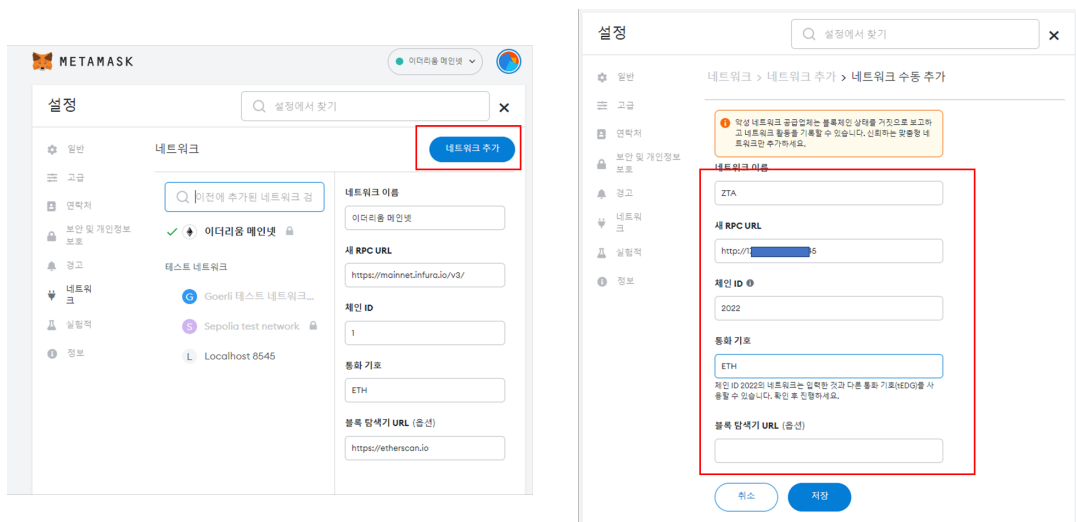


# Installation Manual for the ZITI Agent, IBN GUI Manager and IBN Backend API Services.

## Prerequisite Installation.

Currently the ZTAgent and the IBN GUI Manager are not released as a production build, but they hold all the functionalities of the ZT&T project as a development build. This means that the software binaries have not been compiled but the source code is available for use. In order to run the ZTAgent, IBN API services and the IBN Manager a few steps are required. (NOTE: The blockchain should be up and running, OpenZiti deployment should be up and running)

- Install Metamask on the Machine that will run the IBN GUI from <https://metamask.io/>
- Configure Metamask to include the following data of your blockchain. Point to the IP Address and the port of your blockchain node and the Chain Id of your configuration.



## IBN Manager GUI.

This section also installs the Identity and Policy management smart contracts in the blockchain automatically. **\*\*Be sure to have your blockchain running\*\***. You only require having NodeJS 16 installed in the machine that runs the IBN Manager. **\*\*ONLY LINUX IS SUPPORTED\*\***

1. Make sure you have git installed and clone the project from the github repo:

```
git clone https://github.com/nc1427/IBNPermissioning.git
```

2. Install yarn using npm

```
npm install --global yarn
```

3. Install the dependencies for the project. You need to run this command inside the folder of the repository that was just cloned.

```
yarn install
```

4. Create an environmental file “.env” with the following information. You can leave the first 5 records with the default values, but be sure to add the BESU\_NODE\_PERM\_ACCOUNT with the primary account’s Public Address used during the configuration of the blockchain (Besu Instructions), BESU\_NODE\_PERM\_KEY with the private key of the above account, BESU\_NODE\_PERM\_ENDPOINT with the Http RPC address of the deployed blockchain node and the CHAIN\_ID with the values used during blockchain deployment in the genesis configuration file. Place the .env file in the root of the project folder

```
RETAIN_ADMIN_CONTRACT=false
RETAIN_NODE_RULES_CONTRACT=false
RETAIN_ACCOUNT_RULES_CONTRACT=false
NODE_INGRESS_CONTRACT_ADDRESS=0x00000000000000000000000000000000000009999
ACCOUNT_INGRESS_CONTRACT_ADDRESS=0x0000000000000000000000000000000000008888
BESU_NODE_PERM_ACCOUNT=ADMIN ACCOUNT OF YOUR BLOCKCHAIN
BESU_NODE_PERM_KEY=PRIVATE KEY OF YOUR BLOCKCHAIN
BESU_NODE_PERM_ENDPOINT=http://YOU_BLOCKCHAIN_NODE:PORT
CHAIN_ID=YOUR_BLOCKCHAIN_CHAIN
```

5. Build the Project.

```
yarn build
```

6. Deploy the Identity and Policy Address Smart Contracts. This command will use information on the .env file to connect to the blockchain and deploy the required smart contracts

```
yarn truffle migrate --reset --network besu
```

7. Verify the deployment logs in console and get the values of the Identity Smart Contract Public Address and the Policy Smart Contract Public Address. **\*\*Save them – DO NOT USE THE ONES IN THE IMAGES\*\***

```

Replacing 'AccountRules'
-----
> transaction hash: 0xd0bdd69b3ea377f371e84acfbcb6a800e5f7f46703b109bad8a250ac5cd28cf5c
> Blocks: 0 Seconds: 0
> contract address: 0xc9dCd1f32237399727C414aF6b2C959343E6b76f
> block number: 21
> block timestamp: 1667358034
> account: 0xdb79e9f4244441b47263DA3afAe11c42Cf099964
> balance: 99.7169374
> gas used: 2129259 (0x207d6b)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.04258518 ETH

> Rules deployed with AccountIngress.address = 0xDc7eE17B65bAeA90193adE7D1a7dB0534718c34e
> and storageAddress = 0x75015BbF74E3ebee966B1dd6Da21c198653e453A
>>> Set storage owner to Rules.address 0xc9dCd1f32237399727C414aF6b2C959343E6b76f
> Updated AccountIngress contract with Rules address = 0xc9dCd1f32237399727C414aF6b2C959343E6b76f
RULES 0x72756c6573

```

```

Replacing 'PolicyRules'
-----
> transaction hash: 0xf8532078119f6da392329c15ebd67032fe0aff22231a4fc501239a973f99eed9
> Blocks: 0 Seconds: 0
> contract address: 0x68E77710efE66C4CB4F075A274022d77f8903cc2
> block number: 28
> block timestamp: 1667358041
> account: 0xdb79e9f4244441b47263DA3afAe11c42Cf099964
> balance: 99.54009314
> gas used: 3723991 (0x38d2d7)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.07447982 ETH

> Rules deployed with PolicyIngress.address = 0x6B1c761A5DA82214791862a8C9e487d6630b22AD
> and storageAddress = 0x5577aC94D01bdEB215DAD995996F7efd75678ab2
>>> Set storage owner to Rules.address 0x68E77710efE66C4CB4F075A274022d77f8903cc2
> Adding Initial ROLE TYPES ...
> Initial ROLE TYPES added: [object Object],[object Object],[object Object]
> Updated PolicyIngress contract with Rules address = 0x68E77710efE66C4CB4F075A274022d77f8903cc2
RULES 0x72756c6573 0x68E77710efE66C4CB4F075A274022d77f8903cc2

```

8. Deploy the Enrollment Token Smart Contracts, Session Token Smart contracts and MFA verification smart contracts using REMIX. <https://remix.ethereum.org/>. Using Metamask and the configuration of the deployed Blockchain. Obtain the Public Address of the Deployed Contracts. ONE BY ONE. \*\* DO NOT USE FROM THE PICTURES\*\*



9. Run the IBN MANAGER GUI application. On the main directory of the project run:

```
yarn start
```

IBN Backend (API Services).

This requires having installed the following tools. Git, Anaconda and NodeJS version **16** (**\*\*Very important to keep this version\*\***). **\*\*THESE INSTRUCTIONS ONLY APPLY TO LINUX\*\***

1. Make sure you have git installed and clone the project from the github repo:

```
git clone https://github.com/nc1427/ZT-T.git
```

2. Navigate to the `conda_env` folder of the cloned repo and import the **IBN\_API.yml** file as a conda environment. This will install all the python dependencies automatically. (replace ENVNAME with any name for your environment)

```
conda env create -n ENVNAME --file IBN_API.yml
```

3. Navigate to the project `/src` directory and execute the following command.

```
npm install eth-crypto --save
```

4. Create and `.env` file with the following information. Place the file inside the `/src` folder. Fill the details of the WEBPROVIDER with your blockchain node and port, The ZITIRPC with the location of the SDP controller, THE OTTADDRESS with the public Address of the Enrollment Tokens, The IDENTITYCONTRACT with the public address of the Identity Management Contract, the MFAREPO with the public address of the MFA\_Verification contract, the SESSIONTOKENADDRESS with the public address of the session token contract, the POLICYCONTRACT with the public address of the policy management contract, the CHAINID with the chain-id of the configuration of the blockchain, the ZITIUSERNAME and the ZITIPWD with the user name and password that was created during the installation of OpenZiti. The EMAILUSER and EMAILPWD with the user account the details of the Email that will be used for sending automatic MFA messages.

```
WEB3PROVIDER=http://BLOCKCHAIN_NODE_IP:PORT
ZITIRPC='https://OPENZITI_IP:PORT/edge/management/v1/'

OTTADDRESS=ADDRESS_OTT
IDENTITYCONTRACT=ADDRESS_IDENTITY
MFAREPO=ADDRESS_MFA
SESSIONTOKENADDRESS = ADDRESS_SESSION_TOKENS
POLICYCONTRACT=ADDRESS_POLICY

CHAINID=NODE_CHAIN

ZITIUSERNAME=ZITIUSERNAME
ZITIPWD=ZITIPASSWORD

EMAILUSER=EMAIL_FOR_MFA
EMAILPWD=PASSWORD_FOR_EMAIL
```

5. From inside the /src directory run the IBN API Server

```
python RPCServerLINUX.py
```

6. The server will create your Blockchain Keys automatically and asks for a password for encryption. Input any password but **\*\*Remember It\*\***.

```
Please provide a password for encryption of Blockchain Keys:
```

7. The server will save the Blockchain keys on the machine it is running, the file is encrypted, always remember your password and do not disclose it to anyone. When running the IBN API Server displays in console the Blockchain Public Address associated to its private key. **\*\*You will need this Address later\*\***

```
0xe72E07981EC8eFC118ab0b905E552c06fdE36458
* Serving Flask app "RPCServer" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.0.64:3003/ (Press CTRL+C to quit)
```

8. Remember the IP and port where the server is running, it is necessary for the ZTAgent configuration.

ZTAgent.

This requires having installed the following tools. Git, Anaconda and NodeJS version 16 (**\*\*Very important to keep this version\*\***). You need to install the ZITI EDGE TUNNEL for Windows <https://github.com/openziti/desktop-edge-win/releases/download/2.1.7/Ziti.Desktop.Edge.Client-2.1.7.exe>

or Download the Linux version and save it in the same folder of the ZTAgent.

[https://github.com/openziti/ziti-tunnel-sdk-c/releases/download/v0.20.7/ziti-edge-tunnel-Linux\\_x86\\_64.zip](https://github.com/openziti/ziti-tunnel-sdk-c/releases/download/v0.20.7/ziti-edge-tunnel-Linux_x86_64.zip)

1. Make sure you have git installed and clone the project from the github repo:

```
git clone https://github.com/nc1427/ZT-T.git
```

2. Navigate to the `conda_env` folder of the cloned repo and import the **ZTAgent.yml** file as a conda environment. This will install all the python dependencies automatically. (replace ENVNAME with any name for your environment)

```
conda env create -n ENVNAME --file ZTAgent.yml
```

3. Navigate to the project /src directory and execute the following command.

```
npm install eth-crypto --save
```

4. Create and .env file with the following information. Place the file inside the /src folder. Fill the details of the WEBPROVIDER with your blockchain node and port, The IBNBAKCEND with the location of the IBN API SERVER, THE OTTADDRESS with the public Address of the Enrollment Tokens, The IDENTITYCONTRACT with the public address of the Identity Management Contract, the MFAREPO with the public address of the MFA\_Verification contract, the SESSIONTOKENADDRESS with the public address of the session token contract, the POLICYCONTRACT with the public address of the policy management contract, the CHAINID with the chain-id of the configuration of the blockchain. You can leave the EDGETUNNEL configuration as the default.

```
WEB3PROVIDER=http://BLOCKCHAIN_NODE_IP:PORT
IBNBACKEND='http://172.18.102.81:3003/'

OTTADDRESS=ADDRESS_OTT
IDENTITYCONTRACT=ADDRESS_IDENTITY
MFAREPO=ADDRESS_MFA
SESSIONTOKENADDRESS = ADDRESS_SESSION_TOKENS
POLICYCONTRACT=ADDRESS_POLICY

CHAINID=NODE_CHAIN

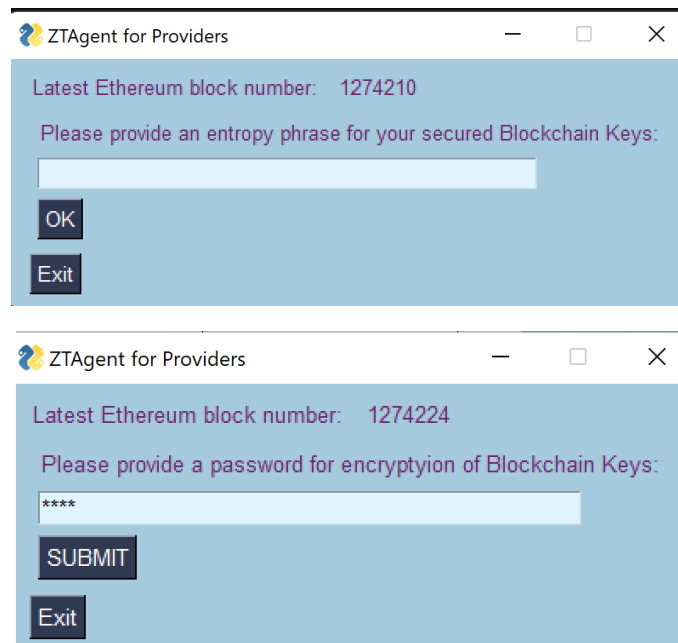
ZITI_IDENTITIES=''
CHAINID=2022

EDGETUNNEL='C:\Windows\System32\config\systemprofile\AppData\Roaming\Net-
Foundry\myId.json'
```

5. From inside the /src directory run the ZITIAgent software. Choose a version for your OS and type of connection.

```
python ZTAgentClient.py
python ZTAgentProvider.py
python ZTAgentClientLinux.py
python ZTAgentProviderLinux.py
```

6. The ZTAgent will request to create the Blockchain Ips that will be encrypted and stored in the machine where the software is running.



7. The ZTAgent will save the Blockchain keys on the machine it is running, the file is encrypted, always remember your password and do not disclose it to anyone.

## LAST STEPS

By accessing the IBN GUI with the Admin accounts of the Blockchain we can add the address of the IBN API server as an administrator key. The IBN Backend requires this for interacting with the whole system.

1. In the IBN GUI go the Admin tabs.

## Admins (6)

[+ ADD ADMIN](#)

Account Address	Status
0x43A5a54f2Ad508eDdFFbFB4A4Dd3612D603e949C	Active
0x1c6cc1aD60e5f88c06044956b0335F5eb3162133	Active

2. Click on add Admin.

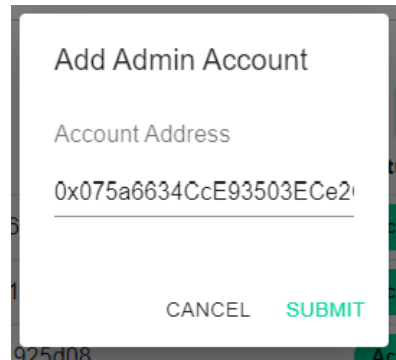
### Add Admin Account

Account Address

Ex: 0xAc03BB73b6a9e1085%

CANCEL SUBMIT

3. Input the Blockchain Public Address of the IBN Backend API server.



Add Admin Account

Account Address

0x075a6634CcE93503ECe2

CANCEL SUBMIT

4. Submit and execute the transaction on Metamask. The IBN API Server is added as an admin of the ZT&T system.