Access Node Instantiation and Compliance Guide

This document describes the Access Node integration process, including the steps to follow, the necessary documentation and the verification criteria, as well as clarifying any possible configuration doubts for preparing the environment.

The goal is to provide a clear and concise guide for users participating in this process. To achieve this, the document will be divided into sections or checkpoints. These will detail everything necessary to proceed, including other processes with their respective guides. Failure to meet these checkpoints will prevent us from continuing with the Access Node integration procedure.

Note that for this guide, all the environments mentioned that we will work with will be either a local environment or SBX, but it could be applied to higher environments.

- 1. Registration validation: DOME
- 2. Issuance Validation: LEAR Credential Machine
- 3. Data validation: DLT
- 4. AN Configuration: Secrets
- 5. AN Configuration: Chart.yaml
- 6. AN Configuration: Values.yaml
- 7. Registering the node in the Trusted Lists
- 8. AN preparing the synchronization tests
- Tests to Perform

1. Registration validation: DOME

The first step in the setup process is to confirm that our company is registered in DOME and therefore we have a LEAR Credential Employee with Onboarding permissions available so that we can issue our credentials.

If you are **not registered** or there is any problem with the credential's permissions, consult this guide:

Onboarding Guide

2. Issuance Validation: LEAR Credential Machine

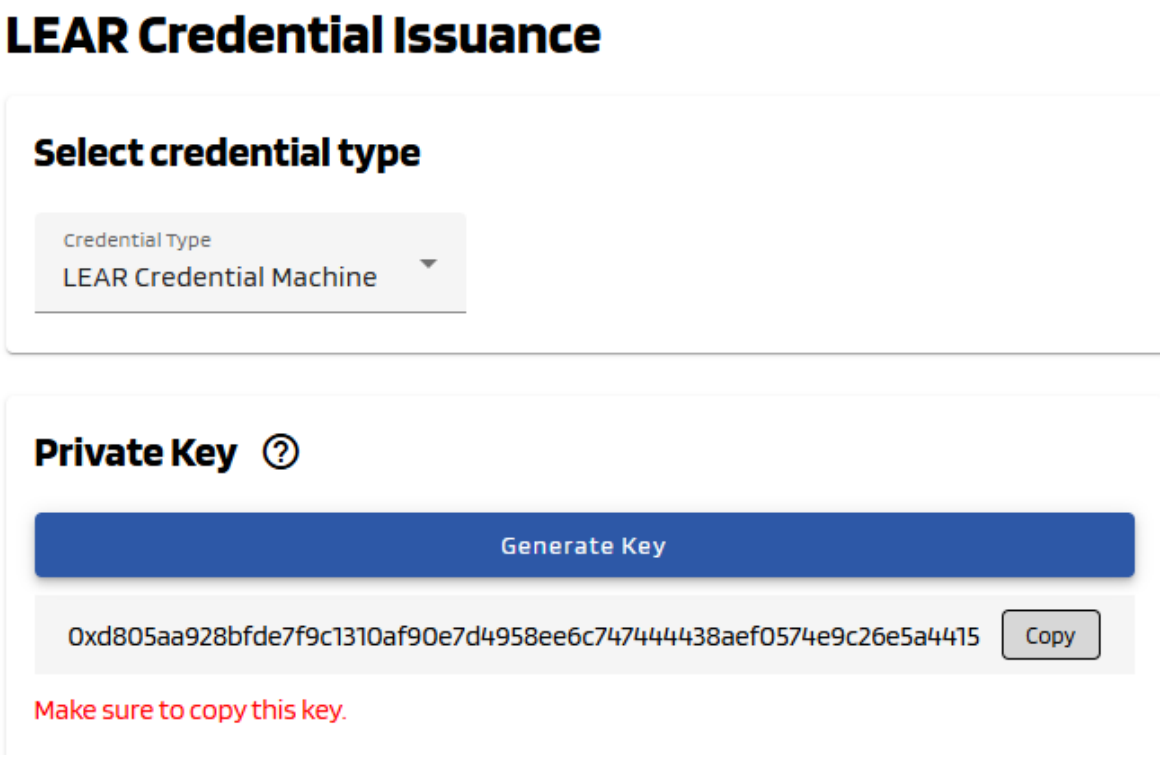
The **second step** is to validate that we currently have a LEAR Credential Machine at our disposal. To do this, we must confirm that we have access to the Wallet and see if our credential has been downloaded.

If you do not have access or do not have the required credentials, please review the credential issuance guide .

Once we have our issued credentials, we will need to consult the next 3 points:

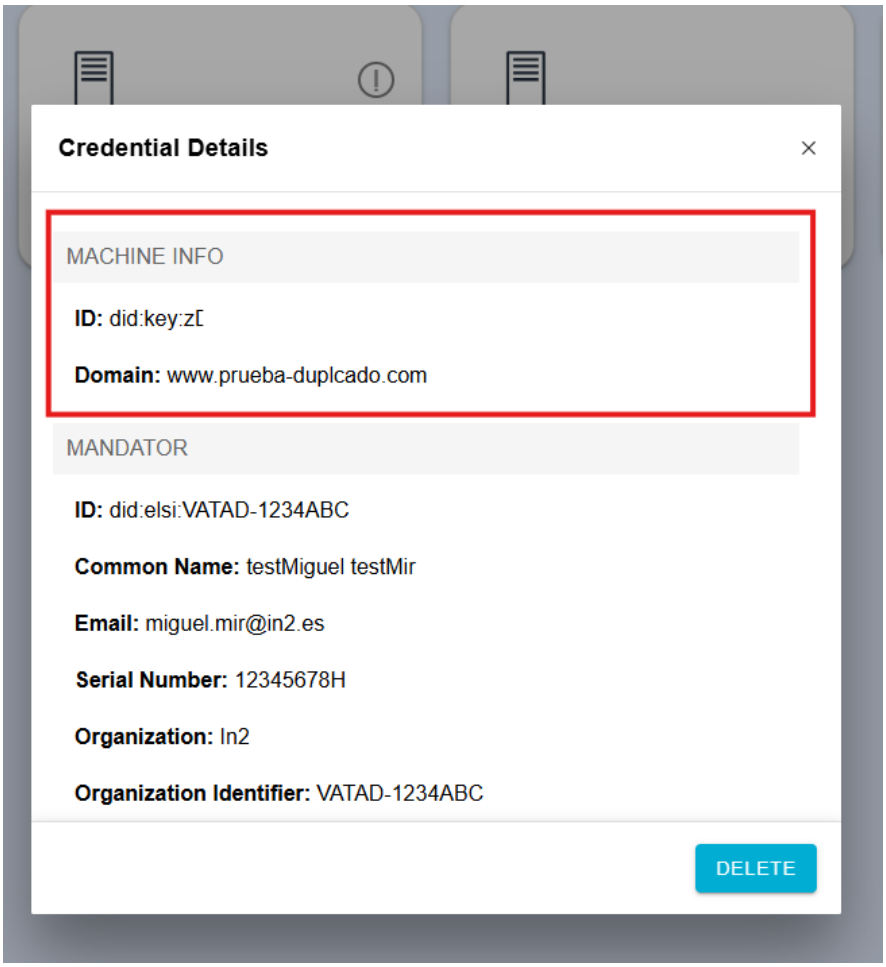
- **Private key** > This is the key generated during the LEAR Credential Machine issuance process . It is important to **save it**, as we will need it later.

Image when we issued a new LEAR Credential Machine

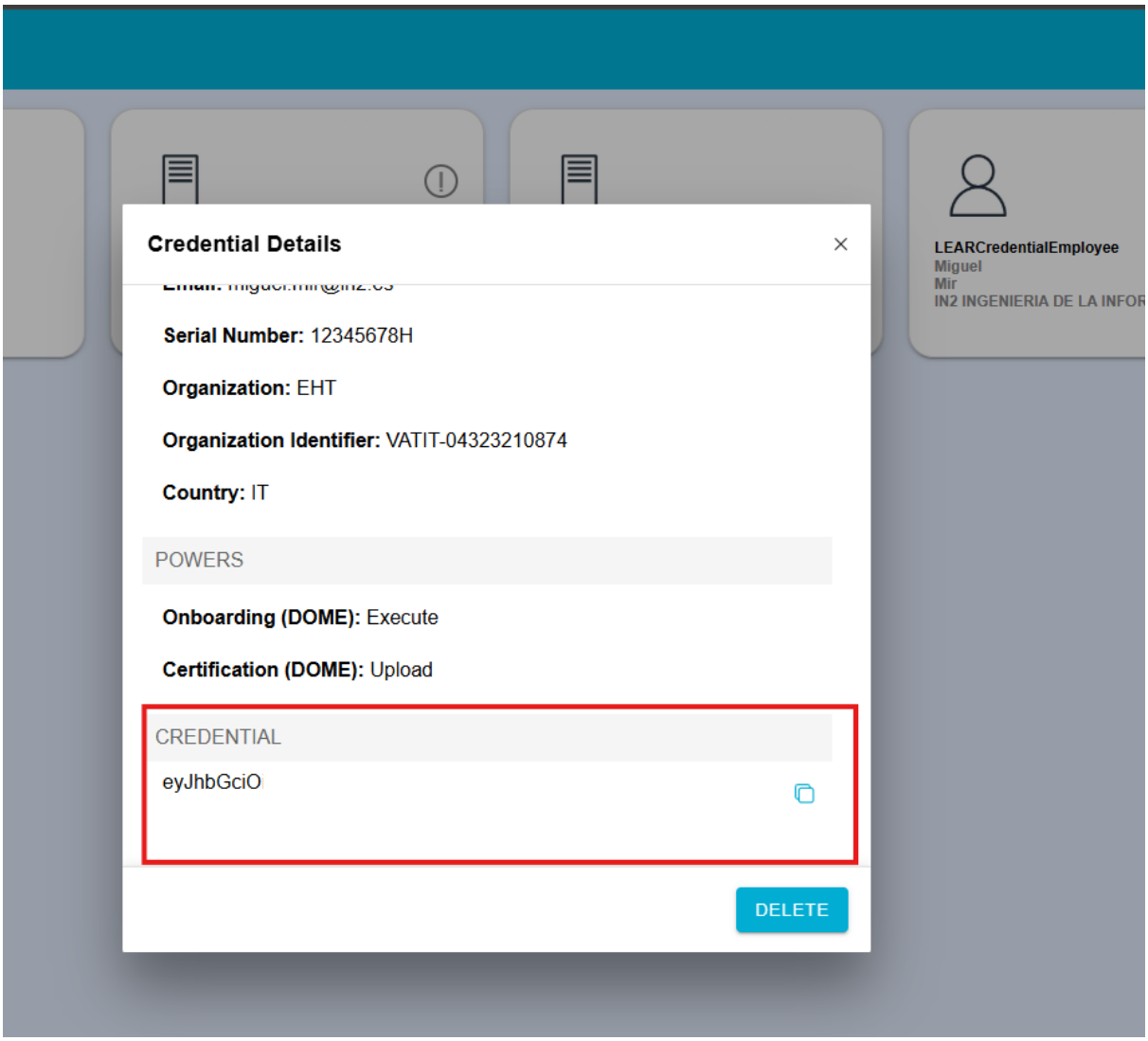


- **DID key** > Auto-generated key at the time of creation that identifies the machine.

After downloaded your credential, you can view your info

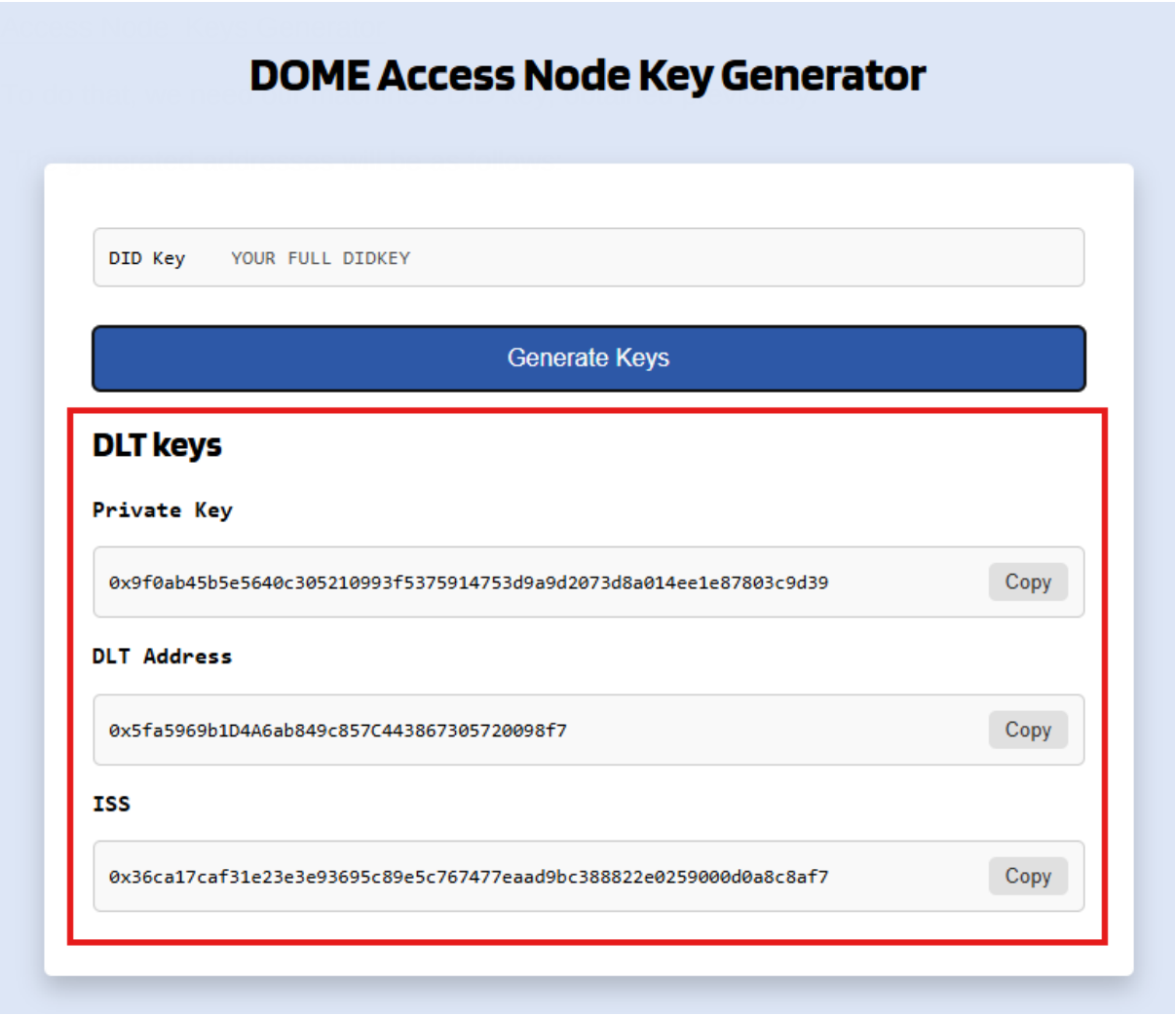


- **LEAR Credential** > Auto-generated token that identifies your credential. You could find it at the bottom of your machine info.


LEARCredentialEmployee
Miguel
Mir
IN2 INGENIERIA DE LA INFOR
Credential Details
Serial Number: 12345678H
Organization: EHT
Organization Identifier: VATIT-04323210874
Country: IT
POWERS
Onboarding (DOME): Execute
Certification (DOME): Upload
CREDENTIAL
eyJhbGciO
DELETE

3. Data validation: DLT

Once we have the issued credential, we need to generate our machine's addresses using this tool: Dome



4. AN Configuration: Secrets

Check this guide if you will work with secrets: Github sealed secrets guide

5. AN Configuration: Chart.yaml

This is how your chart.yaml should look. All necessary dependencies are included in the AN configuration.

Check the last version here:

```
apiVersion: v2
name: access-node
description: Chart holder for ArgoCD
type: application
version: 0.7.31
appVersion: 0.1.0
dependencies:
- name: access-node
  condition: access-node.enabled
  repository: https://dome-marketplace.github.io/access-node
  version: <check the last version>
```

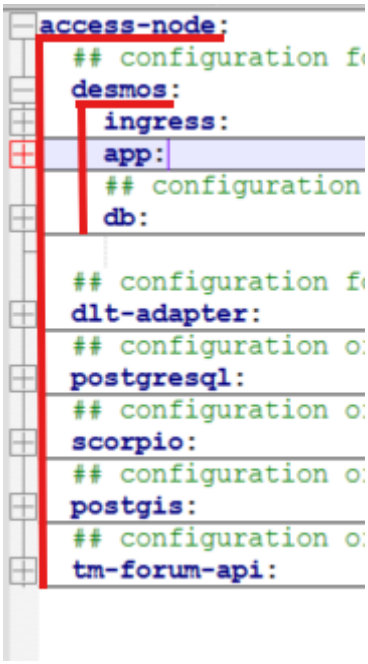
6. AN Configuration: Values.yaml

First, we need to consider whether we've decided to work with secrets (remember rename the file to values.yaml), or not, as the file format may vary slightly.

Values with secrets.yaml

Values without secrets.yaml

First of all, we will check that the file header starts as shown in the image:



Now let’s configure all the blocks.

Ingress

Modify the ingress hots and ensure your domain is properly exposed.

```
desmos:
  ingress:
    enabled: true
    className: nginx
    annotations:
      cert-manager.io/cluster-issuer: letsencrypt-in2-test-issuer
    tls:
      - hosts:
          - <YOUR DESMOS DOMAIN>
        secretName: desmos-tls-secret
  app:
```

For example this is our domain: ***desmos.dome-marketplace-lcl.org***

App

The next thing to check is that we have specified your DID key in the following field:

```
app:
  ## profile (test == dev)
  profile: dev
  logLevel:
    root: INFO
    app: DEBUG
  # internal server port
  internalServerPort: 8080
  ## information about the access-node operator
  operator:
    # -- did of the organization running the node
    organizationIdentifier: did:key:XXXXX
  ## connection information for the context broker
  broker:
```

Next, we will need to review the configured domains:

```
# -- operator external domain
externalDomain:
  protocol: "<YOUR DESMOS DOMAIN PROTOCOL, https RECOMMENDED>"
  domain: "<YOUR DESMOS DOMAIN>"
# -- configuration to set your private key
```

We will need to configure your protocol (in this case it was http but usually it would be https) and the domain of our machine, the URL with which it will be identified.

The next step is to configure the private key, which, if we are working with secrets, will look like this:

```
  existingSecret:
    # -- should an existing secret be used
    enabled: true
    # -- name of the secret
    name: access-node-secret
    # -- key to retrieve the password from
    key: desmos-private-key
```

Or without secrets:

```
# -- configuration to set your private key
privateKey:
  value: <YOUR PRIVATE KEY>
learCredentialMachineInBase64:
  value: <YOUR LEAR CREDENTIAL MACHINE IN BASE64>
```

DB

The next step will be to configure our databases.

```
## configuration of the database to be used by the
db:
  # -- existing secret to retrieve the db password
  existingSecret:
    # -- should an existing secret be used
    enabled: true
    # -- name of the secret
    name: access-node-secret
    # -- key to retrieve the password from
    key: desmos-db-password
```

Here we need to configure our database host; this example uses an embedded database. We'll also need to specify our database details and whether or not we're using secrets.

```
## configuration of the database to be us
db:
  password: <YOUR DESMOS DB PASSWORD>
```

As you can see, without secrets, we will need to specify our database password and disable secrets.

DLT-adapter

The next block is the DLT block; here we will need to configure the data obtained with the dome key generator:

```
## configuration for the dlt-adapter - see https://
dlt-adapter:
  existingSecret:
    enabled: true
    name: access-node-secret
    key: dlt-private-key
  env:
    ISS: "<YOUR DLT ADAPTER ISS>"
```

In case of working without secrets:

```
## configuration for the dlt-adapter - se
dlt-adapter:
  env:
    PRIVATE_KEY: <YOUR PRIVATE KEY>
    ISS: "<YOUR DLT ADAPTER ISS>"
```

Postgresql

For the configuration of our Postgres, we must take into account the configuration of the secrets.

```
## configuration of postgres to be used for the blockchain-connector
postgresql:
  auth:
    # -- existing secret
    existingSecret: access-node-secret
    secretKeys:
      adminPasswordKey: desmos-db-password
      userPasswordKey: desmos-db-password
```

And for the case of no secrets:

```
## configuration of postgres to be used for t
postgresql:
  auth:
    password: <YOUR DESMOS DB PASSWORD>
```

Scorpio

For the Scorpio configuration:

```
## configuration of the context-broker - see https://git
scorpio:
  ## configuration of the database to be used by broker
  db:
    # -- existing secret to retrieve the db password
    existingSecret:
      # -- should an existing secret be used
      enabled: true
      # -- name of the secret
      name: access-node-secret
      # -- key to retrieve the password from
      key: scorpio-db-password
```

And for the configuration without secrets, we would only have to configure the password as shown in the image:

```
## configuration of the context-broker - see htt
scorpio:
  ## configuration of the database to be used by
  db:
    password: <YOUR SCORPIO DB PASSWORD>
```

Postgis

```
## configuration of postgis to be used for scorpio - see https://git
postgis:
  ## auth configuration for the database
  auth:
    # -- existing secret
    existingSecret: access-node-secret
    secretKeys:
      adminPasswordKey: scorpio-db-password
      userPasswordKey: scorpio-db-password
```

And without secrets:

```
## configuration of postgis to be used for sc
postgis:
  ## auth configuration for the database
  auth:
    password: <YOUR SCORPIO DB PASSWORD>
```

This would be all the configuration to take into account to prepare our AN to be able to perform the synchronization tests.

Logically, there are more blocks in the file that we haven't mentioned, but we suggest not editing them without having the necessary knowledge.

7. Registering the node in the Trusted Lists

In order for your AN node to communicate and connect with ours, it must be registered on the following two lists.

Trusted Access node list

Trusted Access Node list file

The format to follow is as follows:

```
- name: <your company name>
  dlt_address: <dlt_address generated above>
```

Trusted Services list

Trusted services list file

The format to follow is as follows:

- clientId : <"did:key of your machine">
  redirectUris : []
  scopes : []
  clientAuthenticationMethods: ["client_secret_jwt"]
  authorizationGrantTypes: ["client_credentials"]
  postLogoutRedirectUris : []
  requireAuthorizationConsent: false
  requireProofKey: false
  jwkSetUrl: " https://verifier.dome-marketplace-sbx.org/oidc/did/did:key:<did key of our machine>"
  tokenEndpointAuthenticationSigningAlgorithm: "ES256"

8. AN preparing the synchronization tests

These are the steps that will be taken before the tests:

8.1. Backup before testing

Perform a **dump backup** of the Scorpio data so we can restore it afterwards and **delete the data** of your DB. Remember to put it out of your docker or it **will disappear** if you restart your AN.

How you could delete the data

```
kubectl get pvc -n <namespace>
kubectl delete pvc postgris-0
kubectl delete pod postgis-0 -n <namespace> --grace-period=0 --force
```

If you change any of your settings, be careful with the PVC's; they could interfere negatively. It's advisable to remove them whenever there's a major change.

8.2. Configuration for testing

Add the **externalAccessNodesUrls** configuration **inside** the **APP block** as shown on the image below, which should be adapted for these tests. (It will be reconfigured later to point to the correct location.)

```
externalAccessNodesUrls:
  enableCustomUrls: true
  customUrls: "https://desmos.dome-marketplace-lcl.org"
```

How it should looks

8.3. Review the values configuration and Trusted Lists registry

Verify that you complete all the steps above and your machine is correctly registered in the Trusted lists

Trusted Lists github

8.4. Install AN

After doing the previous steps, be sure that you installed your AN correctly. I'm leaving these commands here in case they're helpful. Remember that you need to throw these commands inside your values.yaml directory.

```
helm dependency build
helm install myaccessnode . -f values.yaml
```

If you already have the current AN installed, uninstall it to prevent any mixing between the new and old configurations.

```
helm uninstall myaccessnode
```

8.5. Desmos running without errors

Confirm that your desmos pod is working correctly. You could check it with

```
kubectl logs desmos-648bdbc988-pl2ft -n <your-namespace> -c desmos
```

8.6. Send an email

Send an email to domesupport@in2.es informing them that everything is ready to perform the integration tests.

Tests to Perform

We will perform two tests:

- Synchronization with the blockchain
- P2P verification

We will create a temporary entity, one on your node and another on ours, and see if it synchronizes via the blockchain.

We can do it using Postman or using the curl command by postforwarding to Scorpio:

```
kubectl port-forward svc/scorpio 9090:9090 -n <your namespace>
```

Or if we have access to the bae marketplace, from there; For example, this is our configured site: Bae marketplace local

```
curl --location 'http://localhost:9090/ngsi-ld/v1/entities/?type=catalog' \
--header 'Content-Type: application/json' \
--data '{
"id": "urn:catalog:MyDummyTest",
"type": "catalog",
"version": "2.5",
"lastUpdate": "2024-07-09T12:00:00Z",
"lifecycleStatus": "Launched"
}'
```

And we will validate that it has been created correctly using curl

```
curl --location 'http://localhost:9090/ngsi-ld/v1/entities/urn:catalog:MyDummyTest'
```

For P2P validation, we first need to create a Request Point (RP) by removing our node from the Trusted Access Nodes List. Create a new entity as shown above, and we will validate that we have NOT processed that entity.

Run the following command on the node where the entity was created and validate that this time, we have received it.

```
kubectl port-forward svc/desmos 8080:8080 -n <your namespace>
```

```
curl --location 'http://localhost:8080/backoffice/v2/actions/sync' \
--header 'Authorization: Bearer
eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwidXNlIjoic2lnIiwiY3J2IjoiUC0
yNTYiLCJ4IjoiTlJMbXBVTjE3VGsxWUd3WHNuVVlkbG9qTFpxZHhqRVhqNHlTSnZKZWRJMCIsInkiOiJw
V2ozVEdPeGRvTGpzRU1rbDFJQ3RPYmRENm12bzZXaDJTMjdKelNweEZRIn19.eyJodG0iOiJQT1NUIi
wiaHR1IjoiaHR0cHM6Ly9kZW1vcy5kb21lLW1hcmtldHBsYWNlLWxjbC5vcmcvYXBpL3YxL2VudGl0aWV
zLzEyMzQ1Njc4IiwiaWF0IjoxNzM1ODI4MjI2LCJqdGkiOiJkZDQyNzVmYi00MjhmLTQ2NjMtOGI4ZS0zMT
UxMDVkZmQ2ZjIifQ.vZ33bJ2rYFEki5rM6uFmE46t4Ot4BCAIrnQXd-GWNnVwXWPtdhmP9gWa7u5xJS-
ZrNJAkbj4Wh_tjxiSQyRt5Q'
```