Liquidity SDK Documentation

Release latest

Nov 06, 2018

Contents:

CHAPTER 1

Communities

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

1.1 Liquidity SDK

Liquidity SDK facilitates the easy integration of off-chain payments and exchange within your application and is divided into two parts. Firstly, the wallet automaton which is a self-hosted liquidity wallet management tool for your transactions. Secondly, the Liquidity language library that allows you to access wallet automaton directly from your favourite language or from REST endpoints.

1.1.1 Architecture

Liquidity is built around hubs that create a link between the blockchain and off-chain transaction ecosystems. The blockchain remains the ultimate source of trust in the case of conflict while the hub manages the off-chain state of network participants. To leverage the complexity of an active state management, the wallet automaton queries the hub to obtain its last state. It provides the up-to-date state through REST endpoint that the language library is using.

To provide a concrete example, the flow to perform a full transfer of *32 wei* to a specific user is given below. Language Library is considered to be the end user.

1. The user asks the automaton to perform a transfer of 32 wei

2. The automaton creates the transfer
3. The automaton sends the transfer
4. At regular intervals, the automaton checks if the hub has included the transfer
5. When the transfer is included in an update, the automaton recognises it
6. When the user checks if there is a pending transfer, no notice is received because the transfer has been performed

If the process fails at any point, the user is able to contact the automaton and perform the security associated operations.

1.1.2 Wallet automaton

Liquidity network is a non-custodian payment system. The core component being the wallet automaton through which the user can perform operations on the network while remaining in control of their funds, making the end user the custodian. For applications looking to implement off-chain payments, this component handles communication with the liquidity hub, ensuring it behaves correctly.

In terms of technology, the automaton is a docker container that synchronises with the hub and provides various endpoints. All endpoints use the internal state of the automaton. The automaton is hosted by the user and has knowledge of their private key, therefore it is able to sign off-chain transfers 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 automaton. It wraps the provided REST API using language specific features. For now, the language library is only available for Node.JS. If you have built an implementation for your preferred language, you can submit an issue on liquidity SDK repo <https://github.com/liquidity-network/liquidity-sdk>‘__.

In the transfer sequence described above, the user has to call on the automaton regularly in order to ascertain if there are 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 Prerequisite

- 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 automaton properly.

- A Liquidity registered wallet: Official website

  This wallet will be managed by your automaton. Click on `Add New Wallet`, select the wallet you want to 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 automaton 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-sdk
```

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": "0xac8c3D5242b425DE1b86b17E407D8E949D994010",
  "HUB_PROVIDER_URL": "https://beta.liquidity.network"
}
```

3. Start your wallet

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

For those of you who are using a graphic docker version, launch `docker-compose.yml` file located at the root of the liquidity SDK directory.

This command will start your wallet automaton. It can take some time for it to synchronise with the hub, especially when first launched.

4. Celebrate

You have finished the installation! The SDK is self-hosted on your machine and is accessible under `localhost:3600`.

You can test this by accessing https://localhost:3600/wallet/information here you should be able to see the current state of your wallet.

1.2.3 Documentation

The following documentation is structured into categories. Each category contains a list of endpoints made accessible by the wallet automaton. These endpoints are documented with what they provide, how to call them, their expected result and an example.

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

1.3 Transfers

Transfers endpoint manage wallet transfers and are used to send and check the state of your transfers. All transfers made are saved to a database. Consequently, it is possible to retrieve your funds if the hub is unresponsive, satisfying the non-custodian property.

1.3.1 Send

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

Endpoint

```
POST /transfers/send
```

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 /tranfers/send
data: {
  "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
  "amount": 1
}
```

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

1.3.2 List

A List of all transfers performed by the automaton during this round. Filters can be applied.

Endpoint

```
GET /transfers/list
```

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/list?status=pending
```

```
[
    {
        "sender": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
        "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
        "amount": "1",
        "created_on": "1970-01-01T00:00:00.000Z",
```

(continues on next page)

(continued from previous page)

```
        "nonce": "1270040570",
        "txId": 1,
        "status": "pending",
    }
]
```

1.4 Invoice

A tool for merchants to generate a transaction that is then fulfilled by their 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 automaton is used.

Endpoint

```
POST /invoices/generate
```

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*
encoded.url	required	Cross platform redirection url		*https://lqd.money/?data=eyJ1dWlkIjoiNTQxMmEyZjMz...*
encoded.raw	required	Encoded invoice data		*eyJ1dWlkIjoiNTQxMmEyZjMz...*

Example

```
POST /invoices/generate
data: {
  "amount": 1,
  "recipient": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57"
}
```

```
{
    "uuid": "288e19e69032480784305838b6158055",
    "destinations": [
        {
            "networkId": 1,
            "contractAddress": "0xac8c3D5242b425DE1b86b17E407D8E949D994010",
            "walletAddresses": ["0x627306090abaB3A6e1400e9345bC60c78a8BEf57"],
        }
    ],
    "amount": "1",
    "currency": "ETH",
    "details": "0x290decd9548b62a8d60345a988386fc84ba6bc95484008f6362f93160ef3e563",
    "nonce": 1270040570,
    "encoded": {
        "url": "https://lqd.money/?
↪data=eyJ1dWlkIjoiNTQxMmEyZjMzMzAyNDIwZGExNjU1ZjI5Y2FhMWUxNGMiLCJkZXN0aW5hdGlvbnMiOlt7Im5ldHdvcmtJZ
↪%3D"
```

(continues on next page)

(continued from previous page)

```
        "raw":
↪"eyJ1dWlkIjoiNTQxMmEyZjMzMzAyNDIwZGExNjU1ZjI5Y2FhMWUxNGMiLCJkZXN0aW5hdGlvbnMiOlt7Im5ldHdvcmtJZCI6MS
↪%3D"
    }
}
```

1.4.2 List

List all transfers performed by the automaton during this round. Filters can be applied.

Endpoint

```
GET /invoices/list
```

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/list?nonce=1270040570
```


```
{
    "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 information regarding the current state of the automaton.

1.5.1 Information

Retrieve all information related to the wallet managed by the automaton.

Endpoint

```
GET /wallet/information
```

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/information
```


```
{
    "address": "0x627306090abaB3A6e1400e9345bC60c78a8BEf57",
    "ethereumNodeUrl": "https://mainnet.infura.io/",
    "ethereumNetworkId": 1,
    "hubContractAddress": "0xac8c3D5242b425DE1b86b17E407D8E949D994010",
    "hubProviderUrl": "https://beta.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 automaton communicates and retrieves information stored on the hub. Hub endpoints are exposed through this category and allow the user to receive low level information.

1.6.1 Wallets

List all wallets registered on the 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
```

```
[
    {
        "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
```


```
{
    "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
```

```
[
    {
        "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
```

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

CHAPTER 2

About

Liquidity Network revolutionises the way people trade and interact. Through its network of payment hubs, Liquidity Network enables free, instant and secure off-chain transfers and the exchange of digital assets without the oversight of any third parties. Moreover, it gives people complete ownership over their funds, they are the custodian at any point.

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

To bring this vision to life, we strongly encourage collaboration and ask developers to work on and implement our technology. The following guide covers all methods for you to start applying the technology today!