Proximus EnCo Blockchain & CloudEngine

Workshop Guide

-

1 Create & activate your EnCo account

Go to http://enco.io

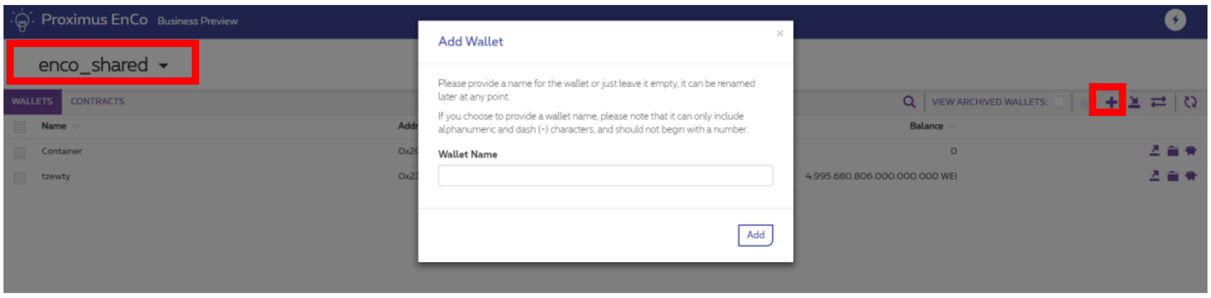
Click on register, follow the procedure.

Once registered and logged in:

- Go to EnCo Marketplace
- Go to EnCo Marketplace, select **CloudEngine API**, register to the Trial Plan. The Trial Plan will allow you to run 150 free transactions. If you exceed this number, you will need to upgrade to the Premium Plan and provide your credit card details, if not associated to your account already. Please note that you will not be charge for CloudEngine transactions under the Premium plan as long as your scripts are triggered by or triggering any of the following EnCo assets:
  - SMS
  - Sensor-as-a-Service (SEaaS)
  - Blockchain-as-a-Service
  - Blockchain Management
  - MyThings API
- Go back to EnCo Marketplace, select **Blockchain Management API**, register to the Beta Plan. The Beta Plan will allow you to run 100 free transactions per day.
- Optionally, go back to EnCo Marketplace, select **SMS API**, select "Plan & Pricing", subscribe to "Trial Plan". You will need to associate a credit card to your account, this is a legal obligation needed for a complete authentication of users (anonymous SMS is not legal in Belgium). You will not be invoiced for SMS usage under the SMS Trial plan.
- Fetch your oAuth2 token in the token management framework:

2 Configure your Blockchain Wallet

In the Blockchain Management (https://blockchain.enco.io/), select the EnCo_shared Blockchain. It is a testing blockchain based on Ethereum, shared amongst all the EnCo users. Of course, if we had to setup this use-case in production, it would make sense to reflect on the Architecture and the ownership of the blockchain nodes. Today we will focus on the development of applications, using an underlying blockchain layer. Create a wallet using the UI (also possible via API call on the Blockchain Management API):



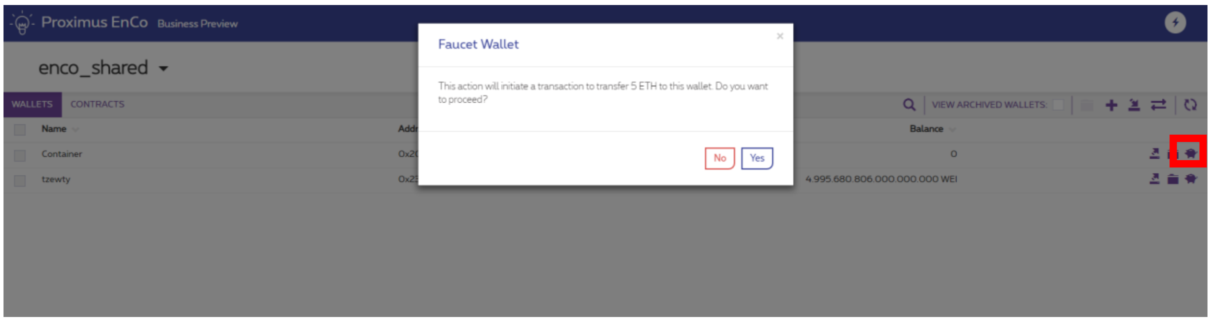
Please provide a name for the wallet or just leave it empty, it can be renamed later at any point.

If you choose to provide a wallet name, please note that it can only include alphanumeric and dash (-) characters, and should not begin with a number.

Name	Addr	Balance
Container	0x2	0
tzewty	0x2	4.995.680.806.000.000.000 WEI

```
curl -sH "Authorization: Bearer MYBEARER" -H "Content-Type: application/json" -X POST https://api.
enco.io/bc-mgmt/1.0.0/chains/enco_shared/wallets?name=XXXXXX
```

In the Ethereum blockchain, you need tokens to be able to write transactions on the Blockchain. So you need to use the faucet API/UI to generate some tokens.

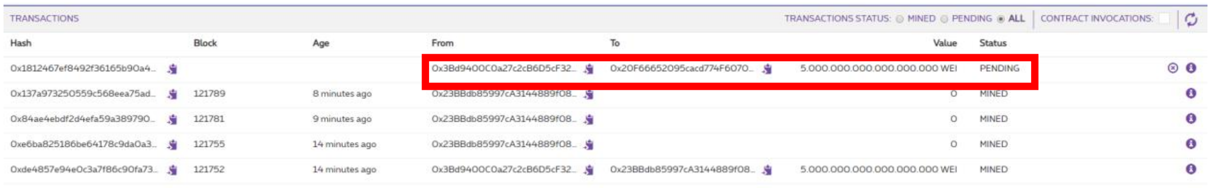


This action will initiate a transaction to transfer 5 ETH to this wallet. Do you want to proceed?

Name	Addr	Balance
Container	0x2	0
tzewty	0x2	4.995.680.806.000.000.000 WEI

```
curl -sH "Authorization: Bearer MYBEARER" -H "Content-Type: application/json" -X GET https://api.
enco.io/bc-mgmt/1.0.0/chains/enco_shared/faucet/XXXXXXX?amount=5
```

Wait and verify when your transaction is mined (less than 15seconds) thanks to the transaction explorer.



Hash	Block	Age	From	To	Value	Status
0x1812467ef8492f36165b90a4...			0x3Bd9400C0a27c2cB6D5cF32...	0x20F66652095cacd774F6070...	5.000.000.000.000.000.000 WEI	PENDING
0x137a973250559c568eea75ad...	121789	8 minutes ago	0x23BBdb85997cA3144889f08...		0	MINED
0x84ae4ebdf2d4efa59a389790...	121781	9 minutes ago	0x23BBdb85997cA3144889f08...		0	MINED
0xe6ba825186be64178c9da0a3...	121755	14 minutes ago	0x23BBdb85997cA3144889f08...		0	MINED
0xde4857e94e0c3a7f86c90fa73...	121752	14 minutes ago	0x3Bd9400C0a27c2cB6D5cF32...	0x23BBdb85997cA3144889f08...	5.000.000.000.000.000.000 WEI	MINED

The “To:” address matches with the address of the wallet you just created. Status should move from “PENDING” to “MINED”.

3 Get your Solidity Smart Contract

The code of the smart contract is deployed on GitHub at the following address: https://github.com/Enabling/Blockchain-containers

4 Deploy your Container smart contract

Let’s first deploy a dummy contract. Let’s define a DummyContainer as object that got an owner, a door (opened or closed) and an ID.

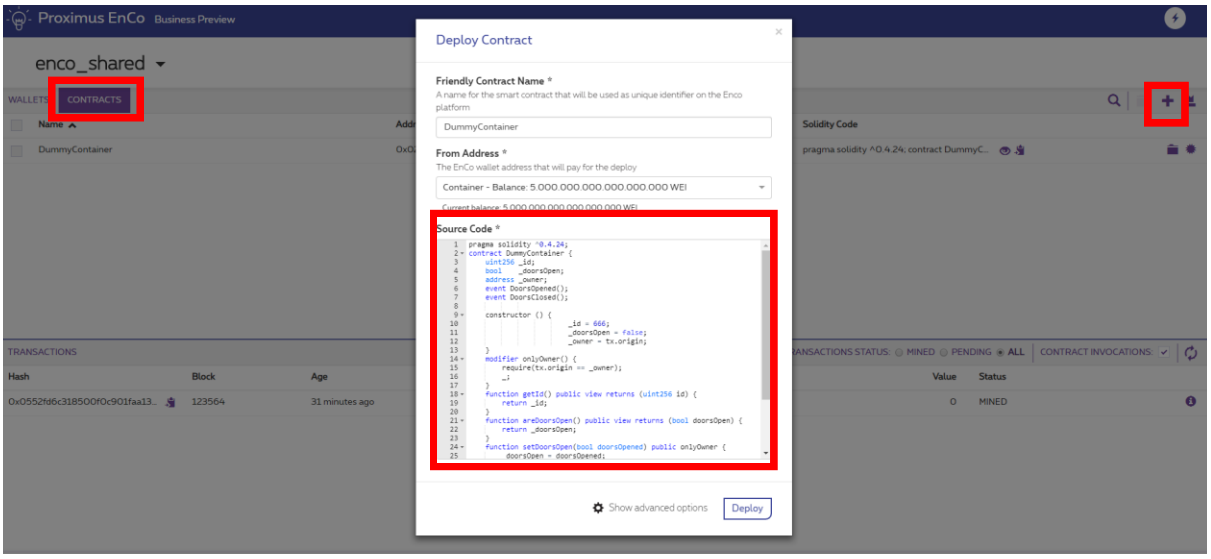
```
pragma solidity ^0.4.24;
contract DummyContainer {
   uint256 _id;
   bool    _doorsOpen;
   address _owner;
   event DoorsOpened();
   event DoorsClosed();

   constructor () {
                  _id = 666;
                  _doorsOpen = false;
                  _owner = tx.origin;
   }
   modifier onlyOwner() {
      require(tx.origin == _owner);
      _;
   }
   function getId() public view returns (uint256 id) {
      return _id;
   }
   function areDoorsOpen() public view returns (bool doorsOpen) {
      return _doorsOpen;
   }
   function setDoorsOpen(bool doorsOpened) public onlyOwner {
      _doorsOpen = doorsOpened;
      if (_doorsOpen) {
         emit DoorsOpened();
      } else {
         emit DoorsClosed();
      }
    }

}
```

Copy this code and deploy it using the Blockchain Management Tool.

Sensitivity: Confidential



Deploy Contract

Friendly Contract Name *
A name for the smart contract that will be used as unique identifier on the Enco platform
DummyContainer

From Address *
The EnCo wallet address that will pay for the deploy
Container - Balance: 5.000.000.000.000.000.000 WEI
Current balance: 5.000.000.000.000.000.000 WEI

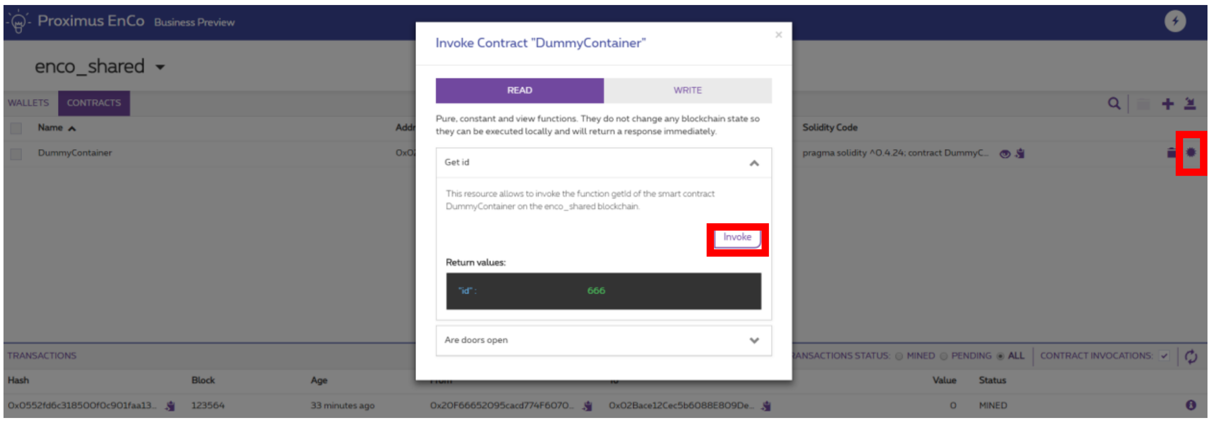
Source Code *

```
pragma solidity ^0.4.24;
contract DummyContainer {
    uint256 _id;
    bool    _doorsOpen;
    address _owner;
    event DoorsOpened();
    event DoorsClosed();

    constructor () {
                  _id = 666;
                  _doorsOpen = false;
                  _owner = tx.origin;
    }
    modifier onlyOwner() {
        require(tx.origin == _owner);
        _;
    }
    function getId() public view returns (uint256 id) {
        return _id;
    }
    function areDoorsOpen() public view returns (bool doorsOpen) {
        return _doorsOpen;
    }
    function setDoorsOpen(bool doorsOpened) public onlyOwner {
         doorsOpen = doorsOpened;
```

Show advanced options   Deploy

Let’s now test the functions of the Smart-Contract:


Proximus EnCo Business Preview
enco_shared
WALLETS
CONTRACTS
Name
DummyContainer
Solidity Code
pragma solidity ^0.4.24; contract DummyC...
TRANSACTIONS
Hash
Block
Age
0x0552fd6c318500f0c901faa13...
123564
33 minutes ago
0x20F66652095cacd774F6070...
0x02Bace12Cec5b6088E809De...
Value
Status
0
MINED

Invoke Contract "DummyContainer"

READ   WRITE

Pure, constant and view functions. They do not change any blockchain state so they can be executed locally and will return a response immediately.

Get id

This resource allows to invoke the function getId of the smart contract DummyContainer on the enco_shared blockchain.

Invoke

Return values:

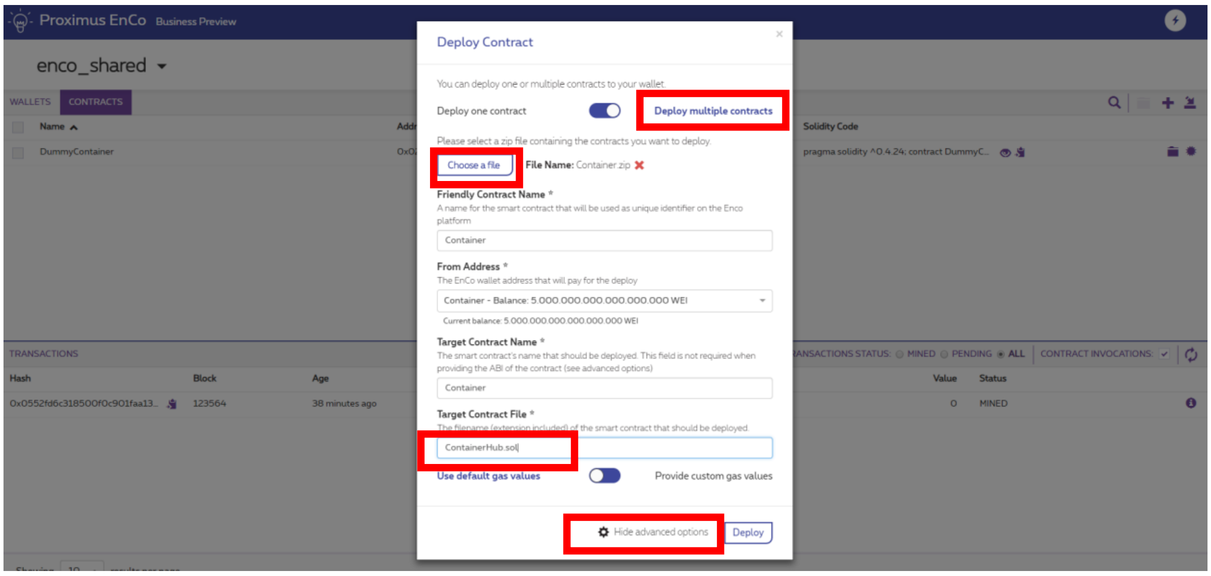
```
"id" :          666
```

Are doors open

This type of smart-contract is good for testing purposes, but it is not very practical when building a complete application. How to keep track of the containers you own? How to validate their status… So, a simple solution to implement is a “ContainerHub”. The code is available on GitHub (https://github.com/Enabling/Blockchain-containers).

To deploy a more complex structure of smart-contracts, we are going to use the advanced deployment features. In the Blockchain Management microsite, go to the Contracts view, and click the “+” to add a new contract. Click on “Show advanced options”. As shown in the screen capture below:

- Click on “Deploy multiple contracts”
- From our github, download both Container.sol and ContainerHub.sol on your computer, and create a zip file including these 2 files
- In the Deploy contract popup, click on “Choose a file” and point to the newly created zip file
- Give the friendly name “ContainerHub”
- Select your blockchain wallet under “From address”
- Give “ContainerHub” as Target Contract Name
- Give “ContainerHub.sol” as Target Contract File.
- Keep the default gas values


Proximus EnCo Business Preview

enco_shared

WALLETS CONTRACTS

Name	Addr	Solidity Code
DummyContainer	0x0	pragma solidity ^0.4.24; contract DummyC...

TRANSACTIONS — TRANSACTIONS STATUS: MINED PENDING ALL — CONTRACT INVOCATIONS:

Hash	Block	Age	Value	Status
0x0552fd6c318500f0c901faa13...	123564	38 minutes ago	0	MINED

Deploy Contract

You can deploy one or multiple contracts to your wallet.

Deploy one contract — **Deploy multiple contracts**

Please select a zip file containing the contracts you want to deploy.

Choose a file — **File Name:** Container.zip

Friendly Contract Name *
A name for the smart contract that will be used as unique identifier on the Enco platform
Container

From Address *
The EnCo wallet address that will pay for the deploy
Container - Balance: 5.000.000.000.000.000.000 WEI
Current balance: 5.000.000.000.000.000.000 WEI

Target Contract Name *
The smart contract's name that should be deployed. This field is not required when providing the ABI of the contract (see advanced options)
Container

Target Contract File *
The filename (extension included) of the smart contract that should be deployed.
ContainerHub.sol

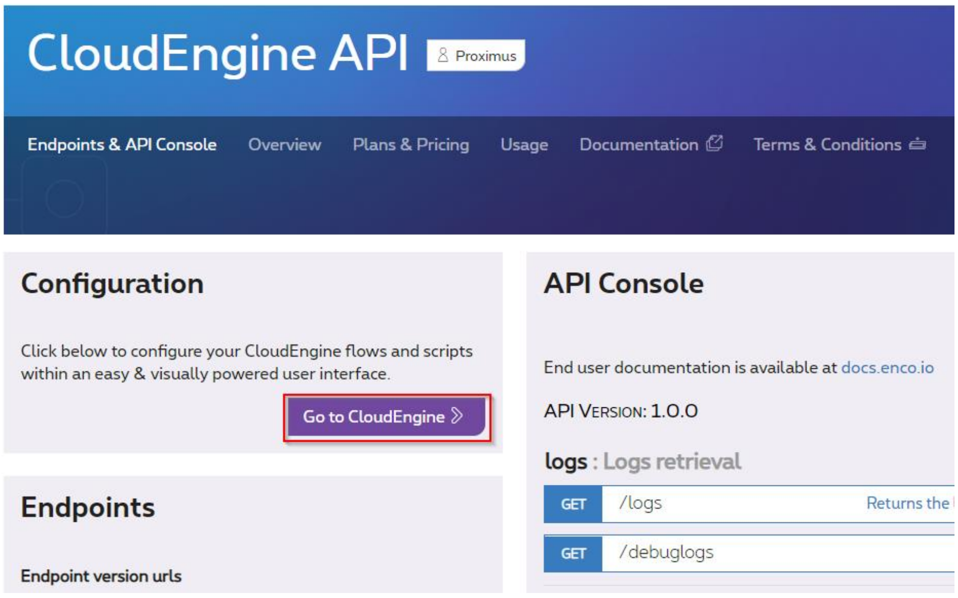
Use default gas values — Provide custom gas values

Hide advanced options — Deploy

5 Use your Smart Contract in CloudEngine

We will use EnCo CloudEngine's capabilities to simulate the reception of data from an IoT device and trigger the smart contract accordingly.

Go to the EnCo marketplace, select the CloudEngine API, click "go to CloudEngine"


CloudEngine API — Proximus

Endpoints & API Console | Overview | Plans & Pricing | Usage | Documentation | Terms & Conditions

Configuration

Click below to configure your CloudEngine flows and scripts within an easy & visually powered user interface.

Go to CloudEngine

Endpoints

Endpoint version urls

API Console

End user documentation is available at docs.enco.io

API Version: 1.0.0

logs : Logs retrieval

GET /logs — Returns the

GET /debuglogs

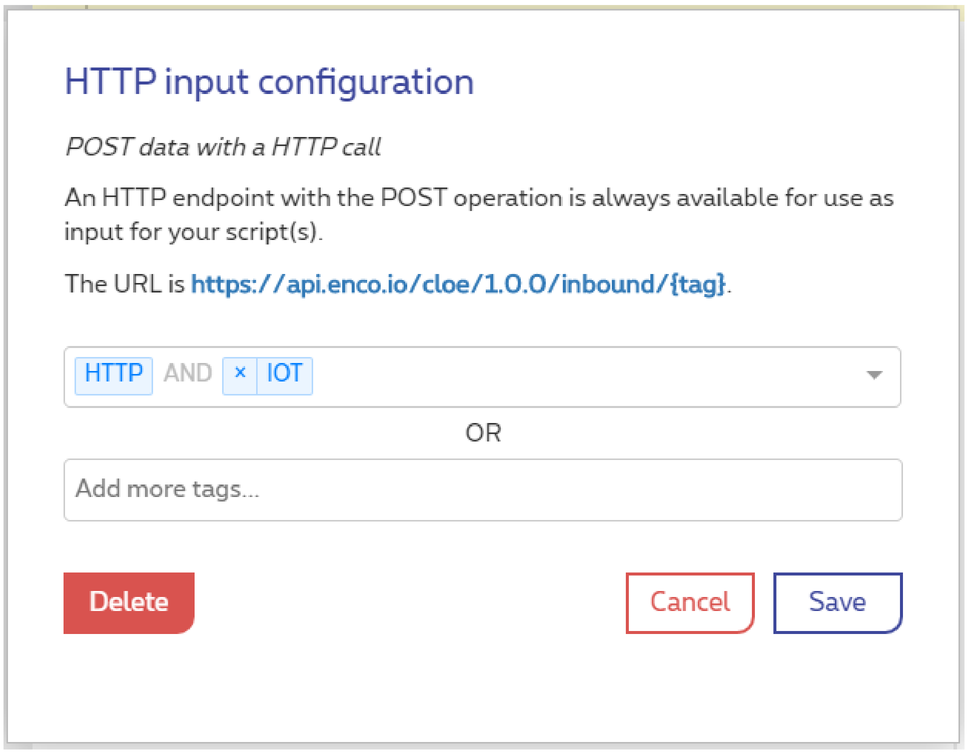
Or go straight to : https://cloudengine.enco.io/

Select "Create", then directly click on "**Convert to Script**" at the bottom left of the screen.

Good to know:

Documentation about CloudEngine script mode and EnCoScript (a derivative from JavaScript) can be found here : https://docs.enco.io/docs/script-mode

Give you script a name, then click on the "**+**" icon in the input column on the left. Select the "**HTTP**" inbound endpoint, then close the selection list. Click on your newly added HTTP endpoint and add a new keyword next to HTTP, such as "IOT" for instance:



HTTP input configuration

POST data with a HTTP call

An HTTP endpoint with the POST operation is always available for use as input for your script(s).

The URL is **https://api.enco.io/cloe/1.0.0/inbound/{tag}**.

HTTP AND × IOT

OR

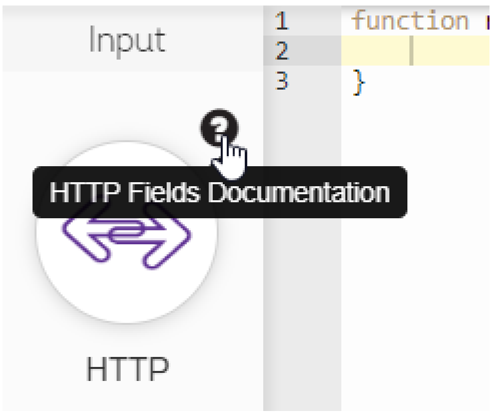
Add more tags...

Delete Cancel Save

Click on Save to close the dialog box. We just told CloudEngine that if there is an incoming data stream with the keywords HTTP and IOT, this specific script will be triggered.

Good to know:

To know the endpoint URL to which a device would send its data to activate this script, just click on the "?" next to your HTTP endpoint :



```
function
 
}
```

In the documentation panel which just opened, scroll down to the end of the text, until you find the title “Usage examples for current flow”, and you will find an URL such as https://api.enco.io/cloe/1.0.0/inbound/HTTP?tags=IOT, which includes all tags you have defined for this specific script. Note that for an external system to be able to do a POST to this URL, it will require to include a hear of type :

Authorization : Bearer xyz

Where the token xyz can be obtained via OAUTH or as a permanent, revocable token from your EnCo project settings. See Token Management under your Default project (top menu bar in the marketplace) or go to https://docs.enco.io/docs/authentication-1 for more info on OAUTH.

Now is a good moment to Save your script (bottom right button).

5.1 First CloE script

Get the first CloE sample script called “CLOE-container-simple” from github (https://github.com/Enabling/Blockchain-containers). Simply copy/paste the content of the file in your CLOE script, overwriting everything.

Spot the following lines and change the <> with your own wallet & contract names, as created in the Blockchain Management part of this workshop. Please note that names are used, and not addresses.

```
blockchain.loadWallet("<YOUR_WALLET_NAME>");

blockchain.loadContract("<YOUR_CONTRACT_NAME");
```

Save your script.

What does this script do?

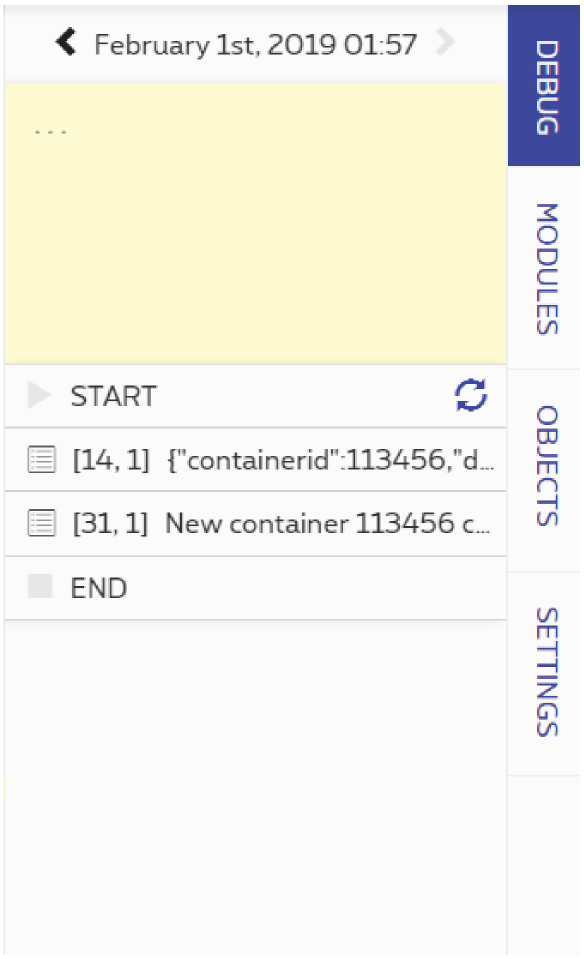
- Load up a few modules (json, array, debug, blockchain)
- Triggers when incoming data arrives on your HTTP endpoint with the tags “HTTP” and “IOT”
- Parse the incoming data to a JSON structure
- Extract the containerid from your incoming data
- Loads up your blockchain wallet and smart contract
- Checks if the containerid is already known on the chain – if not, create a new container with that id.

Time to test! As you don’t have a connected container sending data to your CloE script, we will simulate incoming data. Click on the “**Test inputs**” button, and enter the following in the “BODY” field:

```
{"containerid":123456,"door_sensor_value":false}
```

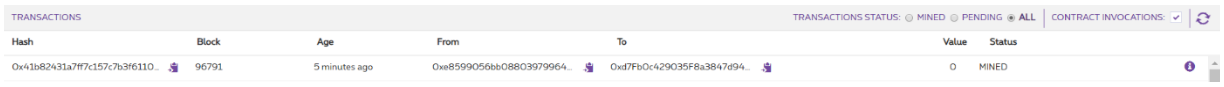
Now click on “Test” and close the popup. You can check the output of the script (we used “debug.log” statements here and there) but selecting the DEBUG view on the right panel. You may need to hit the blue refresh arrow wheel. You can view the complete debug text of each line by clicking on the line.

Sensitivity: Confidential



```
[14, 1]  {"containerid":113456,"d...
[31, 1]  New container 113456 c...
```

Going back to the Blockchain Management Microsite (blockchain.enco.io), you will see indeed new transactions – hopefully already mined or currently mined – resulting in the new container being created.



Hash	Block	Age	From	To	Value	Status
0x41b82431a7ff7c157c7b3f6110...	96791	5 minutes ago	0xe8599056bb08803979964...	0xd7Fb0c429035F8a3847d94...	0	MINED

Clicking on the purple information icon next to the transaction, you can get more details and recognize for instance the wallet which triggered the transaction, the address of the contract being triggered, ...

5.2 Extending your CloE script

Get the second CloE sample script called "CloE-container-full" from Github. Again, copy/paste the content in your CLOE script, overwriting everything.

As in 5.1, update the script with your own wallet and contract names. If you have also subscribed to the SMS API on the EnCo Marketplace, edit the script to include your own phone number in the `object sms =` lines. If you don't have any SMS API subscription, comment out these lines as well as the `sms.send()` ones.

Save your script.

What does this script do more than the previous one?

- Checks if door_sensor_value Boolean message is in the incoming message, if so it will set the container door status accordingly and send you a SMS in case the doors of your container are open

Time to test! just as in 6.1, we will simulate incoming data. Click on the "**Test inputs**" button, and test the following messages, each time checking the resulting script log and transactions on your blockchain on blockchain.enco.io

```
{"containerid":123456,"door_sensor_value":false}

{"containerid":123456,"door_sensor_value":true}
```

Incredible! In about ½ hour, you deployed a smart contract on a chain and started using it with simple scripting on CloudEngine. Using your contract in your own app wouldn't have been much more difficult... Instead of calling the CloudEngine blockchain methods, you would have called it using simple REST API's.

Example:

Setting the door status on a container:

```
$ curl -sH "Authorization: Bearer <YOUR_TOKEN>" -H "Content-Type: application/json" -d '{"id": <YOUR_CONTAINER_ID>, "doors": true}' -X POST  https://api.enco.io/bc-mgmt/1.0.0/chains/enco_shared/contracts/ContainerHub/setDoorsOpen?walletIdentifier=<YOUR_WALLET>
```