web3-fusion-extend Documentation

Fusion.org

Feb 26, 2019

User Documentation

Fusion is an enterprise financial infrastructure blockchain solution simplifying the full digital life of any asset.

With the ethreum world users are familiar that they have a wallet and a balance. If user own ERC20 or ERC721 tokens wallets applications hide the technicalities of calling contracts to get their values.

Fusion evolutionary approach moves the asset and its balance to the users ethereum wallet.

No contracts are needed as multiple balances each with a unique asset Id can be contained on the chain in the wallet.

Other inflationary practices to code such as creating escrows and atomic swaps are also eliminated by simply having the asset in the users account.

Fusion allows all assets to have time periods, these time periods enable safe lending of an asset.

So a balance of 1 Fusion Token can be Timelocked for 30 days, sent to another account.

At the end of the 30 days that other account no longer can access that value in their wallet.

All done natively on the chain without contracts.

The goal of Fusion is to:

- **make assets (tokens) a natural part of send and receive of value**
  - eliminate need for ERC20 contracts
  - eliminate need for ERC721 contracts
  - simply the familiar paridigm of from, to, value -> from, to, assetId, value
- **introduce the concept of timelock of an asset as a natural wallet feature**
  - split the use of any asset into any sliver of time
  - allow that time portion of that asset to be securely lent to someone
- **quantum swap - introduce a one transaction method to exchange assets between parties**
  - the ability to exchange assets between accounts in one transaction on the blockchain without chance of error
  - assets that are timelocked can be exchanged
  - swaps can be made public to all users or a targeted user.

The purpose of the fusionapi libary is to take the popular web3.js package and open it up to the extended functionallity of that the Fusion blockchain server provides.

Contents:

Keyword Index, Search Page

CHAPTER 1

Welcome to fusionapi's documentation!

Fusion is an enterprise financial infrastructure blockchain solution simplifying the full digital life of any asset.

With the ethreum world users are familiar that they have a wallet and a balance. If user own ERC20 or ERC721 tokens wallets applications hide the technicalities of calling contracts to get their values.

Fusion evolutionary approach moves the asset and its balance to the users ethereum wallet.

No contracts are needed as multiple balances each with a unique asset Id can be contained on the chain in the wallet.

Other inflationary practices to code such as creating escrows and atomic swaps are also eliminated by simply having the asset in the users account.

Fusion allows all assets to have time periods, these time periods enable safe lending of an asset.

So a balance of 1 Fusion Token can be Timelocked for 30 days, sent to another account.

At the end of the 30 days that other account no longer can access that value in their wallet.

All done natively on the chain without contracts.

The goal of Fusion is to:

- **make assets (tokens) a natural part of send and receive of value**
  - eliminate need for ERC20 contracts
  - eliminate need for ERC721 contracts
  - simply the familiar paridigm of from, to, value -> from, to, assetId, value
- **introduce the concept of timelock of an asset as a natural wallet feature**
  - split the use of any asset into any sliver of time
  - allow that time portion of that asset to be securely lent to someone
- **quantum swap - introduce a one transaction method to exchange assets between parties**
  - the ability to exchange assets between accounts in one transaction on the blockchain without chance of error
  - assets that are timelocked can be exchanged

  - swaps can be made public to all users or a targeted user.

The purpose of the fusionapi libary is to take the popular web3.js package and open it up to the extended functionallity of that the Fusion blockchain server provides.

Note: This documentation is work in progress and web3-fusion-extend in currently in beta. Please feel free to submit pull request or issues via the edit on github link in the upper right.

CHAPTER 2

Contributing

If you're reading this, you're awesome! Thank you for helping us make this project great and being a part of the web3-fusion-extend community. Here are a few guidelines that will help you along the way.

2.1 Submitting a pull request

web3-fusion-extend is a community project, so pull requests are always welcome, but, before working on a large change, it is best to open an issue first to discuss it with the maintainers.

When in doubt, keep your pull requests small. To give a PR the best chance of getting accepted, don't bundle more than one feature or bug fix per pull request. It's always best to create two smaller PRs than one big one.

As with issues, please make sure your title clear explains the issue.

When adding new features or modifying existing, please attempt to include tests to confirm the new behaviour. You can read more about our test setup here.

2.1.1 Branch Structure

All stable releases are tagged (view tags). At any given time, `master` represents the latest development version of the library. Patches or hotfix releases are prepared on an independent branch.

`master is unsafe`

We will do our best to keep `master` in good shape, with tests passing at all times. However, in order to move fast, we will make API changes that your application might not be compatible with.

2.1.2 How to increase the chance of being accepted?

We will only accept a pull request for which all tests pass. Make sure the following is true:

- The branch is not behind master.

- **If a feature is being added:**
  - If the result was already achievable with the core library, explain why this feature needs to be added to the core.
  - It includes relevant tests.
  - If this is a common use case, considered adding an example to the documentation.
- If a bug is being fixed, test cases that fail without the fix are included.
- The code is formatted (run `yarn prettier`).
- The code is linted (run `yarn lint`).
- If API documentation is being changed in the source, `yarn docs:api` was run.
- If prop types were changed, the TypeScript declarations were updated.
- If TypeScript declarations were changed, `yarn typescript` passed.
- The PR title follows the pattern `[Component] Imperative commit message`. (See: (How to Write a Git Commit Message) for a great explanation)

2.2 Getting started

Please create a new branch from an up to date master on your fork. (Note, urgent hotfixes should be branched off the latest stable release rather than master)

1. Fork the web3-fusion-extend repository on Github
2. Clone your fork to your local machine `git clone git@github.com:<yourname>/web3-fusion-extend.git`
3. Create a branch `git checkout -b my-topic-branch`
4. Make your changes, lint, then push to to GitHub with `git push --set-upstream origin my-topic-branch`.
5. Visit GitHub and make your pull request.

If you have an existing local repository, please update it before you start, to minimise the chance of merge conflicts.

```
git remote add upstream git@github.com:FusionFoundation/web3-fusion-extend.git
git checkout master
git pull upstream master
git checkout -b my-topic-branch
yarn
```

2.2.1 Coding style

Please follow the coding style of the project. web3-fusion-extend uses eslint, so if possible, enable linting in your editor to get real-time feedback. The linting rules can be run manually with the following command `yarn lint`.

You can also run `yarn prettier` to reformat the code.

Finally, when you submit a pull request, they are run again by Circle CI, but hopefully by then your code is already clean!

2.3 License

By contributing your code to the FusionFoundation/web3-fusion-extend GitHub repository, you agree to license your contribution under the MIT license.

Note: This documentation is work in progress and web3-fusion-extend in currently in beta. Please feel free to submit pull request or issues via the edit on github link in the upper right.

CHAPTER 3

Getting Started

web3-fusion-extend is a collection of libraries which allow you to interact with a local or remote fusion node, using a HTTP or IPC connection.

Fusion offers a radical approach to representing value within a block chain environment.

A public address contain multiple assets and balances for these assets.

An assetId is the id returned when an asset is created and actions can be performed on it.

The asset creator also has the ability to increase and decrease supply.

This enables cross chain and cross functional systems to be built that enable the interchange of assets.

Assets can also be TimeLocked. When an asset is time locked its ownership is leant for the period specified. At the end of time lock period the rights of the asset are returned to the original owner.

With the representation of assets, the need to exchange assets securely and simply becomes paramount.

The Fusion protocol introduces quantumSwaps which are composed of three functions: makeSwap - tell others what you will exchange for your asset recallSwap - cancel the request for an exchange takeSwap - exchange your asset for the other parties asset listed in the make swap

This package extends the Ethereum compatible JavaScript API which implements the Generic JSON RPC spec to support the Fusion protocol.

It's available on npm as a node module. You need to run a local Ethereum node to use this library.

- *Documentation*

3.1 Installation

Node.js

```
npm install web3-fusion-extend
```

Yarn

```
yarn add web3-fusion-extend
```

3.2 Usage

Create a web3 object as your normally would and then call web3FusionExtend with that object. web3 will then have two additional interfaces (fsn and fsntx)

```
var web3FusionExtend = require('web3-fusion-extend')
web3 = new Web3(provider);
web3 = web3FusionExtend.extend(web3)
console.log(web3.fsn.consts.FSNToken);
```

```
console.log(web3); // {fsn: .., fsntx: ...} // It's here!
```

There you go, now you can use it:

```
var balance = web3.eth.getBalance(coinbase);
web3.fsn
    .getAllBalances( web3.eth.coinbase ) // fsn supports multiple assets and balances␣
↪on an address
    .then( balances => {
      console.log( balances )
      assert(  balances[web3.fsn.consts.FSNToken] , "there should be a balance for␣
↪fusion tokens always"  )
      done();
    })
    .catch(err => {
      done(err);
    });
```

You can find more examples in the test directory.

There is also a full block explorer api written as an example.

- *BlockExplorerApi*

3.3 Contribute!

We'd greatly appreciate any *contribution* you make.

3.4 Requirements

Node.js npm

```
# On Linux:
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm
sudo apt-get install nodejs-legacy
```

3.5 Testing (mocha)

When testing a connect string to a local fusion node and a wallet address is needed as environment variables

```
CONNECT_STRING="ws://3.16.110.25:9001" WALLET_ADDRESS=
↪"0x4A5a7Aa4130e407d3708dE56db9000F059200C62" npm test
```

3.6 Community

- Gitter
- Forum

3.7 Documentation

Documentation

- *Overview*
- *fsn*
- *fsntx*
- *Constants*

3.8 License

CHAPTER 4

autoPurchaseTickets

4.1 Ticket purchase applicaton

An application to keep tickets at a constant level built with the web3-fusion-extend package

- autoPurchaseTicket.js

4.1.1 Example

```
node autoPurchaseTicket --c "wss://example.com" -p "./password.txt" -k "./keystore.key
↪" -n 10 -d 12345
```

```
-c --connectString web socket gateway to connect to
-k  --keyStore keystore file to use
-p  --passPharseFile key file
-g  -- gasPrice gas price 1 - 100 (defaults to 2 gwei)
-n  --Number of tickets to purchase
-d  --Chain Id
```

CHAPTER 5

blockexplorerapi

5.1 Overview

This block explorer api section shows how to collect all information from the fusion block chain and update a mysql database.

You will need a node for the application readAllBlocksToDatabase.js to communicate with as well as a mysql database to store the information.

You will need an environment string called DB_CONNECT_STRING

It can be passed on the command line.

```
DB_CONNECT_STRING="{'host':'mysqlserver,'user':'adminuser','password':'password',
↪'database':'fusionblockdb','connectionLimit':100}" node readAllBlocksToDatabase.js
```

You can then host the api server for the database via:

```
DB_CONNECT_STRING="{'host':'mysqlserver,'user':'adminuser','password':'password',
↪'database':'fusionblockdb','connectionLimit':100}" nodemon   node ./bin/www
```

5.2 Fusion Org and its public explorer api

Fusion organization keeps an api endpoint open at https://api.fusionnetwork.io to assist in application development

You can try commands like https://api.fusionnetwork.io/blocks/latest or https://api.fusionnetwork.io/transactions/latest

Note if you are not a developer the results may look scary but they are the actual last block or transaction info

5.3 Installation

```
npm install
```

5.4 API Commands

If running locally replace api.fusionnetwork.io with your own server link

5.4.1 Assets

- http://api.fusionnetwork.io/assets/0xbbd28ab973a7be78af3d8a3c3f1097c87fc020b2bd9270aa292518e8a93c32ae
- http://api.fusionnetwork.io/assets/all?page=0&size=2&sort=desc

5.4.2 Balance

- http://api.fusionnetwork.io/balances/0xC4A9441afB052cB454240136CCe71Fb09316EA94
- http://api.fusionnetwork.io/balances/all?page=0&size=2&sort=asc

5.4.3 Blocks

- http://api.fusionnetwork.io/blocks/latest
- http://api.fusionnetwork.io/blocks/300
- http://api.fusionnetwork.io/blocks/all?sort=asc&page=2&size=10&field=height&sort=desc
- http://api.fusionnetwork.io/blocks/range?to=10&from=100

fields can be: [timestamp, hash , numberOfTransactions, height]

5.4.4 Swaps

- http://api.fusionnetwork.io/swaps/0xbbd28ab973a7be78af3d8a3c3f1097c87fc020b2bd9270aa292518e8a93c32ae
- http://api.fusionnetwork.io/swaps/all?page=0&size=2&sort=asc

5.4.5 Transactions

- http://api.fusionnetwork.io/transactions/latest
- http://api.fusionnetwork.io/transactions/0x346aab726aa05808698ec9aba5da4e4c4574863e87951b5107d3fdabc290bbaa
- http://api.fusionnetwork.io/transactions/all?sort=asc&page=2&size=10&field=height

fields can be: [timestamp, hash , type, block , asset]

Return an array of transactions from a - seperated array - http://api.fusionnetwork.io/transactions/ts?ts=address1-address2

5.4.6 Fusion Price

- **last price**
  - http://api.fusionnetwork.io/fsnprice
- **historical prices**
  - http://api.fusionnetwork.io/fsnprice/?page=0&size=2&sort=asc
- **last two prices**
  - http://api.fusionnetwork.io/fsnprice/?page=0&size=2&sort=desc

5.4.7 Search

- http://api.fusionnetwork.io/search/{[}block,blockhash,transaction,address{]}
- miner leaderboard http://api.fusionnetwork.io/leaderboard
- **Ticket purchase applicaton**
  - autoPurchaseTicket.js

Example: node autoPurchaseTicket –c"wss://example.com" -p "./password.txt" -k "./keystore.key" -n 10

```
-c --connectString web socket gateway to connect to
-k  --keyStore keystore file to use
-p  --passPharseFile key file
-n  --Number of tickets to purchase
```

CHAPTER 6

calculateMiningReward

6.1 Calculate Mining Reward Example

An example on how to calculate mining rewards can be found in:

- calculateMiningReward.js on github.

6.1.1 Example

Calculate Mining Reward Example

Overview

The following application demonstrates how to calculate mining rewards against the fusion chain.

You will need an environment string called CONNECT_STRING to connect to a gatweway to read blocks

It can be passed on the command line.

Remember to run npm install, to install runtime dependecies

```
npm install
```

You can then run the application via:

```
CONNECT_STRING="wss://gateway.fusionnetwork.io:10001" node caculateMiningReward.js
```

CHAPTER 7

myFusionWallet

A full functionion wallet with asset creation, time locks, and quantum swaps.

You can see it live in action at: https://www.myfusionwallet.com.

And the source can be found on github at: https://github.com/FUSIONFoundation/myfusionwallet.

CHAPTER 8

PSNBlockExplorer

A rich example of using the block explore api.

You can see it live in action at: https://blocks.fusionnetwork.io/.

And the source can be found on github at: https://github.com/FUSIONFoundation/PSNBlockExplorer.

CHAPTER 9

PurchaseTicket

A graphical example of using the web3 api to check balances and purchase tickets.

You can see it live in action at: https://tickets.fusionnetwork.io/.

And the source can be found on github at: https://github.com/FUSIONFoundation/PurchaseTicket.

CHAPTER 10

createAsset

An example on how to create an asset can be found here.

CHAPTER 11

getAllBalances

An example on how to request balances can be found in:
getAllBalances.js on github.

CHAPTER 12

Constants

web3.fsn.consts = { }

12.1 FSNToken

The asset ID representing the Fusion Token

```
FSNToken: "0xffffffffffffffffffffffffffffffffffffffff"
```

12.2 TicketLogAddress

```
TicketLogAddress: "0xfffffffffffffffffffffffffffffffffffffffe",
```

12.3 TicketLogAddress_Topic_To_Function

```
TicketLogAddress_Topic_To_Function: {
  0: "ticketSelected",
  1: "ticketReturn",
  2: "ticketExpired"
},
```

12.4 TicketLogAddress_Topic_ticketSelected

```
TicketLogAddress_Topic_ticketSelected:
  "0x0000000000000000000000000000000000000000000000000000000000000000",
```

12.5 TicketLogAddress_Topic_ticketReturn

```
TicketLogAddress_Topic_ticketReturn:
  "0x0000000000000000000000000000000000000000000000000000000000000001",
```

12.6 TicketLogAddress_Topic_ticketExpired

```
TicketLogAddress_Topic_ticketExpired:
  "0x0000000000000000000000000000000000000000000000000000000000000002",
```

12.7 FSNCallAddress

```
FSNCallAddress: "0xffffffffffffffffffffffffffffffffffffffff",
```

12.8 FSNCallAddress_Topic_To_Function

```
FSNCallAddress_Topic_To_Function: {
  // GenNotationFunc wacom
  0: "GenNotationFunc", // = iota
  // GenAssetFunc wacom
  1: "GenAssetFunc",
  // SendAssetFunc wacom
  2: "SendAssetFunc",
  // TimeLockFunc wacom
  3: "TimeLockFunc",
  // BuyTicketFunc wacom
  4: "BuyTicketFunc",
  // AssetValueChangeFunc wacom
  5: "AssetValueChangeFunc",
  // MakeSwapFunc wacom
  6: "MakeSwapFunc",
  // RecallSwapFunc wacom
  7: "RecallSwapFunc",
  // TakeSwapFunc wacom
  8: "TakeSwapFunc"
        // MakeSwapFuncExt wacom
    9: "MakeSwapFuncExtOld",
    // MakeSwapFuncExt wacom
    10: "MakeSwapFuncExt",
    // TakeSwapFuncExt wacom
```

(continues on next page)

(continued from previous page)

```
    11: "TakeSwapFuncExt",
    // AssetValueChangeFunc wacom
    12: "AssetValueChangeExtFunc"
},
```

12.9 FSNCallAddress_Topic_GenNotationFunc

```
FSNCallAddress_Topic_GenNotationFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000000",
```

12.10 FSNCallAddress_Topic_GenAssetFunc

```
FSNCallAddress_Topic_GenAssetFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000001",
```

12.11 FSNCallAddress_Topic_SendAssetFunc

```
FSNCallAddress_Topic_SendAssetFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000002",
```

12.12 FSNCallAddress_Topic_TimeLockFunc

```
FSNCallAddress_Topic_TimeLockFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000003",
```

12.13 FSNCallAddress_Topic_BuyTicketFunc

```
FSNCallAddress_Topic_BuyTicketFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000004",
```

12.14 FSNCallAddress_Topic_AssetValueChangeFunc

```
FSNCallAddress_Topic_AssetValueChangeFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000005",
```

12.15 FSNCallAddress_Topic_MakeSwapFunc

```
FSNCallAddress_Topic_MakeSwapFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000006",
```

12.16 FSNCallAddress_Topic_RecallSwapFunc

```
FSNCallAddress_Topic_RecallSwapFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000007",
```

12.17 FSNCallAddress_Topic_TakeSwapFunc

```
FSNCallAddress_Topic_TakeSwapFunc:
  "0x0000000000000000000000000000000000000000000000000000000000000008"
```

12.18 FSNCallAddress_Topic_MakeSwapFuncExtOld

```
FSNCallAddress_Topic_MakeSwapFuncExtOld:
  "0x0000000000000000000000000000000000000000000000000000000000000009",
```

12.19 FSNCallAddress_Topic_MakeSwapFuncExt

```
FSNCallAddress_Topic_MakeSwapFuncExt:
  "0x000000000000000000000000000000000000000000000000000000000000000a",
```

12.20 FSNCallAddress_Topic_TakeSwapFuncExt

```
FSNCallAddress_Topic_TakeSwapFuncExt:
  "0x000000000000000000000000000000000000000000000000000000000000000b",
```

12.21 FSNCallAddress_Topic_AssetValueChangeExtFunc

```
FSNCallAddress_Topic_AssetValueChangeExtFunc:
  "0x000000000000000000000000000000000000000000000000000000000000000c"
```

Note: This documentation is work in progress and web3-fusion-extend in currently in beta. Please feel free to submit pull request or issues via the edit on github link in the upper right.

CHAPTER 13

fsn

FSN documentation

- *Overview*
- *fsn*
- *fsntx*
- *Constants*

```
fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",start:"0x100",end:"0x200",
↪value:"0x100"},"123456")
```

```
fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x4A5a7Aa4130e407d3708dE56db9000F059200C62",start:"0x100",end:"0x200",
↪value:"0x100"},"123456")
```

common params CP from send from or call member gas gas limit gasPrice gas price nonce send from or call member account nonce

common send params CSP CP asset the asset ID to receiver value amount

13.1 allAssets

```
fsn.allAssets()
...
```

allAssets get the all assets list no params

13.1.1 Parameters

none

13.1.2 Returns

`Object`: With the following methods:

- `ID` - `Hash`: Description
- `Owner` - `Address`: Description
- `Name` - `String`: Description
- `Symbol` - `String`: Description
- `Decimals` - `Number`: Description
- `Total` - `Number ` `json:",string"``: Description
- `CanChange` - `bool`: Description

13.1.3 Example

```
fsn.allAssets()
```

```
await web3.fsn.allAssets().then(function (res) {
    assetList = res;
});

for (let asset in assetList) {
    let id = assetList[asset]["ID"];
    let owner = assetList[asset]["Owner"];
    let owned = false;
    let assetBalance = '';

    try {
        await web3.fsn.getBalance(id, walletAddress).then(function (res) {
            assetBalance = res;
        });
    } catch (err) {
        console.log(err);
    }
```

13.2 allNotation

```
fsn.allNotation()
...
```

allNotation get the all notation no params

13.2.1 Parameters

none

13.2.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.2.3 Example

```
fsn.allNotation()
```

13.3 allSwaps

```
fsn.allSwaps()
...
```

allSwaps get the all quantum swap list no params

13.3.1 Parameters

none

13.3.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.3.3 Example

```
try {
    await web3.fsn.allSwaps().then(function (res) {
        swapList = res;
    });
} catch (err) {
    console.log(err);
}

for (let asset in swapList) {
    let id = swapList[asset]["ID"];
    let owner = swapList[asset]["Owner"];
    let owned = false;
    let assetBalance = '';

    try {
        await web3.fsn.getBalance(id, walletAddress).then(function (res) {
            assetBalance = res;
        });
    } catch (err) {
        console.log(err);
    }
```

(continues on next page)

(continued from previous page)

```
    let fromAsset = [];
    let toAsset = [];

    try {
        await web3.fsn.getAsset(swapList[asset]["FromAssetID"]).then(function (res) {
            fromAsset = res;
        });
    } catch (err) {

    }

    try {
        await web3.fsn.getAsset(swapList[asset]["ToAssetID"]).then(function (res) {
            toAsset = res;
        });
    } catch (err) {

    }

    owner === walletAddress ? owned = true : owned = false;

    let fromAmount = (swapList[asset].MinFromAmount / $scope.countDecimals(fromAsset.
↪Decimals));

    let toAmount = swapList[asset].MinToAmount / $scope.countDecimals(toAsset.
↪Decimals);
    let swapRate = fromAmount / toAmount;
    let time = new Date(parseInt(swapList[asset]["Time"]) * 1000);

    let tMonth = time.getUTCMonth();
    let tDay = time.getUTCDate();
    let tYear = time.getUTCFullYear();

    let hours = time.getUTCHours();
    let minutes = time.getUTCMinutes();

    if (time.getUTCMinutes() < 10) {
        minutes = "0" + time.getUTCMinutes();
    }
    // Global

    time = $scope.months[tMonth] + ' ' + tDay + ', ' + tYear;
    let timeHours = hours + ':' + minutes;

    // Maker parts
    let minimumswap = fromAmount / parseInt(swapList[asset]["SwapSize"]);

    // Taker specific parts
    let swapratetaker = toAmount / fromAmount;
    let minimumswaptaker = fromAmount * swapratetaker;

    // Targes section

    let targes = '';

    swapList[asset]["Targes"].length > 0 ? targes = 'Private' : targes = 'Public';
```

13.4 allTickets

```
fsn.allTickets()
...
```

allTickets get the all notation no params

13.4.1 Parameters

none

13.4.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.4.3 Example

```
fsn.allTickets()
```

13.5 allTicketsByAddress

```
fsn.allTicketsByAddress(eth.coinbase)
...
```

allTicketsByAddress get all tickets by address address the user's address

13.5.1 Parameters

1. `address` - `String|Number`: Description

13.5.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.5.3 Example

```
fsn.allTicketsByAddress(eth.coinbase)
```

```
this._web3.fsn.allTicketsByAddress(walletAddress)
    .then(res => {
        return {
        allBalances,
        allTickets: res,
        timelockUsableBalance
        };
    });
```

13.6 totalNumberOfTickets

```
fsn.totalNumberOfTickets()
...
```

totalNumberOfTickets return number of active tickets no params

13.6.1 Parameters

none

13.6.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.6.3 Example

```
fsn.totalNumberOfTickets()
```

```
this._web3.fsn
    .totalNumberOfTickets()
    .then(totalTickets => {
        return Object.assign(loadsOfInfo, {
        totalTickets,
        latestBlock: block
        });
    });
```

13.7 totalNumberOfTicketsByAddress

```
fsn.totalNumberOfTicketsByAddress(eth.coinbase)
...
```

totalNumberOfTicketsByAddress return number of tickets an address controls address address that bought tickets

13.7.1 Parameters

```
```

1. `address` - `String|Number`: Description

13.7.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.7.3 Example

```
fsn.totalNumberOfTicketsByAddress(eth.coinbase)
```

13.8 assetToTimeLock

```
fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x1",end:"0x2A300",
↪value:"0x1400000000000000"},"123456")
...
```

assetToTimeLock send the asset to time lock CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

13.8.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `asset` - `String|Number`: Description
   - `to` - `String|Number`: Description
   - `value` - `Number|String|BN|BigNumber`: Description
   - `start` - `Number`: Description
   - `end` - `Number`: Description
2. `passwd` - `String`: Description

13.8.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.8.3 Example

```
fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x1",end:"0x2A300",
↪value:"0x1400000000000000"},"123456")

fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x2a900",end:
↪"0x3A300",value:"0x1400000000000000"},"123456")

fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start: getHexDate('2018-12-
↪01') ,end: getHexDate('2019-01-01'),value:"0x1340000000000000"},"123456")
```

13.9 getHexDate

```
fsn.getHexDate('2018-12-01')
...
```

getHexDate helper function to convert date to posix time in hex

13.9.1 Parameters

none

13.9.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.9.3 Example

```
fsn.getHexDate('2018-12-01')
fsn.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:fsn.coinbase,start:"0x1",end:"0x100000",value:"0x100"},"123456")
```

13.10 timeLockToAsset

```
fsn.timeLockToAsset({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:fsn.coinbase,start:"0x0",end:"0x0",value:"0x100"},"123456")
...
```

timeLockToAsset send the time lock to asset CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

13.10.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `asset` - `String|Number`: Description
   - `to` - `String|Number`: Description
   - `value` - `Number|String|BN|BigNumber`: Description
   - `start` - `Number`: Description
   - `end` - `Number`: Description
2. `passwd` - `String`: Description

13.10.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.10.3 Example

```
fsn.timeLockToAsset({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:fsn.coinbase,start:"0x0",end:"0x0",value:"0x100"},"123456")
```

13.11 timeLockToTimeLock

```
fsn.timeLockToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",start:"0x101",end:"0x200",
↪value:"0x100"},"123456")
...
```

timeLockToTimeLock send the time lock CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

13.11.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

2. `passwd` - `String`: Description

13.11.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.11.3 Example

```
fsn.timeLockToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",start:"0x101",end:"0x200",
↪value:"0x100"},"123456")
```

13.12 buyTicket

```
fsn.buyTicket({from:fsn.coinbase},"123456")
...
```

buyTicket buy the ticket CP see the top and the "from" ignore from who buy the ticket password the account password

13.12.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `start` - `Number`: Description
- `end` - `Number`: Description

2. `passwd` - `String`: Description

13.12.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.12.3 Example

```
fsn.buyTicket({from:fsn.coinbase},"123456")
```

13.13 genAsset

```
fsn.genAsset({from:fsn.coinbase,name:"FusionTest",symbol:"FST",decimals:1,total:"0x200
↪"},"123456")
fsn.genAsset({from:"0x91db50f5c36ae7616009d4e94462dca4d4c7e833",name:"JONESY",symbol:
↪"JSY",decimals:1,total:"0x2000000000"},"123123123")
...
```

genAsset generate a asset CP see the top and the "from" ignore from who gen the asset and the owner of the asset name the name of asset symbol the symbol of asset decimals the asset decimal digit total the total number of the asset and the owner will be get same number asset CanChange whether asset can be incremented or decremented by the owner [optional] password the account password

13.13.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.

- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `name` - `String`: Description
- `symbol` - `String`: Description
- `decimals` - `Number`: Description
- `total` - `Number|String|BN|BigNumber`: Description
- `canChange` - `bool`: Description

2. `passwd` - `String`: Description

13.13.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.13.3 Example

```
fsn.genAsset({from:fsn.coinbase,name:"FusionTest",symbol:"FST",decimals:1,total:"0x200
↪"},"123456")
fsn.genAsset({from:"0x91db50f5c36ae7616009d4e94462dca4d4c7e833",name:"JONESY",symbol:
↪"JSY",decimals:1,total:"0x2000000000"},"123123123")
```

```
web3.fsn
  .genAsset(
    {
      from: process.env.WALLET_ADDRESS,
      name: assetName,
      symbol: assetShortName,
      decimals: 18,
      total: "0x0",
      CanChange: true
    },
    process.env.PASSPHRASE
  )
  .then(transactionReceipt => {
    return waitForTransactionToComplete(transactionReceipt)
      .then(transactionReceipt => {
        if (transactionReceipt.status != "0x1") {
          done(new Error("transaction error"));
          return;
        }
        let data = JSON.parse(
          web3.fsn.hex2a(transactionReceipt.logs[0].data)
        );
        // console.log("json data => ", data);
        assetId = data.AssetID;
        assert(assetId, "there should be an asset id");
        done();
      })
      .catch(err => {
        console.log(
```

(continues on next page)

(continued from previous page)

```
          "gen asset (waitForTransactionToComplete) created the following error",
          err
        );
        done(err);
      });
  })
  .catch(err => {
    console.log("gen asset created the following error", err);
    done(err);
  });
```

13.14 genNotation

```
fsn.genNotation({from:fsn.coinbase},"123456")
...
```

genNotation gen a notation for a account CP see the top and the "from" ignore from who gen the notation password the account password

13.14.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.

2. `passwd` - `String`: Description

13.14.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.14.3 Example

```
fsn.genNotation({from:fsn.coinbase},"123456")
```

13.15 sendAsset

```
fsn.sendAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")

fsn.sendAsset({from:fsn.coinbase,to:"0x91db50F5c36aE7616009d4e94462DcA4D4c7e833",
↪value:"0x2",asset:
↪"0x88d18f81620e5684e880dddcf0b6c167a9154d4c499bc9fad47b98634110eeec"},"123456")
...
```

sendAsset send asset to other account CSP see the top

13.15.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `asset` - `String|Number`: Description
   - `to` - `String|Number`: Description
   - `value` - `Number|String|BN|BigNumber`: Description
2. `passwd` - `String`: Description

13.15.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.15.3 Example

```
fsn.sendAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")

fsn.sendAsset({from:fsn.coinbase,to:"0x91db50F5c36aE7616009d4e94462DcA4D4c7e833",
↪value:"0x2",asset:
↪"0x88d18f81620e5684e880dddcf0b6c167a9154d4c499bc9fad47b98634110eeec"},"123456")
```

13.16 decAsset

```
fsn.decAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",value:
↪"0x1",asset:"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},
↪"123456")
...
```

decAsset decrease account asset balance CSP see the top and the "from","to" ignore from the asset owner to the dec account password the account password

13.16.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `isInc` - `Boolean`: Description
- `transacData` - `String`: Description

2. `passwd` - `String`: Description

13.16.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.16.3 Example

```
fsn.decAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",value:
↪"0x1",asset:"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},
↪"123456")
```

```
web3.fsn
   .decAsset(
     {
       from: process.env.WALLET_ADDRESS,
       to: process.env.WALLET_ADDRESS,
       value: "0000000000000000001",
       asset: assetId
     },
     process.env.PASSPHRASE
   )
```

(continues on next page)

(continued from previous page)

```
  .then(transactionReceipt => {
    return waitForTransactionToComplete(transactionReceipt)
      .then(transactionReceipt => {
        if (transactionReceipt.status !== true) {
          done(new Error("transaction error " + transactionReceipt));
          return;
        }
        let data = JSON.parse(
          web3.fsn.hex2a(transactionReceipt.logs[0].data)
        );
        // console.log("json data => ", data);
        done();
      })
      .catch(err => {
        console.log(
          "dec asset (waitForTransactionToComplete) created the following error",
          err
        );
        done(err);
      });
  })
  .catch(err => {
    console.log("dec asset created the following error", err);
  });
```

13.17 incAsset

```
fsn.incAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",value:
↪"0x1",asset:"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},
↪"123456")
...
```

incAsset increase account asset balance CSP see the top and the "from","to" ignore from the asset owner to the inc account password the account password

13.17.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `isInc` - `Boolean`: Description

- `transacData` - `String`: Description

2. `passwd` - `String`: Description

13.17.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.17.3 Example

```
fsn.incAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",value:
↪"0x1",asset:"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},
↪"123456")
```

```
web3.fsn
  .incAsset(
    {
      from: process.env.WALLET_ADDRESS,
      to: process.env.WALLET_ADDRESS,
      value: "1000000000000000000",
      asset: assetId
    },
    process.env.PASSPHRASE
  )
  .then(transactionReceipt => {
    return waitForTransactionToComplete(transactionReceipt)
      .then(transactionReceipt => {
        if (transactionReceipt.status !== true) {
          done(new Error("transaction error " + transactionReceipt));
          return;
        }
        let data = JSON.parse(
          web3.fsn.hex2a(transactionReceipt.logs[0].data)
        );
        // console.log("json data => ", data);
        done();
      })
      .catch(err => {
        console.log(
          "inc asset (waitForTransactionToComplete) created the following error",
          err
        );
        done(err);
      });
  })
  .catch(err => {
    console.log("inc asset created the following error", err);
  });
```

13.18 getAddressByNotation

```
fsn.getAddressByNotation(104)
...
```

getAddressByNotation get the notation of the address notation account notation

13.18.1 Parameters

1. `notation` - `Number`: Description
2. `blockNr` - `Number`: Description

13.18.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.18.3 Example

```
fsn.getAddressByNotation(104)
```

```
try {
    await web3.fsn.getAddressByNotation(parseInt(address)).then(function (res) {
        console.log(res);
    });
    $scope.$eval(function () {
        $scope.walletAddressError = false;
        $scope.validWalletAddress = true;
        $scope.checkingUSAN = false;
    });
    return;
} catch (err) {
    $scope.$eval(function () {
        $scope.walletAddressError = true;
        $scope.validWalletAddress = false;
        $scope.checkingUSAN = false;
    });
    return;
}
```

13.19 getAllBalances

```
fsn.getAllBalances("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
...
```

getAllBalances get all assets balances address the user's address blockNumber default now block number

13.19.1 Parameters

1. `address` - `String|Number`: Description
2. `blockNr` - `Number`: Description

13.19.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.19.3 Example

```
fsn.getAllBalances("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
```

```
await web3.fsn.getAllBalances(walletAddress).then(function (res) {
    for (let contractaddress in res) {
        $scope.myAssets.push(contractaddress);
    }
})
```

13.20 getAllTimeLockBalances

```
fsn.getAllTimeLockBalances("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
...
```

getAllTimeLockBalances get all time lock balances address the user's address blockNumber default now block number

13.20.1 Parameters

1. `address` - `String|Number`: Description
2. `blockNr` - `Number`: Description

13.20.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.20.3 Example

```
fsn.getAllTimeLockBalances("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
```

```
try {
  let timeLockBalances = await web3.fsn.getAllTimeLockBalances(address);
  all = JSON.stringify({
    timeLockBalances
  });
```

13.21 getAsset

```
fsn.getAsset("0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
...
```

getAsset get the asset info assetID the asset ID

13.21.1 Parameters

1. `assetID` - `String|Number`: Description
2. `blockNr` - `Number`: Description

13.21.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.21.3 Example

```
fsn.getAsset("0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
```

```
try {
    await web3.fsn.getAsset(data["FromAssetID"]).then(function (res) {
        fromAsset = res;
    });
} catch (err) {
    console.log(err);
}
```

13.22 getBalance

```
fsn.getBalance("0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
...
```

getBalance like name assetID the asset ID address the user's address blockNumber default now block number

13.22.1 Parameters

1. `assetID` - `String|Number`: Description
2. `address` - `String|Number`: Description
3. `blockNr` - `Number`: Description

13.22.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.22.3 Example

```
fsn.getBalance("0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
```

```
try {
    await web3.fsn.getBalance(id, walletAddress).then(function (res) {
        assetBalance = res;
    });
} catch (err) {
    console.log(err);
}
```

13.23 getNotation

```
fsn.getNotation("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
...
```

getNotation get the notation of address address the account address

13.23.1 Parameters

1. `address` - `String|Number`: Description
2. `blockNr` - `Number`: Description

13.23.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.23.3 Example

```
fsn.getNotation("0x9c48c796cb0bed51a14291bc8cc56dab5aed7b5c")
```

```
try {
  let notation = await web3.fsn.getNotation(address);
  all = JSON.stringify({
    notation
  });
```

13.24 getTimeLockBalance

```
fsn.getTimeLockBalance(
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"0x9c48c796cb0bed51a14291bc8cc56ed7b5c")
...
```

getTimeLockBalance like name assetID the asset ID address the user's address blockNumber default now block number

13.24.1 Parameters

1. `assetID` - `String|Number`: Description
2. `address` - `String|Number`: Description
3. `blockNr` - `Number`: Description

13.24.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.24.3 Example

```
fsn.getTimeLockBalance(
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"0x9c48c796cb0bed51a14291bc8cc56ed7b5c")
```

13.25 makeSwap

```
fsn.makeSwap({from:fsn.coinbase,FromAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
            ToAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffff0000000000000000000000",MinToAmount:1,
↪MinFromAmount:2,SwapSize:2,Targes:[]},"123456")
...
```

makeSwap create a quantum swap CP see the top FromAssetID sell asset id MinFromAmount the min amount of the sell asset ToAssetID buy asset id MinToAmount the min amount of the buy asset SwapSize the max sell package size Targes the address list of the "who can buy" can be null password the owner password

13.25.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `FromAssetID` - `String|Number`: Description
- `FromStartTime` - `Number`: Description
- `FromEndTime` - `Number`: Description
- `MinFromAmount` - `Number|String|BN|BigNumber`: Description
- `ToAssetID` - `String|Number`: Description
- `ToStartTime` - `Number`: Description
- `ToEndTime` - `Number`: Description
- `MinToAmount` - `Number|String|BN|BigNumber`: Description
- `SwapSize` - `Number`: Description
- `Targes` - `Array String|Number`: Description
- `Time` - `Number`: Description

2. `passwd` - `String`: Description

13.25.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.25.3 Example

```
fsn.makeSwap({from:fsn.coinbase,FromAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
          ToAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffff0000000000000000000000",MinToAmount:1,
↪MinFromAmount:2,SwapSize:2,Targes:[]},"123456")
```

13.26 recallSwap

```
fsn.recallSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"},"123456")
...
```

recallSwap destroy a quantum swap and get the asset back CP see the top SwapID the swap ID password the owner password

13.26.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID` - `String|Number`: Description

2. `passwd` - `String`: Description

13.26.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.26.3 Example

```
fsn.recallSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"},"123456")
```

13.27 takeSwap

```
fsn.takeSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",Size:"0x1"},
↪"123456")
...
```

takeSwap buy a quantum swap CP see the top SwapID the swap ID Size the package size password the owner password

13.27.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account

- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID` - `String|Number`: Description
- `Size` - `Number`: Description

2. `passwd` - `String`: Description

13.27.2 Returns

`Object`: With the following methods:

- `Object`: Description

13.27.3 Example

```
fsn.takeSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",Size:"0x1"},
↪"123456")
```

```
fsn.getSnapshot( blocknumber )
fsn.getSnapshotHash(blockHas)
```

```
web3.fsn.getHexDate = function( d ) {
    return "0x" + (( new Date(d)).getTime() / 1000).toString(16)
};

web3.fsn.hex2a = function( hexData ) {
    hexData = hexData.replace( "0x", "" )
    let hex = hexData.toString();//force conversion
    let str = '';
    for (let i = 0; (i < hex.length && hex.substr(i, 2) !== '00'); i += 2)
        str += String.fromCharCode(parseInt(hex.substr(i, 2), 16));
    return str;
}
```

Note: This documentation is work in progress and web3-fusion-extend in currently in beta. Please feel free to submit pull request or issues via the edit on github link in the upper right.

CHAPTER 14

fsntx

FSNTX commands

- *Overview*
- *fsn*
- *Constants*

```
personal.unlockAccount( eth.coinbase, "123456" )
var tx = fsntx.buildBuyTicketTx( {from:eth.coinbase} )
tx.from = eth.coinbase
var tx2 = eth.signTransaction( tx )
fsntx.sendRawTransaction(tx2.tx)
```

14.1 sendRawTransaction

```
fsntx.sendRawTransaction()
...
```

sendRawTransaction

14.1.1 Parameters

1. `tx` - `*types.Transaction`: Description

14.1.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.1.3 Example

```
fsntx.sendRawTransaction()
```

```
$scope.recallSwap = async function (swap_id) {
    if (walletService.wallet !== null) {
        let password = walletService.password;
        let accountData = uiFuncs.getTxData($scope);
        let walletAddress = accountData.from;

        let data = {
            from: walletAddress,
            SwapID: swap_id
        };

        if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
            $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
        }

        try {
            await web3.fsntx.buildRecallSwapTx(data).then(function (tx) {
                tx.from = walletAddress;
                tx.chainId = _CHAINID;
                data = tx;
                if ($scope.wallet.hwType == "ledger") {
                    return;
                }
                return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                    console.log(txHash);
                    $scope.recallSwapSuccess.open()
                })
            })
        } catch (err) {
            $scope.errorModal.open();
            console.log(err);
        }
        if ($scope.wallet.hwType == "ledger") {
            let ledgerConfig = {
                privKey: $scope.wallet.privKey ? $scope.wallet.getPrivateKeyString()␣
↪: "",
                path: $scope.wallet.getPath(),
                hwType: $scope.wallet.getHWType(),
                hwTransport: $scope.wallet.getHWTransport()
            }
            let rawTx = data;
            var eTx = new ethUtil.Tx(rawTx);
            if (ledgerConfig.hwType == "ledger") {
                var app = new ledgerEth(ledgerConfig.hwTransport);
                var EIP155Supported = true;
                var localCallback = async function (result, error) {
                    if (typeof error != "undefined") {
                        if (callback !== undefined) callback({
                            isError: true,
                            error: error
                        });
```

(continues on next page)

(continued from previous page)

```
                        return;
                    }
                    var splitVersion = result['version'].split('.');
                    if (parseInt(splitVersion[0]) > 1) {
                        EIP155Supported = true;
                    } else if (parseInt(splitVersion[1]) > 0) {
                        EIP155Supported = true;
                    } else if (parseInt(splitVersion[2]) > 2) {
                        EIP155Supported = true;
                    }
                    var oldTx = Object.assign(rawTx, {});
                    let input = oldTx.input;
                    return uiFuncs.signed(app, rawTx, ledgerConfig, true, function
↪(res) {
                        oldTx.r = res.r;
                        oldTx.s = res.s;
                        oldTx.v = res.v;
                        oldTx.input = input;
                        oldTx.chainId = "0x1";
                        delete oldTx.isError;
                        delete oldTx.rawTx;
                        delete oldTx.signedTx;
                        web3.fsntx.sendRawTransaction(oldTx).then(function (txHash) {
                            $scope.recallSwapSuccess.open()
                        })
                    })
                }
                $scope.notifier.info('Please, confirm transaction on Ledger.');
                await app.getAppConfiguration(localCallback);
            }
        }
    }
}
```

14.2 buildGenNotationTx

```
fsntx.buildGenNotationTx()
...
```

buildGenNotationTx

14.2.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.

14.2.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.2.3 Example

```
await web3.fsntx.buildGenNotationTx({
    from: walletAddress
}).then((tx) => {
    tx.chainId = _CHAINID;
    data = tx;
    tx.from = walletAddress;
    if ($scope.wallet.hwType == "ledger"){
        return;
    }
    return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).then(txHash =>
↪ {
        $scope.requestedSAN = true;
        $scope.$apply(function () {
            $scope.addressNotation.value = 'USAN Requested';
            $scope.addressNotation.value = 'USAN Requested';
        });
    })
});
```

14.3 genNotation

```
fsntx.genNotation({from:fsn.coinbase},"123456")
...
```

genNotation gen a notation for a account CP see the top and the "from" ignore from who gen the notation password the account password

14.3.1 Parameters

1. `Object`: The transaction object

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.

2. `passwd` - `string`: Description

14.3.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.3.3 Example

```
fsntx.genNotation({from:fsn.coinbase},"123456")
```

14.4 buildGenAssetTx

```
fsntx.buildGenAssetTx()
...
```

buildGenAssetTx

14.4.1 Parameters

1. `Object`: The transaction object

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `name` - `string`: Description
- `symbol` - `string`: Description
- `decimals` - `Number`: Description
- `total` - `Number|String|BN|BigNumber`: Description
- `canChange` - `Boolean`: Description

14.4.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.4.3 Example

```
try {
    await web3.fsntx.buildGenAssetTx(data).then(tx => {
        tx.chainId = _CHAINID;
        data = tx;
        if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType == "trezor") {
            return;
        } else {
            return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                $scope.$eval(function () {
                    $scope.assetCreate.errorMessage = '';
                    $scope.assetCreate.assetHash = txHash;
                });
                $scope.createAssetFinal.open();
            });
        }
    });
} catch (err) {
    $scope.errorModal.open();
}
```

14.5 genAsset

```
fsntx.genAsset({from:fsn.coinbase,name:"FusionTest",symbol:"FST",decimals:1,total:
↪"0x200"},"123456")
fsntx.genAsset({from:"0x91db50f5c36ae7616009d4e94462dca4d4c7e833",name:"JONESY",
↪symbol:"JSY",decimals:1,total:"0x2000000000"},"123123123")
...
```

genAsset generate a asset CP see the top and the "from" ignore from who gen the asset and the owner of the asset name the name of asset symbol the symbol of asset decimals the asset decimal digit total the total number of the asset and the owner will be get same number asset CanChange whether asset can be incremented or decremented by the owner [optional] password the account password

14.5.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `name` - `string`: Description
- `symbol` - `string`: Description
- `decimals` - `Number`: Description
- `total` - `Number|String|BN|BigNumber`: Description
- `canChange` - `bool`: Description

2. `passwd` - `string`: Description

14.5.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.5.3 Example

```
fsntx.genAsset({from:fsn.coinbase,name:"FusionTest",symbol:"FST",decimals:1,total:
↪"0x200"},"123456")
fsntx.genAsset({from:"0x91db50f5c36ae7616009d4e94462dca4d4c7e833",name:"JONESY",
↪symbol:"JSY",decimals:1,total:"0x2000000000"},"123123123")
```

14.6 buildSendAssetTx

```
fsntx.buildSendAssetTx()
...
```

buildSendAssetTx

14.6.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description

14.6.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.6.3 Example

```
fsntx.buildSendAssetTx()
```

```
$scope.sendAsset = async function () {
    $scope.successMessagebool = true;
    let accountData = uiFuncs.getTxData($scope);
    let from = accountData.from;
    let to = $scope.sendAsset.toAddress;
    let decimals = '';
    let asset = $scope.assetToSend;
    let hash = '';
    let data = {};

    if (to.length < 42) {
        await web3.fsn.getAddressByNotation(parseInt(to)).then(function (address) {
            to = address;
        });
    }

    await web3.fsn.getAsset(asset).then(function (res) {
        decimals = parseInt(res["Decimals"]);
    });

    let amount = $scope.sendAsset.amountToSend.toString();

    amount = $scope.makeBigNumber(amount, decimals);

    if ($scope.transactionType == "none") {

        if (!$scope.account && ($scope.wallet.hwType !== "ledger") && ($scope.wallet.
↪hwType !== "trezor")) {
            $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
        }

        try {
            await web3.fsntx.buildSendAssetTx({
                from: from,
                to: to,
                value: amount.toString(),
                asset: asset
            }).then((tx) => {
                console.log(tx);
                tx.from = from;
                tx.chainId = _CHAINID;
                data = tx;
                if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType ==
↪"trezor") {
                    return;
                }
                return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                    hash = txHash;
                    $scope.sendAssetFinal.open();
                    $scope.$eval(function () {
                        $scope.successHash = hash;
```

(continues on next page)

(continued from previous page)

```
                    $scope.successHash = hash;
                });
            })
        });
    } catch (err) {
        console.log(err);
        $scope.errorModal.open();
    }
```

14.7 sendAsset

```
fsntx.sendAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")

fsntx.sendAsset({from:fsn.coinbase,to:"0x91db50F5c36aE7616009d4e94462DcA4D4c7e833",
↪value:"0x2",asset:
↪"0x88d18f81620e5684e880dddcf0b6c167a9154d4c499bc9fad47b98634110eeec"},"123456")
...
```

sendAsset send asset to other account CSP see the top

14.7.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `asset` - `String|Number`: Description
   - `to` - `String|Number`: Description
   - `value` - `Number|String|BN|BigNumber`: Description
2. `passwd` - `string`: Description

14.7.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.7.3 Example

```
fsntx.sendAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")

fsntx.sendAsset({from:fsn.coinbase,to:"0x91db50F5c36aE7616009d4e94462DcA4D4c7e833",
↪value:"0x2",asset:
↪"0x88d18f81620e5684e880dddcf0b6c167a9154d4c499bc9fad47b98634110eeec"},"123456")
```

14.8 buildAssetToTimeLockTx

```
fsntx.buildAssetToTimeLockTx()
...
```

buildAssetToTimeLockTx

14.8.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

14.8.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.8.3 Example

```
fsntx.buildAssetToTimeLockTx()
```

```
$scope.sendAsset = async function () {
    $scope.successMessagebool = true;
    let accountData = uiFuncs.getTxData($scope);
    let from = accountData.from;
    let to = $scope.sendAsset.toAddress;
    let decimals = '';
    let asset = $scope.assetToSend;
    let hash = '';
    let data = {};

    if (to.length < 42) {
        await web3.fsn.getAddressByNotation(parseInt(to)).then(function (address) {
            to = address;
        });
    }

    await web3.fsn.getAsset(asset).then(function (res) {
        decimals = parseInt(res["Decimals"]);
    });

    let amount = $scope.sendAsset.amountToSend.toString();

    amount = $scope.makeBigNumber(amount, decimals);

    if ($scope.transactionType == "none") {

        if (!$scope.account && ($scope.wallet.hwType !== "ledger") && ($scope.wallet.
↪hwType !== "trezor")) {
            $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
        }

        try {
            await web3.fsntx.buildSendAssetTx({
                from: from,
                to: to,
                value: amount.toString(),
                asset: asset
            }).then((tx) => {
                console.log(tx);
                tx.from = from;
                tx.chainId = _CHAINID;
                data = tx;
                if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType ==
↪"trezor") {
                    return;
                }
                return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                    hash = txHash;
                    $scope.sendAssetFinal.open();
                    $scope.$eval(function () {
                        $scope.successHash = hash;
                        $scope.successHash = hash;
                    });
                })
            });
```

(continues on next page)

(continued from previous page)

```
        } catch (err) {
            console.log(err);
            $scope.errorModal.open();
        }

        $scope.$apply(function () {
            $scope.successHash = hash;
        });
    }
    if ($scope.transactionType == "daterange") {

        if ($scope.sendAsset.fromTime == '') {
            $scope.sendAsset.fromTime = new Date();
        }

        let fromTime = getHexDate(convertDate($scope.sendAsset.fromTime));
        let tillTime = getHexDate(convertDate($scope.sendAsset.tillTime));
        if (!$scope.account && ($scope.wallet.hwType !== "ledger") && ($scope.wallet.
↪hwType !== "trezor")) {
            $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
        }

        try {
            await web3.fsntx.buildAssetToTimeLockTx({
                asset: asset,
                from: from,
                to: to,
                start: fromTime,
                end: tillTime,
                value: amount
            }).then((tx) => {
                tx.from = from;
                tx.chainId = _CHAINID;
                data = tx;
                if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType ==
↪"trezor") {
                    return;
                }
                return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                    $scope.sendAssetFinal.open();
                    $scope.$eval(function () {
                        $scope.successHash = txHash;
                        $scope.successHash = txHash;
                    });
                })
            });
        } catch (err) {
            $scope.errorModal.open();
        }
    }
```

14.9 assetToTimeLock

```
fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x1",end:"0x2A300",
↪value:"0x1400000000000000"},"123456")

fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x2a900",end:
↪"0x3A300",value:"0x1400000000000000"},"123456")

fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start: getHexDate('2018-12-
↪01') ,end: getHexDate('2019-01-01'),value:"0x1340000000000000"},"123456")
...
```

assetToTimeLock send the asset to time lock CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

14.9.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

2. `passwd` - `string`: Description

14.9.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.9.3 Example

```
fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x1",end:"0x2A300",
↪value:"0x1400000000000000"},"123456")

fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start:"0x2a900",end:
↪"0x3A300",value:"0x1400000000000000"},"123456")

fsntx.assetToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0xa7455DF112c953F3c73c2C25559965e1A8a20024",start: getHexDate('2018-12-
↪01') ,end: getHexDate('2019-01-01'),value:"0x1340000000000000"},"123456")
```

14.10 buildTimeLockToTimeLockTx

```
fsntx.buildTimeLockToTimeLockTx()
...
```

buildTimeLockToTimeLockTx

14.10.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

14.10.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.10.3 Example

```
fsntx.buildTimeLockToTimeLockTx()
```

```
if (!$scope.account && ($scope.wallet.hwType !== "ledger") && ($scope.wallet.hwType !
↪== "trezor")) {
    $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString($scope.
↪wallet.getPrivateKey()));
}

try {
    await web3.fsntx.buildTimeLockToTimeLockTx({
        asset: asset,
        from: from,
        to: to,
        start: fromTime,
        end: tillTime,
        value: amount
    }).then((tx) => {
        tx.from = from;
        tx.chainId = _CHAINID;
        data = tx;
        if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType == "trezor") {
            return;
        }
        return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
            $scope.$eval(function () {
                $scope.sendAssetFinal.open();
                $scope.successHash = txHash;
                $scope.successHash = txHash;
            });
        })
    });
} catch (err) {
    $scope.errorModal.open();
}
```

14.11 timeLockToTimeLock

```
fsntx.timeLockToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",start:"0x101",end:"0x200",
↪value:"0x100"},"123456")
...
```

timeLockToTimeLock send the time lock CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

14.11.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account

- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

2. `passwd` - `string`: Description

14.11.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.11.3 Example

```
fsntx.timeLockToTimeLock({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",start:"0x101",end:"0x200",
↪value:"0x100"},"123456")
```

14.12 buildTimeLockToAssetTx

```
fsntx.buildTimeLockToAssetTx()
...
```

buildTimeLockToAssetTx

14.12.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - ‘‘String|Number‘: Description

- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

14.12.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.12.3 Example

```
fsntx.buildTimeLockToAssetTx()
```

```
$scope.sendBackToAssetsFunction = async function (id) {
    let accountData = uiFuncs.getTxData($scope);
    id = $scope.timeLockToAssetId;
    let tlData = $scope.timeLockList[id];

    let from = accountData.from;

    if (!$scope.account && ($scope.wallet.hwType !== "ledger") && ($scope.wallet.
↪hwType !== "trezor")) {
        $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
    }

    let startTime = web3.utils.numberToHex(tlData.posixStartTime);
    let endTime = web3.utils.numberToHex(tlData.posixEndTime);

    // JavaScript / Go incompatibility -1 error
    if (tlData.posixEndTime === 18446744073709552000) {
        endTime = web3.fsn.consts.TimeForeverStr;
    }

    let data = {};

    try {
        await web3.fsntx.buildTimeLockToAssetTx({
            asset: tlData.asset,
            from: from,
            to: from,
            start: startTime,
            end: endTime,
            value: tlData.rawValue
        }).then((tx) => {
            tx.from = from;
            tx.chainId = _CHAINID;
            data = tx;
            if ($scope.wallet.hwType == "ledger" || $scope.wallet.hwType == "trezor")
↪{
                return;
            }
            return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
```

(continues on next page)

(continued from previous page)

```
                $scope.successModal.open();
            })
        });
    } catch (err) {
        $scope.errorModal.open();
    }
```

14.13 timeLockToAsset

```
fsntx.timeLockToAsset({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:fsn.coinbase,start:"0x0",end:"0x0",value:"0x100"},"123456")
...
```

timeLockToAsset send the time lock to asset CSP see the top startTime the start time of the time lock endTime the end time of the time lock password the account password

14.13.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `start` - `Number`: Description
- `end` - `Number`: Description

2. `passwd` - `string`: Description

14.13.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.13.3 Example

```
fsntx.timeLockToAsset({asset:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",from:fsn.
↪coinbase,to:fsn.coinbase,start:"0x0",end:"0x0",value:"0x100"},"123456")
```

14.14 buildBuyTicketTx

```
fsntx.buildBuyTicketTx()
...
```

buildBuyTicketTx

14.14.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `start` - `Number`: Description
- `end` - `Number`: Description

14.14.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.14.3 Example

```
this._web3.fsntx
  .buildBuyTicketTx({ from: this._walletAddress, end: dayHex })
  .then(tx => {
    console.log(tx);
    // tx.gasLimit =  this._web3.utils.toWei( 21000, "gwei" )
    if (!this._web3 || !this.provider || !this.provider.__connected) {
      // reconnecting need to wait
      cb(new Error("reconnecting"), "reconnecting");
      return;
    }
    return this._web3.fsn.signAndTransmit(
      tx,
      currentDataState.data.signInfo.signTransaction
    );
  })
  .then(txHash => {
    console.log("buy ticket tx -> ", txHash);
    if (!data.activeTicketPurchase) {
      cb(null, "asked to leave");
      return true;
    }
    data.lastStatus = "Pending Tx:" + utils.midHashDisplay(txHash);
```

(continues on next page)

(continued from previous page)

```
    data.lastCall = "purchaseSubmitTicket";
    this.emit("purchaseSubmitTicket", data);
    this.checkConnection();
    return this.waitForTransactionToComplete(
      txHash,
      data,
      new Date().getTime() + 120000
    )
      .then(r => {
        if (r.status) {
          cb(null, "Ticket bought");
        } else {
          cb(new Error("failed to buy"), "Failed to buy ticket will retry");
        }
      })
      .catch(err => {
        cb(err, "Error waiting for ticket to complete");
      });
  })
  .catch(err => {
    console.log(err)
    cb(err, "unknown err");
  });
```

14.15 buyTicket

```
fsntx.buyTicket({from:fsn.coinbase},"123456")
...
```

buyTicket buy the ticket CP see the top and the "from" ignore from who buy the ticket password the account password

14.15.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `start` - `Number`: Description
   - `end` - `Number`: Description
2. `passwd` - `string`: Description

14.15.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.15.3 Example

```
fsntx.buyTicket({from:fsn.coinbase},"123456")
```

14.16 buildIncAssetTx

```
fsntx.buildIncAssetTx()
...
```

buildIncAssetTx

14.16.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `isInc` - `Boolean`: Description
- `transacData` - `string`: Description

14.16.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.16.3 Example

```
fsntx.buildIncAssetTx()
```

```
return web3.fsntx
  .buildIncAssetTx({
    from: key.address,
    to: key.address,
    value: "1000000000000000000",
```

(continues on next page)

(continued from previous page)

```
    asset: assetId
  })
  .then(tx => {
    tx.gasPrice = web3.utils.toWei( new web3.utils.BN( "3" ), "gwei");
    return web3.fsn.signAndTransmit(tx, signInfo.signTransaction).then(tx => {
      totalSent += 1
      console.log( totalSent + "    " +  index * subToDo , tx);
      transactionList.push(tx);
      incAsset(index + 1, numberToDo, done)
    });
  })
  .catch(err => {
    console.log("inc asset created the following error", err);
    done(err);
  });
```

14.17 incAsset

```
fsntx.incAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")
...
```

incAsset increase account asset balance CSP see the top and the “from”,”to” ignore from the asset owner to the inc account password the account password

14.17.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `isInc` - `Boolean`: Description
- `transacData` - `string`: Description

2. `passwd` - `string`: Description

14.17.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.17.3 Example

```
fsntx.incAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")
```

14.18 buildDecAssetTx

```
fsntx.buildDecAssetTx()
...
```

buildDecAssetTx

14.18.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `asset` - `String|Number`: Description
- `to` - `String|Number`: Description
- `value` - `Number|String|BN|BigNumber`: Description
- `isInc` - `Boolean`: Description
- `transacData` - `string`: Description

14.18.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.18.3 Example

```
fsntx.buildDecAssetTx()
```

14.19 decAsset

```
fsntx.decAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")
...
```

decAsset decrease account asset balance CSP see the top and the “from”,”to” ignore from the asset owner to the dec account password the account password

14.19.1 Parameters

1. `Object`: Description
   - `from` - `String|Number` : The address for the sending account
   - `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
   - `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
   - `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
   - `asset` - `String|Number`: Description
   - `to` - `String|Number`: Description
   - `value` - `Number|String|BN|BigNumber`: Description
   - `isInc` - `Boolean`: Description
   - `transacData` - `string`: Description
2. `passwd` - `string`: Description

14.19.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.19.3 Example

```
fsntx.decAsset({from:fsn.coinbase,to:"0x2b1a3eca81ba03a9a4c95f4a04679c90838d7165",
↪value:"0x1",asset:
↪"0x514a46f34e6eb0a98abb3595c4aec33ca8ddf69f135c8fed89e78d0808047965"},"123456")
```

14.20 buildMakeSwapTx

```
fsntx.buildMakeSwapTx()
...
```

buildMakeSwapTx

14.20.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `FromAssetID` - `String|Number`: Description
- `FromStartTime` - `Number`: Description
- `FromEndTime` - `Number`: Description
- `MinFromAmount` - `Number|String|BN|BigNumber`: Description
- `ToAssetID` - `String|Number`: Description
- `ToStartTime` - `Number`: Description
- `ToEndTime` - `Number`: Description
- `MinToAmount` - `Number|String|BN|BigNumber`: Description
- `SwapSize` - `Number`: Description
- `Targes` - `Array String|Number`: Description
- `Time` - `Number`: Description

14.20.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.20.3 Example

```
fsntx.buildMakeSwapTx()
```

```
$scope.makeSwap = async function () {
    targesArray = [];
    let password = walletService.password;
    let accountData = uiFuncs.getTxData($scope);
    let walletAddress = accountData.from;

    let fromAsset = [];
    let toAsset = [];


    try {
        await web3.fsn.getAsset($scope.assetToSend).then(function (res) {
            fromAsset = res;
        });
```

(continues on next page)

(continued from previous page)

```
    } catch (err) {
        $scope.errorModal.open();
        console.log(err);
    }


    try {
        await web3.fsn.getAsset($scope.assetToReceive).then(function (res) {
            toAsset = res;
        });
    } catch (err) {
        $scope.errorModal.open();
        console.log(err);
    }

    if ($scope.makeTarges !== '') {
        let targesArr = $scope.makeTarges.split(',');
        await $scope.processAllTarges(targesArr, 0);

        console.log(targesArray);
    } else {
        targesArray = [];
    }

    if ($scope.makeMinumumSwap == "" || $scope.makeMinumumSwap <= 0) {
        $scope.makeMinumumSwap = 1;
    }

    //Global
    let makeMinimumSwapBN = new BigNumber($scope.makeMinumumSwap);

    //Receive Part
    BigNumber.config({ DECIMAL_PLACES: parseInt(toAsset["Decimals"]-1) });
    let makeReceiveAmountBN = new BigNumber($scope.makeReceiveAmount);
    let makeReceiveAmountDiv = makeReceiveAmountBN.div(makeMinimumSwapBN);
    let makeReceiveString = makeReceiveAmountDiv.toString();
    let makeReceiveFinal = $scope.makeBigNumber(makeReceiveString , parseInt(toAsset[
↪"Decimals"]));

    //Send Part
    BigNumber.config({ DECIMAL_PLACES: parseInt(fromAsset["Decimals"]-1) });
    let makeSendAmountBN = new BigNumber($scope.makeSendAmount);
    let makeSendAmountDiv = makeSendAmountBN.div(makeMinimumSwapBN);
    let makeSendString = makeSendAmountDiv.toString();
    let makeSendFinal = $scope.makeBigNumber(makeSendString , parseInt(fromAsset[
↪"Decimals"]));

    //Convert to Hex

    let minToAmountHex = "0x" + makeReceiveFinal.toString(16);
    let minFromAmountHex = "0x" + makeSendFinal.toString(16);


    let data = {
        from: walletAddress,
        FromAssetID: $scope.assetToSend,
        ToAssetID: $scope.assetToReceive,
```

(continues on next page)

(continued from previous page)

```
        MinToAmount: minToAmountHex,
        MinFromAmount: minFromAmountHex,
        SwapSize: parseInt($scope.makeMinumumSwap),
        Targes: targesArray
    };

    // Send part
    if ($scope.showTimeLockSend == true) {
        if ($scope.sendTimeLock == 'scheduled') {
            let fromStartTime = getHexDate(convertDate($scope.fromStartTime));
            let fromEndTime = web3.fsn.consts.TimeForeverStr;

            data.FromStartTime = fromStartTime;
            data.FromEndTime = fromEndTime;
        }
        if ($scope.sendTimeLock == 'daterange') {
            let fromStartTime = getHexDate(convertDate($scope.todayDate));
            let fromEndTime = getHexDate(convertDate($scope.fromEndTime));

            data.FromStartTime = fromStartTime;
            data.FromEndTime = fromEndTime;
        }
    }

    // Receive part
    if ($scope.showTimeLockReceive == true) {
        if ($scope.receiveTimeLock == 'scheduled') {
            let toStartTime = getHexDate(convertDate($scope.ToStartTime));
            let toEndTime = web3.fsn.consts.TimeForeverStr;

            data.ToStartTime = toStartTime;
            data.ToEndTime = toEndTime;
        }

        if ($scope.receiveTimeLock == 'daterange') {
            let toStartTime = getHexDate(convertDate($scope.todayDate));
            let toEndTime = getHexDate(convertDate($scope.ToEndTime));

            data.ToStartTime = toStartTime;
            data.ToEndTime = toEndTime;
        }
    }

    if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
        $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
    }

    console.log(data);

    try {
        await web3.fsntx.buildMakeSwapTx(data).then(function (tx) {
            console.log(tx);
            tx.from = walletAddress;
            tx.chainId = _CHAINID;
            data = tx;
            if ($scope.wallet.hwType == "ledger") {
```

(continues on next page)

(continued from previous page)

```
                return;
            }
            return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                console.log(txHash);
                $scope.makeSwapConfirmation('end');
            })
        })
    } catch (err) {
        $scope.errorModal.open();
        console.log(err);
    }
```

14.21 makeSwap

```
fsntx.makeSwap({from:fsn.coinbase,FromAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
            ToAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffff0000000000000000000000",MinToAmount:1,
↪MinFromAmount:2,SwapSize:2,Targes:[]},"123456")
...
```

makeSwap create a quantum swap CP see the top FromAssetID sell asset id MinFromAmount the min amount of the sell asset ToAssetID buy asset id MinToAmount the min amount of the buy asset SwapSize the max sell package size Targes the address list of the "who can buy" can be null password the owner password

14.21.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `FromAssetID` - `String|Number`: Description
- `FromStartTime` - `Number`: Description
- `FromEndTime` - `Number`: Description
- `MinFromAmount` - `Number|String|BN|BigNumber`: Description
- `ToAssetID` - `String|Number`: Description
- `ToStartTime` - `Number`: Description
- `ToEndTime` - `Number`: Description
- `MinToAmount` - `Number|String|BN|BigNumber`: Description
- `SwapSize` - `Number`: Description

- `Targes - Array String|Number`: Description
- `Time - Number`: Description

2. `passwd - string`: Description

14.21.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.21.3 Example

```
fsntx.makeSwap({from:fsn.coinbase,FromAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
          ToAssetID:
↪"0xffffffffffffffffffffffffffffffffffffffff0000000000000000000000",MinToAmount:1,
↪MinFromAmount:2,SwapSize:2,Targes:[]},"123456")
```

14.22 buildRecallSwapTx

```
fsntx.buildRecallSwapTx()
...
```

buildRecallSwapTx

14.22.1 Parameters

1. `Object`: Description

- `from - String|Number` : The address for the sending account
- `gas - Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice - Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce - Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID - String|Number`: Description

14.22.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.22.3 Example

```
fsntx.buildRecallSwapTx()
```

```
$scope.recallSwap = async function (swap_id) {
    if (walletService.wallet !== null) {
        let password = walletService.password;
        let accountData = uiFuncs.getTxData($scope);
        let walletAddress = accountData.from;

        let data = {
            from: walletAddress,
            SwapID: swap_id
        };

        if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
            $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
        }

        try {
            await web3.fsntx.buildRecallSwapTx(data).then(function (tx) {
                tx.from = walletAddress;
                tx.chainId = _CHAINID;
                data = tx;
                if ($scope.wallet.hwType == "ledger") {
                    return;
                }
                return web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).
↪then(txHash => {
                    console.log(txHash);
                    $scope.recallSwapSuccess.open()
                })
            })
        } catch (err) {
            $scope.errorModal.open();
            console.log(err);
        }
        if ($scope.wallet.hwType == "ledger") {
            let ledgerConfig = {
                privKey: $scope.wallet.privKey ? $scope.wallet.getPrivateKeyString()␣
↪: "",
                path: $scope.wallet.getPath(),
                hwType: $scope.wallet.getHWType(),
                hwTransport: $scope.wallet.getHWTransport()
            }
            let rawTx = data;
            var eTx = new ethUtil.Tx(rawTx);
            if (ledgerConfig.hwType == "ledger") {
                var app = new ledgerEth(ledgerConfig.hwTransport);
                var EIP155Supported = true;
                var localCallback = async function (result, error) {
                    if (typeof error != "undefined") {
                        if (callback !== undefined) callback({
                            isError: true,
                            error: error
                        });
```

(continues on next page)

(continued from previous page)

```
                        return;
                    }
                    var splitVersion = result['version'].split('.');
                    if (parseInt(splitVersion[0]) > 1) {
                        EIP155Supported = true;
                    } else if (parseInt(splitVersion[1]) > 0) {
                        EIP155Supported = true;
                    } else if (parseInt(splitVersion[2]) > 2) {
                        EIP155Supported = true;
                    }
                    var oldTx = Object.assign(rawTx, {});
                    let input = oldTx.input;
                    return uiFuncs.signed(app, rawTx, ledgerConfig, true, function
↪(res) {
                        oldTx.r = res.r;
                        oldTx.s = res.s;
                        oldTx.v = res.v;
                        oldTx.input = input;
                        oldTx.chainId = "0x1";
                        delete oldTx.isError;
                        delete oldTx.rawTx;
                        delete oldTx.signedTx;
                        web3.fsntx.sendRawTransaction(oldTx).then(function (txHash) {
                            $scope.recallSwapSuccess.open()
                        })
                    })
                }
                $scope.notifier.info('Please, confirm transaction on Ledger.');
                await app.getAppConfiguration(localCallback);
            }
        }
    }
}
```

14.23 recallSwap

```
fsntx.recallSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"},"123456")
...
```

recallSwap destroy a quantum swap and get the asset back CP see the top SwapID the swap ID password the owner password

14.23.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.

- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID` - `String|Number`: Description

2. `passwd` - `string`: Description

14.23.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.23.3 Example

```
   fsntx.recallSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"},"123456")

.. code-block:: javascript

  $scope.recallSwap = async function (swap_id) {
       if (walletService.wallet !== null) {
           let password = walletService.password;
           let accountData = uiFuncs.getTxData($scope);
           let walletAddress = accountData.from;

           let data = {
               from: walletAddress,
               SwapID: swap_id
           };

           if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
               $scope.account = web3.eth.accounts.privateKeyToAccount($scope.
↪toHexString($scope.wallet.getPrivateKey()));
           }

           try {
               await web3.fsntx.buildRecallSwapTx(data).then(function (tx) {
                   tx.from = walletAddress;
                   tx.chainId = _CHAINID;
                   data = tx;
                   if ($scope.wallet.hwType == "ledger") {
                       return;
                   }
                   return web3.fsn.signAndTransmit(tx, $scope.account.
↪signTransaction).then(txHash => {
                       console.log(txHash);
                       $scope.recallSwapSuccess.open()
                   })
               })
           } catch (err) {
               $scope.errorModal.open();
               console.log(err);
           }
           if ($scope.wallet.hwType == "ledger") {
               let ledgerConfig = {
```

(continues on next page)

(continued from previous page)

```
                    privKey: $scope.wallet.privKey ? $scope.wallet.
↪getPrivateKeyString() : "",
                    path: $scope.wallet.getPath(),
                    hwType: $scope.wallet.getHWType(),
                    hwTransport: $scope.wallet.getHWTransport()
                }
                let rawTx = data;
                var eTx = new ethUtil.Tx(rawTx);
                if (ledgerConfig.hwType == "ledger") {
                    var app = new ledgerEth(ledgerConfig.hwTransport);
                    var EIP155Supported = true;
                    var localCallback = async function (result, error) {
                        if (typeof error != "undefined") {
                            if (callback !== undefined) callback({
                                isError: true,
                                error: error
                            });
                            return;
                        }
                        var splitVersion = result['version'].split('.');
                        if (parseInt(splitVersion[0]) > 1) {
                            EIP155Supported = true;
                        } else if (parseInt(splitVersion[1]) > 0) {
                            EIP155Supported = true;
                        } else if (parseInt(splitVersion[2]) > 2) {
                            EIP155Supported = true;
                        }
                        var oldTx = Object.assign(rawTx, {});
                        let input = oldTx.input;
                        return uiFuncs.signed(app, rawTx, ledgerConfig, true, function␣
↪(res) {
                            oldTx.r = res.r;
                            oldTx.s = res.s;
                            oldTx.v = res.v;
                            oldTx.input = input;
                            oldTx.chainId = "0x1";
                            delete oldTx.isError;
                            delete oldTx.rawTx;
                            delete oldTx.signedTx;
                            web3.fsntx.sendRawTransaction(oldTx).then(function␣
↪(txHash) {
                                $scope.recallSwapSuccess.open()
                            })
                        })
                    }
                    $scope.notifier.info('Please, confirm transaction on Ledger.');
                    await app.getAppConfiguration(localCallback);
                }
            }
        }
    }
```

14.24 buildTakeSwapTx

```
fsntx.buildTakeSwapTx()
...
```

buildTakeSwapTx

14.24.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.
- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID` - `String|Number`: Description
- `Size` - `Number`: Description

14.24.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.24.3 Example

```
$scope.takeSwap = async function (asset_id, swap_id, amount) {
    let password = walletService.password;
    let accountData = uiFuncs.getTxData($scope);
    let walletAddress = accountData.from;
    let toAsset = [];

    try {
        await web3.fsn.getAsset(asset_id).then(function (res) {
            toAsset = res;
        });
    } catch (err) {
        console.log(err);
    }

    let data = {
        from: walletAddress,
        SwapID: swap_id.swap_id,
        Size: $scope.takeAmountSwap
    };

    console.log(data);
```

(continues on next page)

(continued from previous page)

```
    if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
        $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
    }

    try {
        await web3.fsntx.buildTakeSwapTx(data).then(function (tx) {
            tx.from = walletAddress;
            tx.chainId = _CHAINID;
            data = tx;
            if ($scope.wallet.hwType == "ledger") {
                return;
            }
            web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).then(txHash␣
↪=> {
                console.log(txHash);
            })

            return $scope.takeSwapEndConfirm.open();
        })
    } catch (err) {
        $scope.errorModal.open();
        console.log(err);
    }
    if ($scope.wallet.hwType == "ledger") {
        let ledgerConfig = {
            privKey: $scope.wallet.privKey ? $scope.wallet.getPrivateKeyString() : "",
            path: $scope.wallet.getPath(),
            hwType: $scope.wallet.getHWType(),
            hwTransport: $scope.wallet.getHWTransport()
        }
        let rawTx = data;
        var eTx = new ethUtil.Tx(rawTx);
```

14.25 takeSwap

```
fsntx.takeSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",Size:"0x1"},
↪"123456")
...
```

takeSwap buy a quantum swap CP see the top SwapID the swap ID Size the package size password the owner password

14.25.1 Parameters

1. `Object`: Description

- `from` - `String|Number` : The address for the sending account
- `gas` - `Number` : (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
- `gasPrice` - `Number|String|BN|BigNumber` : (optional) The price of gas for this transaction in wei.

- `nonce` - `Number` : (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.
- `SwapID` - `String|Number`: Description
- `Size` - `Number`: Description

2. `passwd` - `string`: Description

14.25.2 Returns

`Object`: With the following methods:

- `Object`: Description

14.25.3 Example

```
fsntx.takeSwap({from:fsn.coinbase,SwapID:
↪"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",Size:"0x1"},
↪"123456")
```

```
$scope.takeSwap = async function (asset_id, swap_id, amount) {
    let password = walletService.password;
    let accountData = uiFuncs.getTxData($scope);
    let walletAddress = accountData.from;
    let toAsset = [];

    try {
        await web3.fsn.getAsset(asset_id).then(function (res) {
            toAsset = res;
        });
    } catch (err) {
        console.log(err);
    }

    let data = {
        from: walletAddress,
        SwapID: swap_id.swap_id,
        Size: $scope.takeAmountSwap
    };

    console.log(data);

    if (!$scope.account && ($scope.wallet.hwType !== "ledger")) {
        $scope.account = web3.eth.accounts.privateKeyToAccount($scope.toHexString(
↪$scope.wallet.getPrivateKey()));
    }

    try {
        await web3.fsntx.buildTakeSwapTx(data).then(function (tx) {
            tx.from = walletAddress;
            tx.chainId = _CHAINID;
            data = tx;
            if ($scope.wallet.hwType == "ledger") {
                return;
            }
```

(continues on next page)

(continued from previous page)

```
            web3.fsn.signAndTransmit(tx, $scope.account.signTransaction).then(txHash␣
↪=> {
                console.log(txHash);
            })

            return $scope.takeSwapEndConfirm.open();
        })
    } catch (err) {
        $scope.errorModal.open();
        console.log(err);
    }
    if ($scope.wallet.hwType == "ledger") {
        let ledgerConfig = {
            privKey: $scope.wallet.privKey ? $scope.wallet.getPrivateKeyString() : "",
            path: $scope.wallet.getPath(),
            hwType: $scope.wallet.getHWType(),
            hwTransport: $scope.wallet.getHWTransport()
        }
        let rawTx = data;
        var eTx = new ethUtil.Tx(rawTx);
```

Index

Symbols

A

B

C

F

G