Join the Nuts network

Spec v1 use cases exchange DID documents and credentials over the Nuts network (gRPC). This page connects your node to a network and gives it an identity: the *node DID*. The node DID is also your vendor DID in the rest of this book.

Prerequisites

- A running Nuts node, see Running on Docker.
- A network to join.
- A TLS certificate for your node, for use as client and as server, from a CA that the other nodes in the network trust.
- The address of one or more nodes in the network to connect to first: the *bootstrap nodes*.

Networks

The nodes in a network trust the same CAs, have a certificate from one of them, share the same first transaction and use the same versions of the network protocol. There are four networks:

Network	For	Certificates
development	Testing new features. Nodes run the newest, not yet released version	Open CA on GitHub, see step 1
stable	Building your integration and testing with other vendors. Nodes run a released version	Open CA on GitHub, see step 1
test	Acceptance testing with other vendors and customers. Nodes run a released version	PKIoverheid
production	Production	PKIoverheid

1. Get a TLS certificate

Your node only accepts connections from nodes with a certificate from a CA in its *trust store*: a PEM file with the CA certificates the network agreed on. See RFC008 Certificate Structure.

For the development and stable networks, the CA certificate and key are public. Issue a certificate yourself with nuts-development-network-ca:

```
git clone https://github.com/nuts-foundation/nuts-development-network-ca
cd nuts-development-network-ca
./issue-cert.sh development nuts.yourdomain.example
```

This gives three files:

File	Contains
`nuts.yourdomain.example-development.key`	The private key of your node
`nuts.yourdomain.example-development.pem`	The certificate of your node
`truststore-development.pem`	The trust store of the network

For the test and production networks, get a certificate from PKIoverheid Domein Private Services. Put the root certificate ("Stamcertificaat") and all intermediate certificates under "Domein Private Services" in your trust store, and no other certificates.

2. Configure the network

A bootstrap node is a normal Nuts node that other nodes can connect to. Ask a participant of the network for the gRPC address of its node (port `5555` by default), or ask in `#development` on the Nuts Slack which bootstrap nodes there are.

```
didmethods: ["nuts"]
tls:
  truststorefile: /path/to/truststore-development.pem
  certfile: /path/to/nuts.yourdomain.example-development.pem
  certkeyfile: /path/to/nuts.yourdomain.example-development.key
network:
  bootstrapnodes:
    - nuts-development.other-vendor.example:5555
```

See Configuration for all settings, and Recommended deployment for the proxy in front of your node. **v5:** leave out `didmethods`.

Without the `network` settings the node still starts, but alone: a network of one node. That is fine for local development.

3. Wait until your node is in sync

(Re)start your node. It connects to the bootstrap node and gets a copy of the state of the network: the transactions with the DID documents and credentials of all parties. They also contain the

addresses of the other nodes; after a restart your node connects to those too.

```
GET <internal Nuts interface>/status/diagnostics
```

The diagnostics show the transactions coming in. Your node is in sync when no new ones arrive. How long that takes depends on the size of the network and on the CPU, memory and bandwidth of your node.

4. Create your node DID

Some data, such as private credentials, is only exchanged with a node that has proven who it is. Your node does that with a DID it manages: the node DID.

```
POST <internal Nuts interface>/internal/vdr/v1/did
```

Store the `id` of the returned DID document: this is your node DID. The node creates a key for it and keeps it in its key storage. The DID controls its own document, and can issue credentials.

Then add the `NutsComm` endpoint: the gRPC address on which other nodes reach your node.

```
POST <internal Nuts interface>/internal/didman/v1/did/<node DID>/endpoint
```

```
{
  "type": "NutsComm",
  "endpoint": "grpc://nuts.yourdomain.example:5555"
}
```

- The address has the form `grpc://<host>:<port>`.
- The host must be exactly one of the DNS names (SAN) in the TLS certificate of your node. Otherwise other nodes cannot verify your node DID.
- Other nodes ignore addresses with an IP address or a reserved name (RFC2606, e.g. `localhost` or `example.com`).

Set the node DID in the configuration, and restart your node:

```
network:
  nodedid: <node DID>
```

Nodes of the same organization can share one node DID, e.g. in a cluster.

To send private credentials directly from node to node, the DID document also needs a `node-http-services-baseurl` endpoint. The node adds it by itself when it can: see Configuration.

5. Check the connection

```
GET <internal Nuts interface>/status/diagnostics
```

It shows which nodes your node found and which it is connected to, and that your node DID is configured.

Next: Set up DIDs and services.

Revision #1
Created 2026-11-18 12:48:04 UTC by Rein Krul
Updated 2026-11-18 12:48:31 UTC by Rein Krul