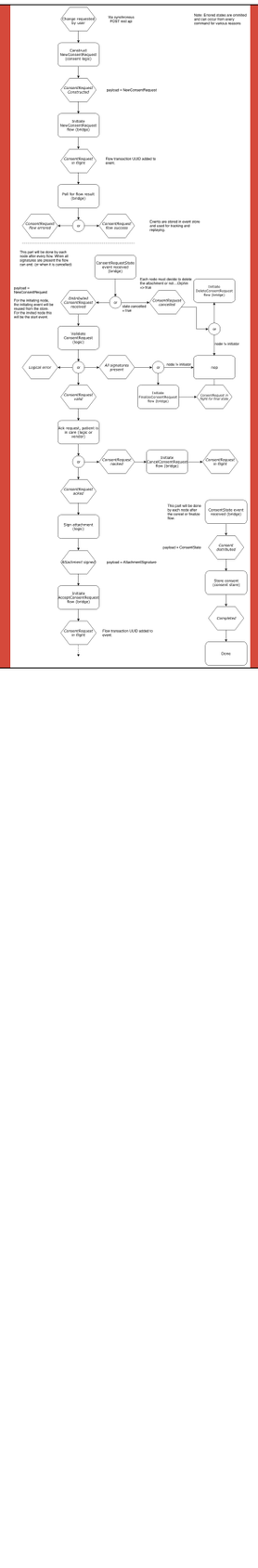
A distributed system must be event based, because it’s unknown what the status is of the other nodes. They can be slow, fast or down. This means that a synchronous user action has to be translated into an asynchronous event. Nuts service space handles this.

Nuts uses event-sourcing with a backup on disk for retries and failover.



The three main components of a Nuts node.

Nuts event state-machine



https://nuts-documentation.readthedocs.io

Registering consent

- POST to nuts-consent-logic API
- Search registry for receiving organisations
- Encrypt
- Send to bridge
- Find corda nodes for receiving parties
- Start corda flow

Different steps done by Nuts components.

Corda

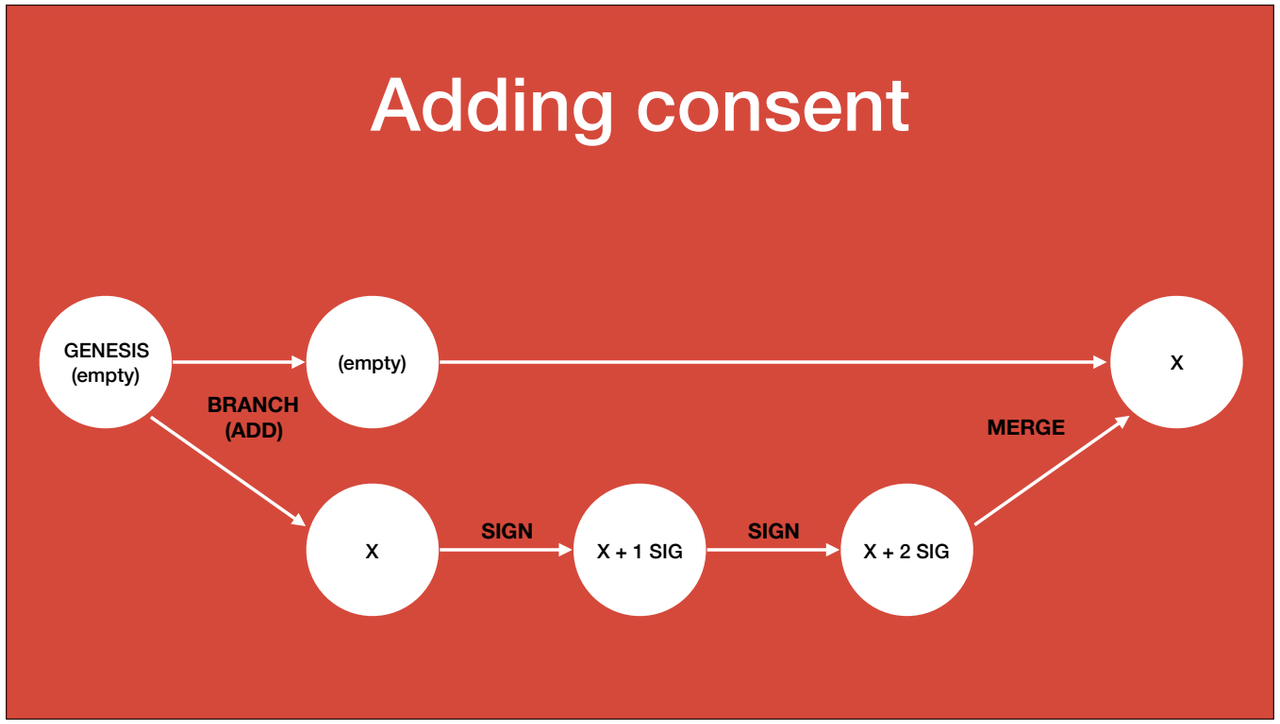
- DLT technology
- Enforces that entire private network runs same software
- Restrict data transfer to specific nodes
- Hide information for specific nodes
- Verify data according to contract
- Consensus
- Event based

For more info https://corda.net

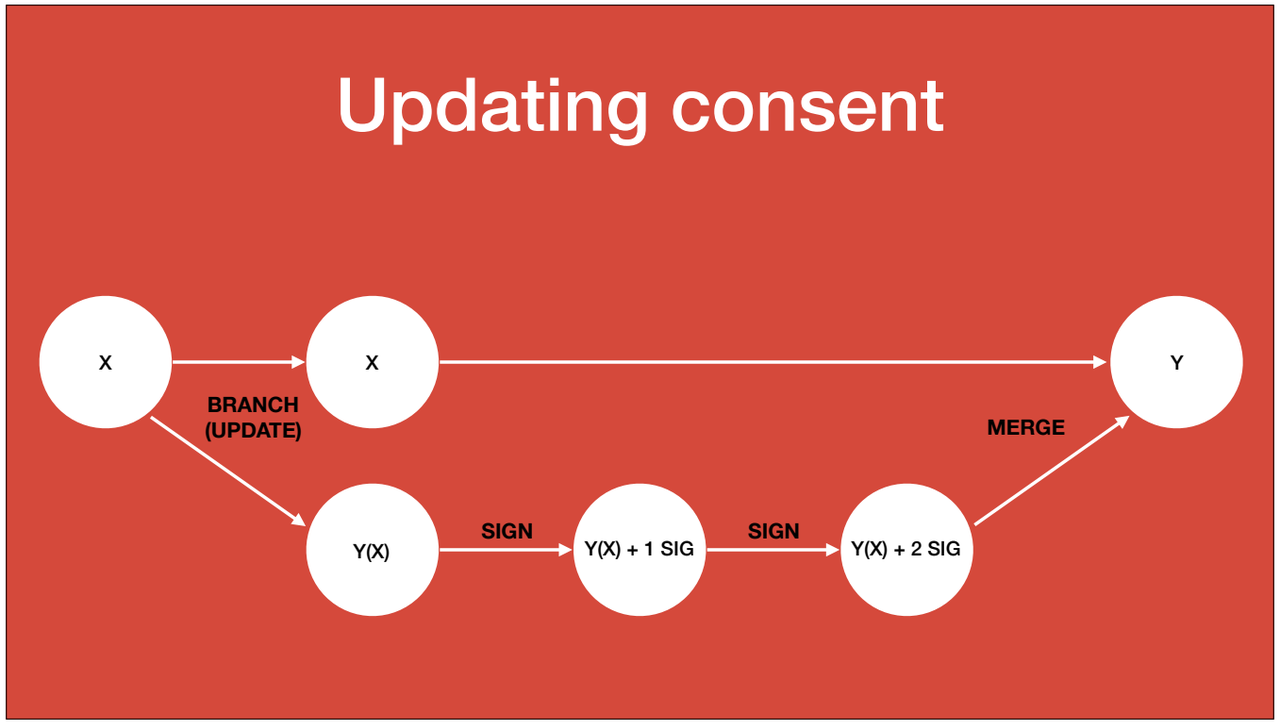
Patient consent

- Unique per custodian, subject, actor
- Multiple active consent records per patient consent:
  - Explicit
  - Implicit like eOverdracht
- Changes over time, but never deleted
- No personal information, just BSN
- Think GIT

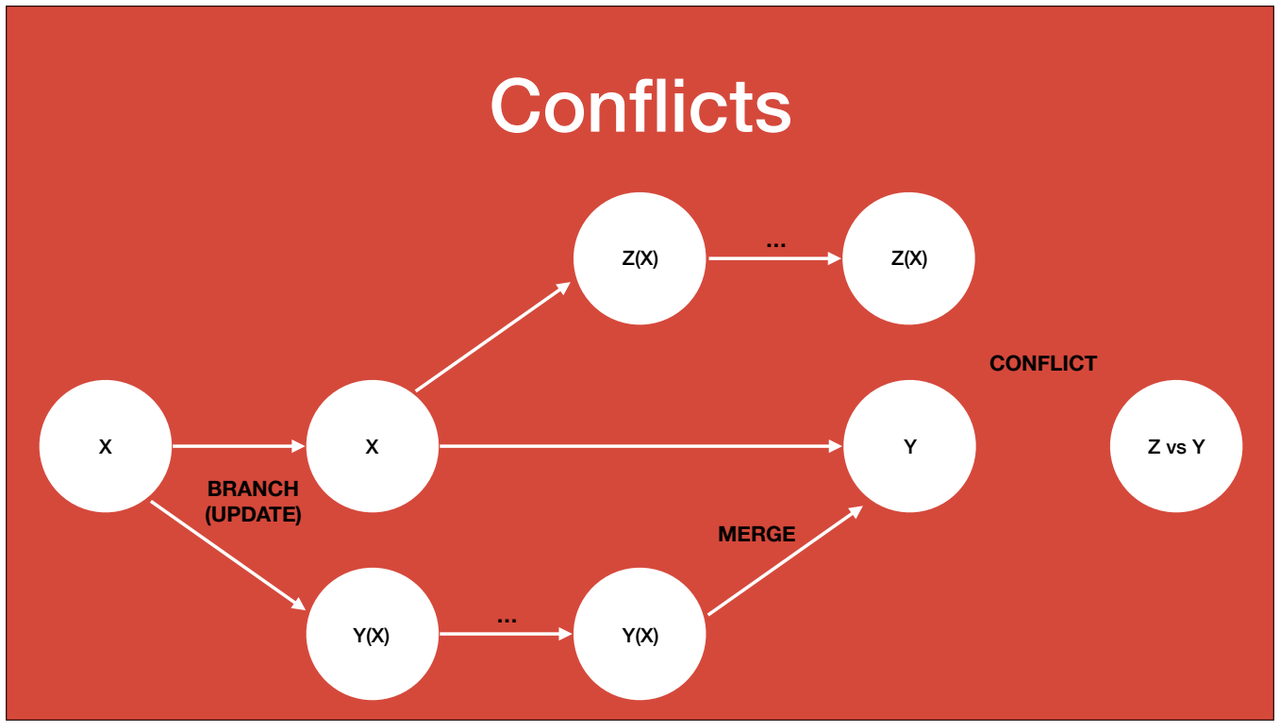
Adding consent



Updating consent



Conflicts

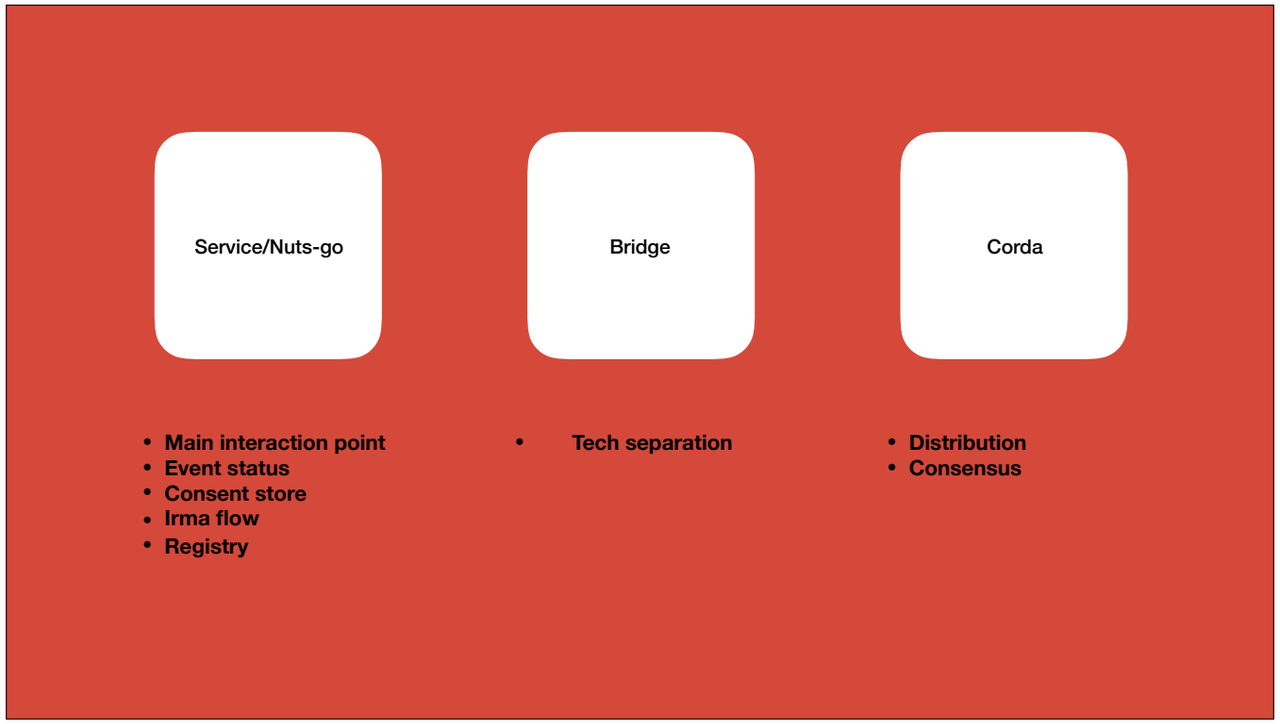


Registry

- Both Corda and Data endpoints
- Care Organisations and their public keys

Public/Private key pairs

- Per organisation
- Encrypts patient consent record
- Signs JWT (containing Irma signature)
- Can be migrated across vendors


Service/Nuts-go
Bridge
Corda
• Main interaction point
• Event status
• Consent store
• Irma flow
• Registry
• Tech separation
• Distribution
• Consensus

Let’s get started

Setup a network together, step by step

In short

- Get a public address for your laptop
- Choose your identities
- Register Corda node
- Update registry
- Start
- Interact

Resources

- https://nuts-documentation.readthedocs.io
- https://github.com/nuts-foundation/nuts-workshops
  - network_in_a_day/participant/
- https://github.com/nuts-foundation/nuts-registry-workshop

cd to nuts-workshops/network_in_a_day/participant/

Github

```
$ git clone git@github.com:nuts-foundation/nuts-workshops
```

This repo contains all sorts of preconfigured files.

Setup ngrok

- https://ngrok.com
- Download & Install
- Signup
- Authtoken: https://dashboard.ngrok.com/auth
- Change token network_in_a_day/participant/ngrok.yml

If you have an auth token in ~/.ngrok somewhere. It won’t work since we use a custom config for starting ngrok.
Since we use high port numbers, firewalls on corporate networks may block stuff.

Start ngrok

```
./ngrok_all.sh
```

(Windows user either run this using bash via powershell or rename to .bat)

ngrok output

```
ngrok by @inconshreveable
(Ctrl+C to quit)

Session Status                online
Account                       Nedap (Plan: Pro)
Version                       2.3.34
Region                        Europe (eu)
Web Interface                 http://127.0.0.1:4040
Forwarding                    tcp://0.tcp.eu.ngrok.io:11192 -> localhost:27886
Forwarding                    http://fa0ba4c2.eu.ngrok.io -> http://localhost:21323
Forwarding                    https://fa0ba4c2.eu.ngrok.io -> http://localhost:21323

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Do not restart mid-session!!

node/node.conf

```
myLegalName="O=Nuts,C=NL,L=Groenlo,CN=nuts_corda_development_CHANGE_ME"
emailAddress="info@nuts.nl"
devMode=true
devModeOptions.allowCompatibilityZone=true
networkServices {
    doormanURL="http://nuts-discovery.eu.ngrok.io"
    networkMapURL="http://nuts-discovery.eu.ngrok.io"
}
p2pAddress="CHANGE"
rpcSettings {
    address="corda:7887"
    adminAddress="corda:7888"
}
```

The p2paddress is without the tcp part of ngrok.
File found in nuts-workshops/network_in_a_day/participant/node/
This is the configuration for the Corda node.

docker-compose-nodes.yml

```yaml
version: "3.7"
services:
  corda:
    image: nutsfoundation/nuts-consent-cordapp:latest-dev
    networks:
      - nuts
    ports:
      - "22222:2222"
      - "27886:CHANGE"
    volumes:
      - "./node/:/opt/nuts/"
    command: "-jar /opt/nuts/corda.jar --network-root-truststore-password=changeit --log-to-console"
    restart: "always"
```

The port for CHANGE is the tcp port used by ngrok (not 27886)

File found in nuts-workshops/network_in_a_day/participant/

node/nuts.yaml

```yaml
verbosity: debug
address: :1323
auth:
  actingPartyCn: CHANGE_ME
  publicUrl: CHANGE_ME
  irmaConfigPath: /opt/nuts/irma
  enableCORS: true
crypto:
  fspath: /opt/nuts/keys
registry:
  datadir: /opt/nuts/registry
  syncMode: github
  syncAddress: https://codeload.github.com/nuts-foundation/nuts-registry-workshop/tar.gz/master
events:
  connectionstring: file:eventstore.db
cbridge:
  address: http://bridge:8080
cstore:
  connectionstring: file:consent.db
```

Nuts-service config file
The actingPartyCn will be used as name in the contract that has to be signed by Irma
The publicUrl is the https url listed by ngrok (with https)
File found in nuts-workshops/network_in_a_day/participant/node/

Register Corda node

```
docker-compose -f docker-compose-initial.yml up

docker-compose -f docker-compose-initial.yml down
```

Node dir contains all kinds of extra files. Particularly a NodeInfo file and some keystores in node/certificates

This has to be done 1 time for key generation and registration to the nuts workshop network
The ‘down’ cleans stuff up.

Startup nuts

```
docker-compose -f docker-compose-nodes.yml up
```

Startup: corda, bridge and service-space

Some checks

```
ssh admin@localhost -p 22222                                    nuts

run networkMapSnapshot
```

```
GET localhost:21323/api/organization/urn:oid:2.16.840.1.113883.2.4.6.1:48000000
```

```
GET localhost:21323/events
```

Password for ssh is ‘nuts’ (without the quotes)
The network map snapshot lists all connected nodes.

Use your favourite REST client to do the other two requests

Generate keys for care organisation

```
POST localhost:21323/crypto/generate?legalEntity=urn:oid:
2.16.840.1.113883.2.4.6.1:XXXXXXXX
```

```
GET localhost:21323/crypto/public_key/urn:oid:
2.16.840.1.113883.2.4.6.1:XXXXXXXX
```

Use your favourite REST client to do the requests. Replace XXXXXXXX with a unique number among participants. Choose one and stick with it.

This will generate a key pair used for encryption and signing

register care organisation

- Fork github.com/nuts-foundation/nuts-registry-workshop
- Clone from own account
- Add entries to:
  - organizations.json
  - endpoints.json
  - endpoints_organizations.json
- Push and Create a PR

Good exercise

organizations.json

```
[
  {
    "name": "Verpleeghuis De Nootjes",
    "identifier": "urn:oid:2.16.840.1.113883.2.4.6.1:XXXXXXXX",
    "publicKey": "-----BEGIN PUBLIC KEY——…\n…——END PUBLIC KEY-----"
  }
]
```

Choose a name
Replace XXXXXXXX with the code chosen.

endpoints.json

```
{
  "endpointType": "urn:nuts:endpoint:consent",
  "identifier": “urn:ietf:rfc:
1779:O=Nuts,C=NL,L=Groenlo,CN=nuts_corda_development_CHANGE_ME",
  "status": "active",
  "version": "0.1.0",
  "URL": "tcp://1.tcp.eu.ngrok.io:CHANGE_ME"
}
```

From node.conf and ngrok output

Replace CHANGE_ME in with chosen CN from node.conf
Replace CHANGE_ME port with port listed by ngrok

endpoints_organizations.json

```
{
  "status": "active",
  "organization": "urn:oid:2.16.840.1.113883.2.4.6.1:XXXXXXXX",
  "endpoint": “urn:ietf:rfc:
1779:O=Nuts,C=NL,L=Groenlo,CN=nuts_corda_development_CHANGE_ME"
}
```

Replace CHANGE_ME in with chosen CN from node.conf
Replace XXXXXXXX with the code chosen.

More checks

```
GET localhost:21323/api/organization/urn:oid:2.16.840.1.113883.2.4.6.1:XXXXXXXX
```

After merge, check the registry for your chosen code

Handout consent


```
{
   "subject": "urn:oid:2.16.840.1.113883.2.4.6.3:999999990",
   "custodian": "urn:oid:2.16.840.1.113883.2.4.6.1:48000000",
   "actor": "urn:oid:2.16.840.1.113883.2.4.6.1:48000001",
   "performer": "urn:oid:2.16.840.1.113883.2.4.6.1:00000000",
   "records": [
      {
         "consentProof": {
            "data": "aa",
            "contentType": "application/pdf"
         },
         "period": {
            "start": "2019-07-01T12:00:00+02:00",
            "end": "2029-07-01T12:00:00+02:00"
         },
         "dataRef": {
             "endpointIdentifier": "urn:nuts:endpoint:fhir",
             "dataIdentifier": "/Patient/544"
         }
      }
   ]
}
```


Done by workshop host

Some checks

```
POST localhost:21323/consent/query

{

  "actor": "urn:oid:2.16.840.1.113883.2.4.6.1:XXXXXXXX",

  "query": "urn:oid:2.16.840.1.113883.2.4.6.3:999999990"

}
```

```
GET localhost:21323/events
```

The upper call queries for registered consent at your node.
The bottom one show the status for received events.

Summary

- Registered node
- Registered care organisation
- Got consent

Next

- Start session using Irma
- Construct JWT
- Get some data

Start session

```
POST localhost:21323/auth/contract/session
```

```
{
  "type": "BehandelaarLogin",
  "language": "NL",
  "legalEntity": "verpleeghuis De nootjes"
}
```

From registry!

The legalEntity name must match the name in the registry.
Output should be visible in console.
Otherwise copy/paste inner json into: https://nl.qr-code-generator.com/

Get result

```
GET localhost:21323/auth/contract/session/{ID}
```

Use the ID from the output of previous call

Get data

```
GET https://nuts-fhir.eu.ngrok.io/t/fhir/Patient/544
Authorization: Bearer 384ausdlfhjp948urcalmthf
```

The resulting bearer token from previous call can be used in a GET request to our hosted endpoint.