REST API blockchain KALIMA

# Introduction

In this tutorial we present the procedure to use to communicate with the Kalima blockchain with the help of a REST API. This last one allows us to use some methods REST to simplify the navigation of a user in his web application (PHP, Node.js, Vue.js, etc…) with the blockchain Kalima.

# Private Docker Register of Kalima System

The REST API of Kalima is given under a docker image placed in a private docker registry belonging to Kalima Systems. This register is secured with https and needs a login to retrieve the image. The link used to access this docker registry private is: [https://docker.registry.kalimadb.com](https://docker.registry.kalimadb.com/) .

The next command allows us to login into the docker registry private of Kalima: **docker login docker.registry.kalimadb.com**

Toget an username and a password, please contact Kalima Systems.

Once the login is done with the docker registry private we need to retrieve the docker image with this command: **docker pull docker.registry.kalimadb.com/kalima-rest-api:latest**

To verify if the image has been retrieved, we can use this command to display all images: **docker images**

# Launch of the API Node.js

Before the launch of the docker image, we need to create a persistent volume.

The command that we need to create a volume persistent is:

**Docker volume** **create volume-name**

To verify the creation of the volume and its location on your PC, we can use this command:

**Docker inspect** **inspect volume-name**

The docker image is now in your PC and the volume is created. To launch the image, you can use this next command:

**docker run --publish 8080:8080 --detach --name kalima-rest-api --mount source=volume-name,target=/home/rcs/jit --env PORT=8080 --env SERIAL\_ID=testDockerImage --env PRIVACHAIN=org.kalima.tuto docker.registry.kalimadb.com/kalima-rest-api:latest**

Possibility of working in https with mode = 1 and passing the crt and key files. To do this, put the .crt and the .key in a certs folder, and move this folder to the docker volume, in the rest folder. To find out where the docker volume is: docker inspect volume-name and see "Mountpoint". Then launch the docker specifying the paths of the certificates:

**docker run --publish 8080:8080 --detach --name kalima-rest-api --mount source=volume-name,target=/home/rcs/jit --env SERIAL\_ID=testDockerImage --env NOTARIES\_LIST=Notaries\_list\_of\_privachain --env PORT=8080 --env MODE=1 --env CERTIFICATE\_CRT=/home/rcs/jit/rest/certs/mycert.crt --env CERTIFICATE\_KEY=/home/rcs/jit/rest/certs/mycert.key docker.registry.kalimadb.com/kalima-rest-api:latest**

Explication of the command:

**. --publish:** this option asks Docker to transfer the traffic entering port 9090 of the host towards port 8080 of the containers. All containers have their own privates' ports, so if you want to reach one from the network, you must transfer the traffic with this option. Or else, security system rules will prevent the traffic network from reaching your container, as a security posture by default. The REST API listens on port 8080 in the container so this value cannot be changed. However, the traffic port (used by your PC) can be customized in terms of the user.

**. --detach:** ask docker to execute this container in the background

**. --name:** specifies a name for this container.

**. --mount:** use a volume persistent volume whose name is **volume-name.** This volume will be used by /home/rcs/jit of docker image.

**. --env:** this optionlets us pass environment variables to the container who launches the image. Up to 6 variables should be passed to the container: “NOTARIES\_LIST“ pass the notary nodes list of the blockchain, “SERIAL\_ID” pass the id who allow us to get the authorization to receive the data from Kalima, “PORT” choose the port we gonna use,”MODE=1” to use the https mode, “CERTIFICATE\_CRT” pass the crit file and “CERTIFICATE\_KEY” pass the key file.

To verify that the container is launch et created, the next command can be used:

**Sudo docker ps**

To look at the logs of the container, we can use this next command:

**Sudo docker logs kalima-rest-api –f**

1. Communication with the Kalima blockchain

The authorization header must be included to authenticate with the REST API, you need a token (Please contact Kalima to get one) to be able to authenticate like this:

**curl -H "Authorization: aea4004084bb9f383803b0b51537b2f0"**

After every request, there are 3 responses:

. 200: success of the request (with a body or none)

. 405: error of the request (with a body like this: Method Not Allowed)

. 407: error of the authentication (with a body like this: Connection refused)

# All GET commands

## GET a transaction

*“To retrieve a transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" '<http://localhost:8080/transaction?address=/sites&key=StAubin>'

**Params:**

address:indicate an address

keys: to choose a specific transaction (keys=\*name\*)

**Response:**

{

"address":"/sites",

"key":"StAubin",

"globalSequence":2402,

"sequence":8,

"previous":7

"body":"eyJpZCI6IlN0QXViaW4iLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEJ5IjoiIiwidXBkYXRlZEJ5IjoiIiwibmFtZSI6IlBvc3RlIHNvdXJjZSBkZSBTYWludCBBdWJpbiIsIm5vVmVyc2lvbiI6IiIsImRlc2NyaXB0aW9uIjoiIiwicHJvcHMiOnt9LCJ0eXBlIjoiIiwic3RhdHVzIjoiIiwib3JnYW5pc2F0aW9uSWQiOiIiLCJ0aHVtYm5haWxVcmwiOiJodHRwOi8vMzcuNTkuMTA4LjE2OjQ1ODcvU3RBdWJpbi8iLCJncmFwaGljc0lkIjoiIiwiem9uZXNJZHMiOlsiR3JvdW5kX0Zsb29yIiwiVW5kZXJmbG9vcl8xIl0sImxvY2F0aW9uSWQiOiJTdEF1YmluIn0=",

"hash":"elXyAtw01KH9EZ+wYXJup/UpP8Ea2m/0HOj3xfnLusw=",

"props":"ZGF0ZT0yMjEwMDUKaXA9LzkwLjQ4LjEwNi4xNjI6NjQzNDEKdGltZT0xNTU0NDYKdHRsPS0xCg=="

}

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

## GET addresses list

*“To retrieve addresses list”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" '[http://localhost:8080/cache/list](http://localhost:9090/cache/list)'

**Response:**

"addresses": [

"/StAubin/Plan/layers",

"/Aubervilliers/Schema/layers",

"/Grigny/Plan/layers",

"/Aubervilliers/Plan/drawings",

"/Montjay/users",

"/Saclay/devices",

"/Essonnes/Plan/drawings",

"/Montjay/zones",

"/Morigny/users",

"/Morigny/zones",

"/Essonnes/Schema/layers",

"/Juine/Plan/drawings",

Etc...

]

}

**Object Keys:**

addresses: array that list all addresses

## GET keys addresses

*“To retrieve keys addresses”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" “<http://localhost:8080/cache/keys?address=/sites&key=StAubin>“

**Params:**

address:indicate an address

keys: to choose a specific transaction (keys=\*name\*)

**Response:**

{

"keys": [

"Aubervilliers",

"Epinay",

"Essonnes",

"Grigny",

"Juine",

"Montjay",

"Morigny",

"Saclay",

"StAubin"

]

}

**Object Keys:**

keys: array that list all keys addresses

## GET Cache Info

*“To retrieve cache info”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" “[http://localhost:8080/cache?address=/sites&key=StAubin](http://localhost:9090/cache?address=/sites&key=StAubin)”

**Params:**

address:indicate an address

keys: to choose a specific transaction (keys=\*name\*)

**Response:**

{

"size":9,

"sequenceMax":8,

"sequenceMin":0,

"address":"/sites",

}

**Object Keys:**

size: amount of info in the cache

sequenceMax: higher sequence in this address

sequenceMin: smaller sequence in this address

address: params that you need to find this transaction

## GET after sequence

*“To retrieve a number of transactions that happens after one transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" “http://localhost:8080/cache/after?address=/Essonnes/devices&n=2&seq=2”

**Params:**

address:indicate an address

n: to choose a specific number of transaction

seq: choose a transaction to start getting transaction who has a sequence that come after it

**Response:**

{

"date":"221005",

"time":"155444",

"messages":[

{

"address":"/Essonnes/devices",

"key":"000425191801DBA8",

"globalSequence":1521,

"sequence":3,

"previous":2,

"body":"",

"hash":"Z0mQiX8qxAw0M6AauaaMLLs1GgRs4w4HsS5px+47iNs=",

"props":"ZGF0ZT0yMjEwMDUKaXA9LzkwLjQ4LjEwNi4xNjI6NjQzNDEKdGltZT0xNTU0NDQKdHRsPS0xCg=="

},

{

"address":"/Essonnes/devices",

"key":"000425191801DBA9",

"globalSequence":1522,

"sequence":4,

"previous":3,

"body":"",

"hash":"vEZfh5pf7xKt2fud49Gmf3F/C30AmrElnOZ4abNisnQ=",

"props":"ZGF0ZT0yMjEwMDUKaXA9LzkwLjQ4LjEwNi4xNjI6NjQzNDEKdGltZT0xNTU0NDQKdHRsPS0xCg=="

}

]

}

**Object Keys:**

address: address of the transaction

body: Payload of the transaction (base 64 encoded)

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

date: transaction date

time: transaction time

messages: all the transaction that has been called by this request in an array

key: name of the transaction

previous: sequence of the previous transaction

hash: associated with the hash function

## GET before sequence

*“To retrieve a number of transactions that happens before one transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" “http://localhost:8080/cache/after?address=/Essonnes/devices&n=2&seq=2”

**Params:**

address:indicate an address

n: to choose a specific number of transaction

seq: choose a transaction to start getting transaction who has a sequence that come after it

**Response:**

{

"date":"221005",

"time":"155444",

"messages":[

{

"address":"/Essonnes/devices",

"key":"000425191801DBA4",

"globalSequence":1518,

"sequence":0,

"previous":-1,

"body":"",

"hash":"eUwHUbPWJmjSA+pQgETjrIx5w41K+QHfWu+9DhlVK/w=",

"props":"ZGF0ZT0yMjEwMDUKaXA9LzkwLjQ4LjEwNi4xNjI6NjQzNDEKdGltZT0xNTU0NDQKdHRsPS0xCg=="

},

{

"address":"/Essonnes/devices",

"key":"000425191801DBA5",

"globalSequence":1519,

"sequence":1,

"previous":0,

"body":"",

"hash":"6gPd+vh4QokVd42ZiZVmjDrOTcC8ZEnn4B0m9TfR5ko=",

"props":"ZGF0ZT0yMjEwMDUKaXA9LzkwLjQ4LjEwNi4xNjI6NjQzNDEKdGltZT0xNTU0NDQKdHRsPS0xCg=="

}

]

}

**Object Keys:**

address: address of the transaction

body: Payload of the transaction (base 64 encoded)

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

date: transaction date

time: transaction time

messages: every transaction that has been called by this request in an array

key: name of the transaction

previous: sequence of the previous transaction

hash: associated with the hash function

# All POST command

## POST a transaction

*“To create a transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" -d “{"address":"/demo/alarms", "key":"test", "body":"dGVzdA==", "ttl":"-1"}'”-H "Content-Type: application/json" -X POST http://localhost:8080/transaction

**Params:**

address:indicate an address

keys: to choose a name (keys=\*name\*)

body: choose the content of our transaction

ttl: to set a timer on the transaction (-1 to stay infinitely)

**Response:**

code http 200 : “Ok”

## Move a transaction

*“To change the location of a transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" -d “{"fromAddress":"/demo/alarms", "toAddress":"/demo/temperature", "key":"test"}” -H "Content-Type: application/json" -X POST http://localhost:8080/transaction/move

**Params:**

"fromAddress":to specify the initial position of the transaction

"toAddress": to specify the finalposition of the transaction

**Response:**

code http 200

# All DELETE commands

## DELETE a transaction

*“To remove a transaction”*

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" -X DELETE “http://localhost:8080/transaction?address=/demo/alarms&key=test”

**Params:**

address:indicate an address

keys: to choose a name (keys=\*name\*) (Optional: If null delete the entire cache)

**Response:**

code http 200

# All LEDGER commands

## User Token

Some of requests to ledger needs a user token. Some of requests returns a limit number of message, even if the params of the request asks for more. Fore exemple, if we want to retrive transactions on the address /x between the sequence 10 and the sequence 10010, the request will return only 20 transactions, because return 10000 transaction at once would be too long and too greedy. However, after using such request, it’s possible to navigate in our previous resarch with requests like nextSearch,previousSearch,firstSearch and lastSearch. The user token is used in that case to find the last transactions sent.

## By Sequence

“Retrive a specific transaction with address and the sequence"

**Request:**

curl -H "Authorization: 1efba00a34dd15ac4217f13d0b903d29" “[http://localhost:8080/ledger/bySequence?address=/addresses&seq=1](http://localhost:9090/ledger/bySequence?address=/addresses&seq=1)”

**Params:**

address : Address of the transaction that we want to retrive

seq : Sequence that we want to retrieve

**Response:**

[ { "address":"/addresses",

"key":"test",

"globalSequence":48,

"sequence":1,

"previous":0,

"body":"dGVzdDI=",

"hash":"mGzuNLtnwfkfW04lXOG7o0TZWpuGY7608RLy8sRRygY=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mwp0dGw9LTEK",

"date":"221108",

"time":"070043"}]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## Between sequence

"Retrieves a transaction list between two sequences on a given address"

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" “http://localhost:8080/ledger/betweenSequences?address=/addresses&fromSeq=0&toSeq=254&pageSize=3&userToken=jerome'’

**Params:**

address: Address of the transaction that we want to retrive

fromSeq: Start of the sequence

toSeq: End of the sequence

pageSize: Size of the « page ». The number of returned transactions can’t be higher that this number

userToken : (return to the section userToken)

**Response:**

[

{

"address":"/addresses",

"key":"test",

"globalSequence":47,

"sequence":0,

"previous":-1,

"body":"dGVzdDE=",

"hash":"ermcFDS5yDZiJwQO3qrTcoPeP5ybY6Tb+u6vSEsGPeA=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mgp0dGw9LTEK",

"date":"221108",

"time":"070042"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":48,

"sequence":1,

"previous":0,

"body":"dGVzdDI=",

"hash":"mGzuNLtnwfkfW04lXOG7o0TZWpuGY7608RLy8sRRygY=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mwp0dGw9LTEK",

"date":"221108",

"time":"070043"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":49,

"sequence":2,

"previous":1,

"body":"dGVzdDM=",

"hash":"S7RQuhLWGygdIEv1ilzMPoksVnIEuXix7hLYICAjBdI=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0NQp0dGw9LTEK",

"date":"221108",

"time":"070045"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## From last

"Retrieves the last n transactions (the most recent transactions)"

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" 'http://localhost:8080/ledger/fromLast?address=/addresses&n=2'

**Params:**

address: Address of the transaction that we want to retrive

n: to choose a specific number of transaction

**Response:**

[

{

"address":"/addresses",

"key":"test",

"globalSequence":48,

"sequence":1,

"previous":0,

"body":"dGVzdDI=",

"hash":"mGzuNLtnwfkfW04lXOG7o0TZWpuGY7608RLy8sRRygY=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mwp0dGw9LTEK",

"date":"221108",

"time":"070043"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":49,

"sequence":2,

"previous":1,

"body":"dGVzdDM=",

"hash":"S7RQuhLWGygdIEv1ilzMPoksVnIEuXix7hLYICAjBdI=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0NQp0dGw9LTEK",

"date":"221108",

"time":"070045"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## Next search

"Return the next page of the previous research. For exemple if pageSize = 10 and that we have previously researched with /ledger/betweenSequences transactions at this address /addresses betwween the sequence 0 and 254 (so we get sequences 0 to 9), this request return transactions whose they sequences go from 10 to 19 "

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" 'http://localhost:8080/ledger/nextSearch?address=/addresses&pageSize=3&userToken=jerome'

**Params:**

address: Address of the transaction that we want to retrive

pageSize: Size of the « page ». The number of returned transactions can’t be higher that this number

userToken : (return to the section userToken)

**Response:**

[

{

"address":"/addresses",

"key":"test",

"globalSequence":57,

"sequence":10,

"previous":9,

"body":"dGVzdA==",

"hash":"a/4yLl+7YCkAextkjb3nm+RZy1DMfV/Bq17oWDkkr7k=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgxMQp0dGw9LTEK",

"date":"221110",

"time":"024811"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":58,

"sequence":11,

"previous":10,

"body":"dGVzdA==",

"hash":"I0BJ0aj71azNW4hhxfGeBUkXGjETOLIHAw+o9oj6MNE=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgxMgp0dGw9LTEK",

"date":"221110",

"time":"024812"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":59,

"sequence":12,

"previous":11,

"body":"dGVzdA==",

"hash":"X4YiCVdoaJO+75xlkDliC+8xY6I1b2FLlQJFvKP52m4=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgxMgp0dGw9LTEK",

"date":"221110",

"time":"024812"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## Previous search

"Return the previous page of the previous research"

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" 'http://localhost:8080/ledger/previousSearch?address=/addresses&pageSize=3&userToken=jerome

**Params:**

address: Address of the transaction that we want to retrive

pageSize: Size of the « page ». The number of returned transactions can’t be higher that this number

userToken : (return to the section userToken)

**Response:**

[

{

"address":"/addresses",

"key":"test",

"globalSequence":47,

"sequence":0,

"previous":-1,

"body":"dGVzdDE=",

"hash":"ermcFDS5yDZiJwQO3qrTcoPeP5ybY6Tb+u6vSEsGPeA=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mgp0dGw9LTEK",

"date":"221108",

"time":"070042"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":48,

"sequence":1,

"previous":0,

"body":"dGVzdDI=",

"hash":"mGzuNLtnwfkfW04lXOG7o0TZWpuGY7608RLy8sRRygY=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mwp0dGw9LTEK",

"date":"221108",

"time":"070043"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":49,

"sequence":2,

"previous":1,

"body":"dGVzdDM=",

"hash":"S7RQuhLWGygdIEv1ilzMPoksVnIEuXix7hLYICAjBdI=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0NQp0dGw9LTEK",

"date":"221108",

"time":"070045"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## First search

"Return the first page of the previous research"

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" 'http://localhost:8080/ledger/firstSearch?address=/addresses&pageSize=3&userToken=jerome

**Params:**

address: Address of the transaction that we want to retrive

pageSize: Size of the « page ». The number of returned transactions can’t be higher that this number

userToken : (return to the section userToken)

**Response:**

{

"address":"/addresses",

"key":"test",

"globalSequence":47,

"sequence":0,

"previous":-1,

"body":"dGVzdDE=",

"hash":"ermcFDS5yDZiJwQO3qrTcoPeP5ybY6Tb+u6vSEsGPeA=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mgp0dGw9LTEK",

"date":"221108",

"time":"070042"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":48,

"sequence":1,

"previous":0,

"body":"dGVzdDI=",

"hash":"mGzuNLtnwfkfW04lXOG7o0TZWpuGY7608RLy8sRRygY=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0Mwp0dGw9LTEK",

"date":"221108",

"time":"070043"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":49,

"sequence":2,

"previous":1,

"body":"dGVzdDM=",

"hash":"S7RQuhLWGygdIEv1ilzMPoksVnIEuXix7hLYICAjBdI=",

"props":"ZGF0ZT0yMjExMDgKaXA9LzE5Mi4xNjguMS4xMjo2MDI2MAp0aW1lPTA3MDA0NQp0dGw9LTEK",

"date":"221108",

"time":"070045"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time

## Last search

"Return the last page of the previous research"

**Request:**

curl -H "Authorization: 543cdaf5-4402-3d41-9aee-9cc3da07c832\_9205" 'http://localhost:8080/ledger/lastSearch?address=/addresses&pageSize=3&userToken=jerome'

**Params:**

address: Address of the transaction that we want to retrive

pageSize: Size of the « page ». The number of returned transactions can’t be higher that this number

userToken : (return to the section userToken)

**Response:**

[

{

"address":"/addresses",

"key":"test",

"globalSequence":98,

"sequence":51,

"previous":50,

"body":"dGVzdA==",

"hash":"tUAUNOStJVdjxygWVQ9fq9rSSgsgz+3OGRzfFq8EQuM=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgyMAp0dGw9LTEK",

"date":"221110",

"time":"024820"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":99,

"sequence":52,

"previous":51,

"body":"dGVzdA==",

"hash":"SucTAbQH4pICK9Wf6Xpht0iMmi+0kzsHcb7Jd8H62t4=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgyMQp0dGw9LTEK",

"date":"221110",

"time":"024821"

},

{

"address":"/addresses",

"key":"test",

"globalSequence":100,

"sequence":53,

"previous":52,

"body":"dGVzdA==",

"hash":"Y9kloQFL1yZQAARJ6ERWzz98j3qchSu0thJExUvWgGg=",

"props":"ZGF0ZT0yMjExMTAKaXA9LzE5Mi4xNjguMS4xMjo2NDgwMQp0aW1lPTAyNDgyMQp0dGw9LTEK",

"date":"221110",

"time":"024821"

}

]

**Object Keys:**

address: address of the transaction

key: name of the transaction

globalSequence: the unique sequence of the transaction in the blockchain

sequence: the unique sequence of the transaction within an address

previous: sequence of the previous transaction

body: Payload of the transaction (base 64 encoded)

hash: associated with the hash function

date: transaction date

time: transaction time