Liquidity SDK Documentation

Release latest

Nov 28, 2018

Contents:

CHAPTER 1

Communities

- Developement communities
  - Github
  - Telegram
- General communities
  - Twitter
  - Telegram
  - Medium

1.1 Liquidity SDK

Liquidity SDK allows you to easily integrate offchain payments and exchanges within your application. It is divided in two different parts. The **wallet daemon** is a self-hosted liquidity wallet management tool for your transactions. The **Liquidity language library** allows you to access wallet daemon directly from your favorite language or from *REST* endpoints.

1.1.1 Architecture

Liquidity is build around hubs creating a link between blockchain and offchain transaction ecosystem. Blockchain remains the ultimate source of trust in case of conflict while hub manage offchain state of network participants. To leverage the complexity of an active state management, the wallet daemon is querying the hub to get its last state. It serves this up-to-date state through REST endpoint the language library is using.

To give a more concrete example, the flow to perform a full transfer to a specific user is given below. Language Library is considered to be the end user.

1. The user ask the daemon to perform a transfer

2. The daemon creates the transfer
3. The daemon sends the transfer
4. At regular interval, the daemon checks if the hub has included the transfer
5. When the transfer is included in an update, the daemon notice it
6. When the user checks if there is any pending transfer, none is returned because its transfer has been performed

If something bad happends during the process, the user is able to contact the daemon and perform security associated operations.

1.1.2 Wallet daemon

Liquidity network provides you with a non custodian payment system. Its core component is the wallet daemon through which the user can perform operations on the network while remaining in control of its funds. For applications looking to implement offchain payments, this component handles communication with liquidity hub, checking it doesn't behave badly.

In term of technology, the daemon is a docker container that synchronises with the hub and provides various endpoints. All endpoints are using the internal state of the daemon. Because the daemon is hosted by the user and has the knowledge of its private key, it is able to sign offchain transfer and leverage the complex verification process that takes place.

1.1.3 Language library

Liquidity language library is a convenient way to communicate with the daemon. It wraps the provided REST API using language specific features. For now, the language library is only available for Node.JS. If you have build an implementation for a language you love, you can submit an issue on liquidity sdk repo.

In the transfer sequence described above, the user has to call the daemon on a regular basis in order to know if it still has any transfers pending. This active wait is not convenient and doesn't integrate well within an application flow. To leverage it, Node.JS library has created a transfer method that returns a promise which is resolved when the transfer has been performed.

```
const liquidity = require('liquidity-sdk')

const to = '0x627306090abaB3A6e1400e9345bC60c78a8BEf57'
const amount = 32

const performedTransfer = await liquidity.transfers.send(to, amount)

console.log(`Tranfer has been ${performedTranfer}`)
```

1.2 Getting started

1.2.1 Pre-requisite

- Docker: Official website

  Docker is a container management system. It supports Windows, mac and most linux distributions. Select the one that applies to your case and follow the instructions.

- Docker-compose: Official website

  Docker compose allows advanced container management. It is required to build the daemon properly.

- A Liquidity registered wallet: Official website

  This wallet will be managed by your daemon. Click on `Add New Wallet`, select the wallet you want register and click on `Save`. Then, open your newly created wallet, click on register and perform Metamask related actions.

1.2.2 Installation

In order to get the daemon up and running on your machine, you must follow the instructions listed below.

1. Clone Liquidity SDK repository

```
git clone git@github.com/liquidity-network/liquidity-daemon
```

Alternatively, you can download it from here

2. Configure your wallet instance

A template of the configuration is provided within `config.template.json`. You have to copy it into `config.json` and edit the configuration.

```
cp config.template.json config.json
```

Your configuration file should look like the following

Mainnet

```
{
  "ETHEREUM_WALLET_PRIVATE_KEY": "0x...",
  "ETHEREUM_NODE_URL": "https://mainnet.infura.io",
  "ETHEREUM_NETWORK_ID": "1",
  "HUB_CONTRACT_ADDRESS": "0x46a1657f132030476496112126Be35384bdDDDAD",
  "HUB_PROVIDER_URL": "https://public.liquidity.network"
}
```

Rinkeby Testnet

```
{
  "ETHEREUM_WALLET_PRIVATE_KEY": "0x...",
  "ETHEREUM_NODE_URL": "https://rinkeby.infura.io",
  "ETHEREUM_NETWORK_ID": "4",
  "HUB_CONTRACT_ADDRESS": "0x140D29486BABCbD57C63edCCCbf2EC92ffBfEc7c",
  "HUB_PROVIDER_URL": "https://rinkeby.liquidity.network"
}
```

3. Start your wallet

```
docker-compose up -d
```

This command will start your wallet daemon (`docker-compose.yml` file located at the project's root directory). It can take some time for it to synchronise with the hub, especially at first launch.

4. Celebrate

You're done with the installation! The SDK is self-hosted on your machine and is accessible under `localhost:3600`.

You can try accessing https://localhost:3600/wallet where you should see the current state of your wallet.

1.2.3 Documentation

The documentation you are about to read is structured in categories. Each category has a list of endpoints made accessible by the wallet daemon. These endpoints are documented with what they provide, how to call them, what are their result and an example is provided.

If you spot any issues, please post an issue on our github repository

1.3 Transfers

Transfers endpoint manages wallet transfers. You should use it to send and checks the state of your transfers. It saves all transfers you make in a database. Therefore it is possible to retrieve your funds if the hub goes down, satisfying the non-custodian property.

1.3.1 Send

Send a transfer to `:recipient` of `:amount` wei.

Endpoint

`POST /transfers`

Request

Name	Required	Description	Default Value	Example
`recipient`	required	Ethereum address to sent the transaction to		*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`amount`	required	Amount to be transfered in *wei*		*1000000000 000000000*

Response

Name	Required	Description	Default Value	Example
`sender`	required	Ethereum address used to perform the transaction		*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`recipient`	required	Ethereum address of the recipient		*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`amount`	required	Amount to be transfered in *wei*		*1000000000 000000000*
`created_on`	required	Date the transaction has been performed (ISO format)		*1970-01-01 T00:00:00.0 00Z*
`nonce`	required	Identifier generated from transaction		*1270040570 *
`transactionId`	required	Unique identifier of the transaction		*1*
`status`	required	Status of the transaction		*confirmed, pending*

Example

```
POST /transfers HTTP/1.1
Content-Type: application/json

{
    "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
    "amount": 1
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "sender": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
   "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
   "amount": "1",
   "created_on": "1970-01-01T00:00:00.000Z",
   "nonce": "1270040570",
   "txId": 1,
   "status": "pending",
}
```

1.3.2 List

List all transfer performed by the daemon during this round. Filters can be applied.

Endpoint

GET /transfers

Request

Name	Required	Description	Default Value	Example
`count`	optional	Amount to be transfered in *wei*	100	*50*
`recipient`	optional	Ethereum address to sent the transaction to		*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`sender`	optional	Ethereum address used to perform the transaction	SDK's ethereum address	*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`amount`	optional	Amount to be transfered in *wei*		*1000000000 000000000*
`transactionId`	optional	Unique identifier of the transaction		*1*
`status`	optional	Status of the transaction	confirmed	*confirmed, pending*

Response

Array

Name	Required	Description	Default Value	Example
`recipient`	required	Ethereum address to sent the transaction to		*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`sender`	required	Ethereum address used to perform the transaction	SDK's ethereum address	*0x62730609 0abaB3A6e14 00e9345bC60 c78a8BEf57*
`amount`	required	Amount to be transfered in *wei*		*1000000000 000000000*
`transactionId`	required	Unique identifier of the transaction		*1*
`status`	required	Status of the transaction		*confirmed, pending*
`nonce`	required	Identifier generated from transaction		*1270040570 *
`created_on`	required	Date the transaction has been performed (ISO format)		*1970-01-01 T00:00:00.0 00Z*

Example

```
GET /transfers?status=pending HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
   {
      "sender": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
```

(continues on next page)

(continued from previous page)

```
        "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
        "amount": "1",
        "created_on": "1970-01-01T00:00:00.000Z",
        "nonce": "1270040570",
        "txId": 1,
        "status": "pending",
    }
]
```

1.4 Invoice

It is useful for merchant to generate a transaction that has to be performed by its client. To that end, Liquidity SDK allows you to generate invoices. These are prefilled transactions that can be used by any liquidity enabled wallet.

When an invoice is paid, a regular transaction is performed and the invoice is resolved.

1.4.1 Generate

Generate an invoice of `:amount` wei. If no `recipient` is provided, the ethereum address of the wallet daemon is used.

Endpoint

`POST /invoices`

Request

Name	Required	Description	Default Value	Example
`amount`	required	Amount to be transfered in *wei*		*1000000000000000000*
`recipient`	optional	Ethereum address to sent the transaction to	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`details`	optional	Details associated to the transaction	''	*"A liquidity transaction "*
`currency`	optional	Token to be used for payment	ETH	*To be implemented*

Response

Name	Required	Description	Default Value	Example
`uuid`	required	Amount to be transfered in *wei*		57056981-32b4-422a-9acb-c03ac4a12404
`destinations`	required	Array of destinations		[{ networkId: 1, contractAddress: 0xac8c3D5242b425DE1b86b17E407D8E949D994010, walletAddresses: [0x627306090abaB3A6e1400e9345bC60c78a8BEf57] }]
`amount`	required	Amount to be transfered in *wei*		*1000000000000000000*
`currency`	required	Unique identifier of the transaction		*ETH*
`details`	required	Sha3 Hash of provided details	"" (empty string)	*0x39ebcda37c1aaa7b6467f16d4f03479e5061031cc61b62342c9216d2ac012a5c*
`nonce`	required	Identifier generated from invoice data		*1270040570*
`encodedUrl`	required	Cross platform redirection url		*https://lqd.money/?data=eyJ1dWlkIjoiNTQxMmEyZjMz...*
`encodedRaw`	required	Encoded invoice data		*eyJ1dWlkIjoiNTQxMmEyZjMz...*

Example

```
POST /invoices HTTP/1.1
Content-Type: application/json

{
    "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
    "amount": 1
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "288e19e69032480784305838b6158055",
    "destinations": [
        {
            "networkId": 1,
            "contractAddress": "0xac8c3D5242b425DE1b86b17E407D8E949D994010",
            "walletAddresses": ["0x627306090abaB3A6e1400e9345bC60c78a8BEf57"],
        }
    ],
    "amount": "1",
```

(continues on next page)

(continued from previous page)


```
    "currency": "ETH",
    "details": "0x290decd9548b62a8d60345a988386fc84ba6bc95484008f6362f93160ef3e563",
    "nonce": 1270040570,
    "encoded": {
        "url": "https://lqd.money/?
↪data=eyJ1dWlkIjoiNTQxMmEyZjMzMzAyNDIwZGExNjU1ZjI5Y2FhMWUxNGMiLCJkZXN0aW5hdGlvbnMiOlt7Im5ldHdvcmtJZC
↪%3D"
        "raw":
↪"eyJ1dWlkIjoiNTQxMmEyZjMzMzAyNDIwZGExNjU1ZjI5Y2FhMWUxNGMiLCJkZXN0aW5hdGlvbnMiOlt7Im5ldHdvcmtJZCI6MS
↪%3D"
    }
}
```


1.4.2 List

List all transfer performed by the daemon during this round. Filters can be applied.

Endpoint

`GET /invoices`

Request

Name	Required	Description	Default Value	Example
`count`	optional	Amount to be transfered in *wei*	100	*50*
`recipient`	optional	Ethereum address to sent the transaction to		*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`sender`	optional	Ethereum address used to perform the transaction	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`amount`	optional	Amount transfered in *wei*		*1000000000000000000*
`nonce`	optional	Identifier generated from invoice data		*1270040570*
`status`	optional	Status of the transaction	confirmed	*confirmed, pending*

Response

Array

Name	Required	Description	Default Value	Example
`recipient`	required	Ethereum address to sent the transaction to		*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`sender`	required	Ethereum address used to perform the transaction	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`amount`	required	Amount transfered in *wei*		*1000000000000000000*
`transactionId`	required	Unique identifier of the transaction		*1*
`status`	required	Status of the transaction		*confirmed, pending*
`nonce`	required	Identifier generated from transaction		*1270040570*
`created_on`	required	Date the transaction has been performed (ISO format)		*1970-01-01T00:00:00.000Z*

Example

```
GET /invoices?nonce=1270040570 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
      "sender": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
      "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
      "amount": "1",
      "created_on": "2018-07-03T12:33:27.409540Z",
      "nonce": "1270040570",
      "txId": "420",
      "status": "confirmed"
  }
]
```

1.5 Wallet

Wallet endpoint provides all informations related to the current state of the daemon.

1.5.1 Information

Retrieve all information about the wallet managed by the daemon.

Endpoint

GET /wallet

Request

Name	Required	Description	Default Value	Example

Response

Name	Required	Description	Default Value	Example
`address`	required	SDK's ethereum address		*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`ethereumNodeUrl`	required	Ethereum node SDK is connected to		*https://mainnet.infura.io/*
`ethereumNetworkId`	required	Network id seen by SDK		*1*
`hubContractAddress`	required	Liquidity Hub contract SDK is connected to		*0xac8c3D5242b425DE1b86b17E407D8E949D994010*
`hubProviderUrl`	required	Liquidity Hub host SDK is connected to		*https://beta.liquidity.network*
`amount`	required	Amount SDK manages off-chain		*1000000000000000000*
`onchain`	required	On-chain information managed by the SDK		*{ amount: 0 }*

Example

```
GET /wallet HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "address": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
    "ethereumNodeUrl": "https://mainnet.infura.io/",
    "ethereumNetworkId": 1,
    "hubContractAddress": "0x46a1657f132030476496112126Be35384bdDDDAD",
    "hubProviderUrl": "https://public.liquidity.network",
    "amount": "1",
    "onchain": {
        "amount": "0"
    }
}
```

1.6 Hub

in your `config.json` file, you have specified a `HUB_PROVIDER_URL`. This url is how the daemon communicates and retrieve information stored on the hub. Hub endpoints are exposed trough this category and allows the user to get low level information.

1.6.1 Wallets

List all wallets registered on associated liquidity hub.

Endpoint

GET /hub/wallets

Request

Name	Required	Description	Default Value	Example

Response

Array

Name	Required	Description	Default Value	Example
address	required	Liquidity registered ethereum address without prepending *0x*		*627306090abaB3A6e1400e9345bC60c78a8BEf57*

Example

```
GET /hub/wallets HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "address": "627306090abaB3A6e1400e9345bC60c78a8BEf57"
    }
]
```

1.6.2 Audit Registration

For a specific `:address`, get information about its registration.

Endpoint

GET /hub/audit/:address/registration

Request

Name	Required	Description	Default Value	Example
`address`	required	Liquidity registered ethereum address	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*

Response

Name	Required	Description	Default Value	Example
`round`	required	Round the wallet was registered		*4*
`wallet_signature`	required	Wallet signature on registration		*691550b1480f09d50789777d176323f9c4c13a0817263f063d35986ce940086d3985175*
`hub_signature`	required	Hub signature on registration		*7795d1f7314bbbbf8a4144a7343ce413d5640099d889093270909e34b21e55f12956b66*

Example

```
GET /hub/audit/0x627306090abaB3A6e1400e9345bC60c78a8BEf57/registration HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "round": 4,
   "wallet_signature":
↪"691550b1480f09d50789777d176323f9c4c13a0817263f063d35986ce940086d398517571e68511025800f94789faf6331
↪",
   "hub_signature":
↪"7795d1f7314bbbbf8a4144a7343ce413d5640099d889093270909e34b21e55f12956b66cf786da22e0c3774ffe72b10781
↪"
}
```

1.6.3 Audit Transfers

For a specific `:address`, get information about its transfers.

Endpoint

`GET /hub/audit/:address/transfers`

Request

Name	Required	Description	Default Value	Example
`address`	required	Liquidity registered ethereum address	SDK's ethereum address	*0x62730609 0abaB3A6e1400e9345bC60 c78a8BEf57*

Response

Array

Name	Required	Description	Default Value	Example
`recipient`	required	Ethereum address to sent the transaction to		*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`sender`	required	Ethereum address used to perform the transaction	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*
`amount`	required	Amount transfered in *wei*		*1000000000000000000*
`transactionId`	required	Unique identifier of the transaction		*1*
`status`	required	Status of the transaction		*confirmed, pending*
`nonce`	required	Identifier generated from transaction		*1270040570*
`round`	required	Round the transaction was performed		*4*

Example

```
GET /hub/audit/0x627306090abaB3A6e1400e9345bC60c78a8BEf57/transfers HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
   {
       "transactionId": "38",
       "round": 4,
       "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
       "sender": "0xd977dA63d086d222EDE0aa68ee84328310485FFE",
       "amount": "50000",
       "nonce": "1270040570"
   }
]
```

1.6.4 Audit Deposits

For a specific `:address`, get information about its deposits.

Endpoint

GET /hub/audit/:address/deposits

Request

Name	Required	Description	Default Value	Example
address	required	Liquidity registered ethereum address	SDK's ethereum address	*0x627306090abaB3A6e1400e9345bC60c78a8BEf57*

Response

Name	Required	Description	Default Value	Example
transactionId	required	Transaction id of the deposit on Ethereum		*ee456c4f5f31e9b44c94df251690469fef4cf1c2b8f603edc62d77*
block	required	Block the transaction has been included into on Ethereum		*5898261*
round	required	Round the wallet was registered		*4*
amount	required	Amount deposited in *wei*		*1000000000000000000*
created_on	required	Date the transaction has been performed (ISO format)		*1970-01-01T00:00:00.000Z*

Example

```
GET /hub/audit/0x627306090abaB3A6e1400e9345bC60c78a8BEf57/deposits HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
   {
       "transactionId":
↪"ee456c4f5f31e9b44c94df251690469fef4cf1c2b8f603edc62d7703acda098c",
       "block": 5898261,
       "round": 4,
       "amount": "50000",
       "time": "2018-07-03T12:33:27.409540Z"
   }
]
```

CHAPTER 2

About

Liquidity Network revolutionizes the way people trade and interact. Through its network of payment hubs, Liquidity Network enables free, instant and secure off-chain transfers and exchange of digital assets and makes people custodian of their own funds.

We believe in the potential of off-chain technology to ensure inclusion and trust minimization. We envision a future where everyone is able to perform money transfers, without the need of a bank account and free of expensive transaction fees or long waiting times.

To bring this vision to life, we strongly encourage collaboration and stimulate developers to work on and implement our technology. The present documentation goes over all methods available for you to start working on it today!

HTTP Routing Table

/hub

/invoices

/transfers

/wallet