
Swap

Release 0.5.0

Meheret Tesfaye Batu

Dec 15, 2021

CONTENTS

1	Swap	1
2	What is a HTLC?	3
2.1	How do HTLC work?	3
2.1.1	Hash Locked	4
2.1.2	Time Locked	5
2.2	Benefits of HTLC's	5
2.2.1	Time Sensitivity	5
2.2.2	Trustless system	5
2.2.3	Validation of the Blockchain	5
2.2.4	Private Information's	5
2.2.5	Trading across multiple Cryptocurrencies	6
3	Installing Swap	7
3.1	Development	7
3.2	Dependencies	8
4	Command Line Interface (CLI)	9
4.1	swap	9
4.1.1	bitcoin	9
4.1.2	bytom	13
4.1.3	ethereum	17
4.1.4	vapor	21
4.1.5	xinfin	24
5	Utils	29
6	Bitcoin	33
6.1	Wallet	33
6.2	Hash Time Lock Contract (HTLC)	41
6.3	Transaction	44
6.3.1	NormalTransaction	46
6.3.2	FundTransaction	47
6.3.3	WithdrawTransaction	49
6.3.4	RefundTransaction	50
6.4	Solver	52
6.4.1	NormalSolver	52
6.4.2	FundSolver	52
6.4.3	WithdrawSolver	53
6.4.4	RefundSolver	54
6.5	Signature	54

6.5.1	NormalSignature	58
6.5.2	FundSignature	58
6.5.3	WithdrawSignature	59
6.5.4	RefundSignature	60
6.6	Remote Procedure Call (RPC)	61
6.7	Utils	64
7	Bytom	67
7.1	Wallet	67
7.2	Hash Time Lock Contract (HTLC)	74
7.3	Transaction	78
7.3.1	NormalTransaction	81
7.3.2	FundTransaction	82
7.3.3	WithdrawTransaction	84
7.3.4	RefundTransaction	85
7.4	Solver	87
7.4.1	NormalSolver	87
7.4.2	FundSolver	87
7.4.3	WithdrawSolver	88
7.4.4	RefundSolver	88
7.5	Signature	89
7.5.1	NormalSignature	93
7.5.2	FundSignature	94
7.5.3	WithdrawSignature	94
7.5.4	RefundSignature	95
7.6	Remote Procedure Call (RPC)	96
7.7	Utils	102
8	Ethereum	105
8.1	Wallet	105
8.2	Hash Time Lock Contract (HTLC)	113
8.3	Transaction	118
8.3.1	NormalTransaction	121
8.3.2	FundTransaction	122
8.3.3	WithdrawTransaction	124
8.3.4	RefundTransaction	125
8.4	Solver	126
8.4.1	NormalSolver	126
8.4.2	FundSolver	127
8.4.3	WithdrawSolver	127
8.4.4	RefundSolver	128
8.5	Signature	128
8.5.1	NormalSignature	132
8.5.2	FundSignature	133
8.5.3	WithdrawSignature	133
8.5.4	RefundSignature	134
8.6	Remote Procedure Call (RPC)	135
8.7	Utils	139
9	Vapor	143
9.1	Wallet	143
9.2	Hash Time Lock Contract (HTLC)	150
9.3	Transaction	154
9.3.1	NormalTransaction	157

9.3.2	FundTransaction	158
9.3.3	WithdrawTransaction	160
9.3.4	RefundTransaction	161
9.4	Solver	163
9.4.1	NormalSolver	163
9.4.2	FundSolver	163
9.4.3	WithdrawSolver	164
9.4.4	RefundSolver	164
9.5	Signature	165
9.5.1	NormalSignature	169
9.5.2	FundSignature	170
9.5.3	WithdrawSignature	170
9.5.4	RefundSignature	171
9.6	Remote Procedure Call (RPC)	172
9.7	Utils	178
10	XinFin	181
10.1	Wallet	181
10.2	Hash Time Lock Contract (HTLC)	189
10.3	Transaction	194
10.3.1	NormalTransaction	197
10.3.2	FundTransaction	198
10.3.3	WithdrawTransaction	200
10.3.4	RefundTransaction	201
10.4	Solver	202
10.4.1	NormalSolver	202
10.4.2	FundSolver	203
10.4.3	WithdrawSolver	203
10.4.4	RefundSolver	204
10.5	Signature	204
10.5.1	NormalSignature	208
10.5.2	FundSignature	208
10.5.3	WithdrawSignature	209
10.5.4	RefundSignature	210
10.6	Remote Procedure Call (RPC)	211
10.7	Utils	214
	Python Module Index	217
	Index	219

SWAP

Cryptocurrencies were created to make it possible for advanced, encrypted payments to be made between two or more people digitally, without the parties involved having to trust each other for the payment be completed. In other words, cryptocurrencies make it possible to send money reliably to other people over the internet without the money being double spent, and without people getting scammed out of their money when they try to make these digital payments.

Note: Hash Time Lock Contracts (HTLC's) are a perfect example of a payment technology for cryptocurrencies which makes all of the aforementioned things possible.

Swap is a python library for Cross-chain atomic swap between the networks of two cryptocurrencies. Cross-chain atomic swap are the cheapest and most secure way to swap cryptocurrencies. It's a brand new decentralized payment environment based on Hash Time Lock Contracts (HTLC's) protocol.

WHAT IS A HTLC?

A Hash Time Lock contract (HTLC) is essentially a type of payment in which two people agree to a financial arrangement where one party will pay the other party a certain amount of Cryptocurrency, such as Bitcoin or Bytom assets. However, because these contracts are Time Locked, the receiving party only has a certain amount of time to accept the payment, otherwise the money can be returned to the sender.

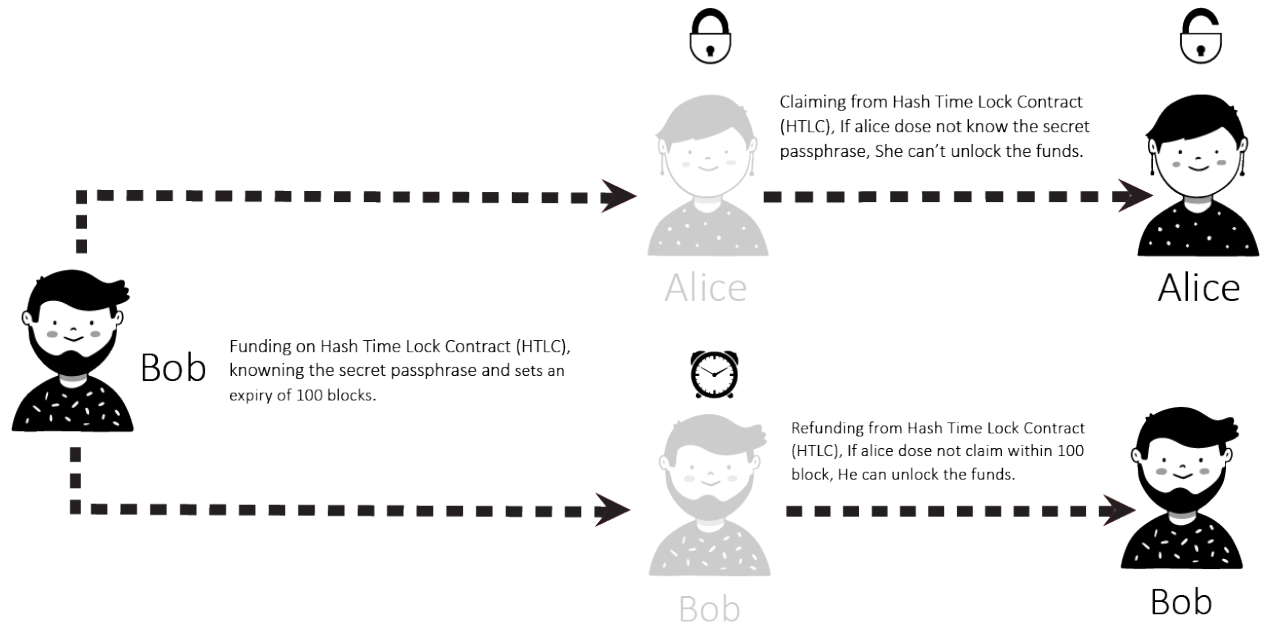
Hash time lock contracts can help to eliminate the need for third parties in contracts between two parties. Third parties that are often involved in contracts are lawyers, banks, etc. Lawyers are often required to draw up contracts, and banks are often required to help store money and then transfer it to the receiving party in the contract.

With hash time lock contracts, two parties could hypothetically set up contracts and transfer money without the need for third parties. This is because the sending party could create the conditional payment, and then the receiving party could agree to it, receive it, and help validate the transaction in the process.

This could potentially revolutionize the way that many businesses interact with one another and dramatically speed up the time that it takes for business deals to be set up.

2.1 How do HTLC work?

The way that Hash Time Lock Contracts work is that the person who will be making the payment sets up a specific hash, which represents the amount of money that will be paid. To receive the payment, the recipient will have to create a cryptographic proof of payment, and he or she will have to do this within the specified amount of time. The amount of time that the recipient has to accept the payment can vary significantly from one Time Locked contract to the next. If the recipient meets the deadline, then the money will be theirs, if he or she fails to meet the deadline, it won't. So, there is often a lot at stake when it comes to meeting deadlines from hash Time Locked contracts, when Cryptocurrencies are being exchanged.

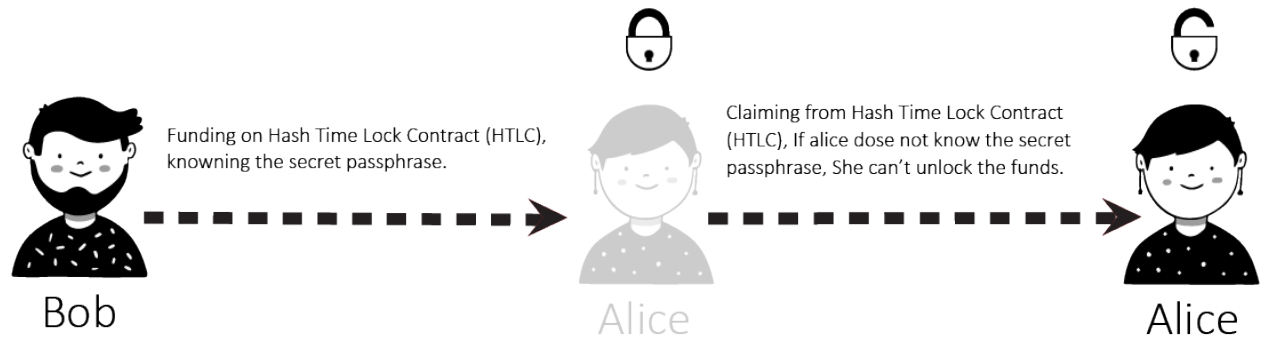


A Hash Time Lock Contract or HTLC is a class of payments that uses Hash Locked and Time Locked to require that the receiver of a payment either acknowledge receiving the payment prior to a deadline by generating cryptographic proof of payment or forfeit the ability to claim the payment, returning(refunding) it to the payer.

Hash Time Lock Contracts (HTLCs) allow payments to be securely routed across multiple payment channels which is super important because it is not optimal for a person to open a payment channel with everyone he/she is transacting with.

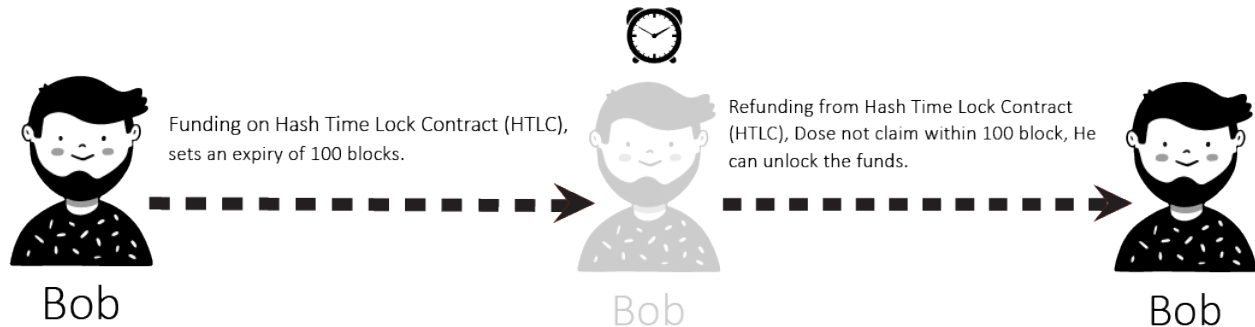
2.1.1 Hash Locked

A Hash Locked functions like "two-factor authentication" (2FA). It requires the intended recipient to provide the correct secret passphrase to claim the funds.



2.1.2 Time Locked

A Time Locked adds a “timeout” expiration date to a payment. It requires the intended recipient to claim the funds prior to the expiry. Otherwise, the transaction defaults to enabling the original sender of funds to claim a refund.



2.2 Benefits of HTLC's

There are many benefits to these types of contracts. First, because they are time sensitive, it prevents the person who is making the payment from having to wait indefinitely to find out whether or not his or her payment goes through. Second, the person who makes the payment will not have to waste his or her money if the payment is not accepted. It will simply be returned.

2.2.1 Time Sensitivity

The time sensitive nature of the transaction prevents the sender from having to wait forever to find out whether their payment went through. If the time runs out, the funds will just be sent back to the sender, so they don't have to worry and can wait for the process to unfold.

2.2.2 Trustless system

As is the case with all smart contracts, trust is not needed as the rules are already coded into the contract itself. Hash Time Lock Contracts take this one step further by implementing a time limit for recipients to acknowledge the payment.

2.2.3 Validation of the Blockchain

Transactions are validated because of the cryptographic proof of payment required by the receiver.

2.2.4 Private Information's

There are no complicated account setups or KYC/AML restrictions. Trade directly from your wallet with a counterparty of your choice. Only the parties involved know the details of the trade.

2.2.5 Trading across multiple Cryptocurrencies

HTLC makes Cross-chain transactions easier and more secure than ever. Cross chain transactions are the next step in the evolution of Cryptocurrency adoption. The easier it becomes to unite the hundreds of blockchain's that currently exist in silos, the faster the technology as a whole can begin to scale and achieve mass adoption.

INSTALLING SWAP

The easiest way to install Swap is via pip:

```
$ pip install swap
```

If you want to run the latest version of the code, you can install from git:

```
$ pip install git+git://github.com/meherett/swap.git
```

For the versions available, see the [tags on this repository](#).

3.1 Development

We welcome pull requests. To get started, just fork this [github repository](#), clone it locally, and run:

```
$ pip install -e . -r requirements.txt
```

Once you have installed, type `swap` to verify that it worked:

```
$ swap
Usage: swap [OPTIONS] COMMAND [ARGS]...

Options:
  -v, --version  Show Swap version and exit.
  -h, --help     Show this message and exit.

Commands:
  bitcoin  Select Bitcoin provider.
  bytom    Select Bytom provider.
  vapor    Select Vapor provider.
```

3.2 Dependencies

Swap has the following dependencies:

- `solc v0.8.3` - Solidity, the Smart Contract Programming Language
- `bytom-wallet-desktop` - version 1.1.0 or greater.
- `vapor-wallet-desktop` - version 1.1.7 or greater.
- `pip` - To install packages from the Python Package Index and other indexes
- `python3` version 3.6 or greater

COMMAND LINE INTERFACE (CLI)

After you have installed, type `swap` to verify that it worked:

```
$ swap
Usage: swap [OPTIONS] COMMAND [ARGS]...

Options:
  -v, --version  Show Swap version and exit.
  -h, --help     Show this message and exit.

Commands:
  bitcoin  Select Bitcoin provider.
  bytom    Select Bytom provider.
  ethereum Select Ethereum provider.
  vapor    Select Vapor provider.
  xinfin   Select XinFin provider.
```

4.1 swap

```
swap [OPTIONS] COMMAND [ARGS]...
```

Options

-v, --version
Show Swap version and exit.

4.1.1 bitcoin

Select Bitcoin provider.

```
swap bitcoin [OPTIONS] COMMAND [ARGS]...
```

decode

Select Bitcoin Transaction raw decoder.

```
swap bitcoin decode [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>

Required Set Bitcoin transaction raw.

-i, --indent <indent>

Set json indent.

Default 4

-o, --offline <offline>

Set Offline decode transaction raw.

Default True

fund

Select Bitcoin Fund transaction builder.

```
swap bitcoin fund [OPTIONS]
```

Options

-a, --address <address>

Required Set Bitcoin sender address.

-ca, --contract-address <contract_address>

Required Set Bitcoin Hash Time Lock Contract (HTLC) address.

-am, --amount <amount>

Required Set Bitcoin fund amount.

-u, --unit <unit>

Set Bitcoin fund amount unit.

Default Satoshi

-n, --network <network>

Set Bitcoin network.

Default mainnet

-v, --version <version>

Set Bitcoin transaction version.

Default 2

htlc

Select Bitcoin Hash Time Lock Contract (HTLC) builder.

```
swap bitcoin htlc [OPTIONS]
```

Options

- sh, --secret-hash** <secret_hash>
Required Set secret 256 hash.
- ra, --recipient-address** <recipient_address>
Required Set Bitcoin recipient address.
- sa, --sender-address** <sender_address>
Required Set Bitcoin sender address.
- e, --endtime** <endtime>
Required Set Expiration block time (Seconds).
- n, --network** <network>
Set Bitcoin network.

Default mainnet
- i, --indent** <indent>
Set json indent.

Default 4

refund

Select Bitcoin Refund transaction builder.

```
swap bitcoin refund [OPTIONS]
```

Options

- a, --address** <address>
Required Set Bitcoin sender address.
- th, --transaction-hash** <transaction_hash>
Required Set Bitcoin funded transaction hash/id.
- n, --network** <network>
Set Bitcoin network.

Default mainnet
- v, --version** <version>
Set Bitcoin transaction version.

Default 2

sign

Select Bitcoin Transaction raw signer.

```
swap bitcoin sign [OPTIONS]
```

Options

- xpk, --xprivate-key <xprivate_key>**
Required Set Bitcoin root xprivate key.
- tr, --transaction-raw <transaction_raw>**
Required Set Bitcoin unsigned transaction raw.
- b, --bytecode <bytecode>**
Set Bitcoin witness HTLC bytecode. [default: None]
- sk, --secret-key <secret_key>**
Set secret key. [default: None]
- e, --endtime <endtime>**
Set Expiration block timestamp. [default: None]
- ac, --account <account>**
Set Bitcoin derivation from account.
Default 1
- ch, --change <change>**
Set Bitcoin derivation from change.
Default False
- ad, --address <address>**
Set Bitcoin derivation from address.
Default 1
- p, --path <path>**
Set Bitcoin derivation from path. [default: None]
- v, --version <version>**
Set Bitcoin transaction version.
Default 2

submit

Select Bitcoin Transaction raw submitter.

```
swap bitcoin submit [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>
Required Set signed Bitcoin transaction raw.

-e, --endpoint <endpoint>
 Set submission endpoint API name.

withdraw

Select Bitcoin Withdraw transaction builder.

```
swap bitcoin withdraw [OPTIONS]
```

Options

-a, --address <address>
Required Set Bitcoin recipient address.

-th, --transaction-hash <transaction_hash>
Required Set Bitcoin funded transaction hash/id.

-n, --network <network>
 Set Bitcoin network.

Default mainnet

-v, --version <version>
 Set Bitcoin transaction version.

Default 2

4.1.2 bytom

Select Bytom provider.

```
swap bytom [OPTIONS] COMMAND [ARGS]...
```

decode

Select Bytom Transaction raw decoder.

```
swap bytom decode [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>
Required Set Bytom transaction raw.

-i, --indent <indent>
Set json indent.

Default 4

fund

Select Bytom Fund transaction builder.

```
swap bytom fund [OPTIONS]
```

Options

-a, --address <address>
Required Set Bytom sender address.

-ca, --contract-address <contract_address>
Required Set Bytom Hash Time Lock Contract (HTLC) address.

-am, --amount <amount>
Required Set Bytom fund amount.

-u, --unit <unit>
Set Bytom amount unit.

Default NEU

-as, --asset <asset>
Set Bytom asset id.

Default ff

-n, --network <network>
Set Bytom network.

Default mainnet

htlc

Select Bytom Hash Time Lock Contract (HTLC) builder.

```
swap bytom htlc [OPTIONS]
```

Options

- sh, --secret-hash** <secret_hash>
Required Set secret 256 hash.
- rpk, --recipient-public-key** <recipient_public_key>
Required Set Bytom recipient public key.
- spk, --sender-public-key** <sender_public_key>
Required Set Bytom sender public key.
- e, --endblock** <endblock>
Required Set Bytom expiration block height.
- n, --network** <network>
Set Bytom network.

Default mainnet
- i, --indent** <indent>
Set json indent.

Default 4

refund

Select Bytom Refund transaction builder.

```
swap bytom refund [OPTIONS]
```

Options

- a, --address** <address>
Required Set Bytom sender address.
- th, --transaction-hash** <transaction_hash>
Required Set Bytom funded transaction id/hash.
- as, --asset** <asset>
Set Bytom asset id.

Default `ffffffffffffffffffffffffffffffffffffffff`
- n, --network** <network>
Set Bytom network.

Default mainnet

sign

Select Bytom Transaction raw signer.

```
swap bytom sign [OPTIONS]
```

Options

- xpk, --xprivate-key <xprivate_key>**
Required Set Bytom xprivate key.
- tr, --transaction-raw <transaction_raw>**
Required Set Bytom unsigned transaction raw.
- b, --bytecode <bytecode>**
Set Bytom witness HTLC bytecode. [default: None]
- sk, --secret-key <secret_key>**
Set secret key. [default: None]
- ac, --account <account>**
Set Bytom derivation from account.
Default 1
- ch, --change <change>**
Set Bytom derivation from change.
Default False
- ad, --address <address>**
Set Bytom derivation from address.
Default 1
- p, --path <path>**
Set Bytom derivation from path. [default: None]
- i, --indexes <indexes>**
Set Bytom derivation from indexes. [default: None]

submit

Select Bytom Transaction raw submitter.

```
swap bytom submit [OPTIONS]
```

Options

- tr, --transaction-raw <transaction_raw>**
Required Set signed Bytom transaction raw.

withdraw

Select Bytom Withdraw transaction builder.

```
swap bytom withdraw [OPTIONS]
```

Options

- a, --address** <address>
 Required Set Bytom recipient address.
- th, --transaction-hash** <transaction_hash>
 Required Set Bytom funded transaction hash/id.
- as, --asset** <asset>
 Set Bytom asset id.
 Default `ffffffffffffffffffffffffffffffffffffffff`
- n, --network** <network>
 Set Bytom network.
 Default `mainnet`

4.1.3 ethereum

Select Ethereum provider.

```
swap ethereum [OPTIONS] COMMAND [ARGS]...
```

decode

Select Ethereum Transaction raw decoder.

```
swap ethereum decode [OPTIONS]
```

Options

- tr, --transaction-raw** <transaction_raw>
 Required Set Ethereum transaction raw.
- i, --indent** <indent>
 Set json indent.
 Default `4`

fund

Select Ethereum Fund transaction builder.

```
swap ethereum fund [OPTIONS]
```

Options

- sh, --secret-hash <secret_hash>**
Required Set secret 256 hash.
- ra, --recipient-address <recipient_address>**
Required Set Ethereum recipient address.
- sa, --sender-address <sender_address>**
Required Set Ethereum sender address.
- e, --endtime <endtime>**
Required Set Expiration block timestamp.
- am, --amount <amount>**
Required Set Ethereum fund amount.
- u, --unit <unit>**
Set Ethereum fund amount unit.
Default Wei
- ca, --contract-address <contract_address>**
Set Ethereum HTLC contact address. [default: None]
- ta, --token-address <token_address>**
Set Ethereum ERC20 token address. [default: None]
- n, --network <network>**
Set Ethereum network.
Default mainnet
- e20, --erc20 <erc20>**
Set Enable Ethereum ERC20 token contract.
Default False

htlc

Select Ethereum Hash Time Lock Contract (HTLC) builder.

```
swap ethereum htlc [OPTIONS]
```

Options

- ca, --contract-address <contract_address>**
Set Ethereum HTLC contact address. [default: None]
- n, --network <network>**
Set Ethereum network.
Default mainnet
- e20, --erc20 <erc20>**
Set Enable Ethereum ERC20 token contract.
Default False
- i, --indent <indent>**
Set json indent.
Default 4

refund

Select Ethereum Refund transaction builder.

swap ethereum refund [OPTIONS]

Options

- th, --transaction-hash <transaction_hash>**
Required Set Ethereum funded transaction hash/id.
- a, --address <address>**
Required Set Ethereum sender address.
- ca, --contract-address <contract_address>**
Set Ethereum HTLC contact address. [default: None]
- n, --network <network>**
Set Ethereum network.
Default mainnet
- e20, --erc20 <erc20>**
Set Enable Ethereum ERC20 token contract.
Default False

sign

Select Ethereum Transaction raw signer.

swap ethereum sign [OPTIONS]

Options

- xpk, --xprivate-key <xprivate_key>**
Required Set Ethereum root xprivate key.
- tr, --transaction-raw <transaction_raw>**
Required Set Ethereum unsigned transaction raw.
- ac, --account <account>**
Set Ethereum derivation from account.
Default 0
- ch, --change <change>**
Set Ethereum derivation from change.
Default False
- ad, --address <address>**
Set Ethereum derivation from address.
Default 0
- p, --path <path>**
Set Ethereum derivation from path. [default: None]

submit

Select Ethereum Transaction raw submitter.

`swap ethereum submit [OPTIONS]`

Options

- tr, --transaction-raw <transaction_raw>**
Required Set signed Ethereum transaction raw.

withdraw

Select Ethereum Withdraw transaction builder.

`swap ethereum withdraw [OPTIONS]`

Options

- th, --transaction-hash <transaction_hash>**
Required Set Ethereum funded transaction hash/id.
- a, --address <address>**
Required Set Ethereum recipient address.
- sk, --secret-key <secret_key>**
Required Set secret password/passphrase.
- ca, --contract-address <contract_address>**
Set Ethereum HTLC contact address. [default: None]

-n, --network <network>
Set Ethereum network.

Default mainnet

-e20, --erc20 <erc20>
Set Enable Ethereum ERC20 token contract.

Default False

4.1.4 vapor

Select Vapor provider.

```
swap vapor [OPTIONS] COMMAND [ARGS]...
```

decode

Select Vapor Transaction raw decoder.

```
swap vapor decode [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>
Required Set Vapor transaction raw.

-i, --indent <indent>
Set json indent.

Default 4

fund

Select Vapor Fund transaction builder.

```
swap vapor fund [OPTIONS]
```

Options

-a, --address <address>
Required Set Vapor sender address.

-ca, --contract-address <contract_address>
Required Set Vapor Hash Time Lock Contract (HTLC) address.

-am, --amount <amount>
Required Set Vapor fund amount.

-u, --unit <unit>
Set Vapor amount unit.

Default NEU

-as, --asset <asset>
Set Vapor asset id.

Default ff

-n, --network <network>
Set Vapor network.

Default mainnet

htlc

Select Vapor Hash Time Lock Contract (HTLC) builder.

```
swap vapor htlc [OPTIONS]
```

Options

-sh, --secret-hash <secret_hash>
Required Set secret 256 hash.

-rpk, --recipient-public-key <recipient_public_key>
Required Set Vapor recipient public key.

-spk, --sender-public-key <sender_public_key>
Required Set Vapor sender public key.

-e, --endblock <endblock>
Required Set Vapor expiration block height.

-n, --network <network>
Set Vapor network.

Default mainnet

-i, --indent <indent>
Set json indent.

Default 4

refund

Select Vapor Refund transaction builder.

```
swap vapor refund [OPTIONS]
```

Options

-a, --address <address>
Required Set Vapor sender address.

-th, --transaction-hash <transaction_hash>
Required Set Vapor funded transaction id/hash.

-as, --asset <asset>
Set Vapor asset id.

Default `ffffffffffffffffffffffffffffffffffffffff`

-n, --network <network>
Set Vapor network.

Default mainnet

sign

Select Vapor Transaction raw signer.

```
swap vapor sign [OPTIONS]
```

Options

-xpk, --xprivate-key <xprivate_key>
Required Set Vapor xprivate key.

-tr, --transaction-raw <transaction_raw>
Required Set Vapor unsigned transaction raw.

-b, --bytecode <bytecode>
Set Vapor witness HTLC bytecode. [default: None]

-sk, --secret-key <secret_key>
Set secret key. [default: None]

-ac, --account <account>
Set Vapor derivation from account.

Default 1

-ch, --change <change>
Set Vapor derivation from change.

Default False

-ad, --address <address>
Set Vapor derivation from address.

Default 1

-p, --path <path>
Set Vapor derivation from path. [default: None]

-i, --indexes <indexes>
Set Vapor derivation from indexes. [default: None]

submit

Select Vapor Transaction raw submitter.

```
swap vapor submit [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>
Required Set signed Vapor transaction raw.

withdraw

Select Vapor Withdraw transaction builder.

```
swap vapor withdraw [OPTIONS]
```

Options

-a, --address <address>
Required Set Vapor recipient address.

-th, --transaction-hash <transaction_hash>
Required Set Vapor funded transaction hash/id.

-as, --asset <asset>
Set Vapor asset id.

Default `ffffffffffffffffffffffffffffffffffffffff`

-n, --network <network>
Set Vapor network.

Default `mainnet`

4.1.5 xinfm

Select XinFin provider.

```
swap xinfm [OPTIONS] COMMAND [ARGS]...
```

decode

Select XinFin Transaction raw decoder.

```
swap xinfm decode [OPTIONS]
```

Options

-tr, --transaction-raw <transaction_raw>
Required Set XinFin transaction raw.

-i, --indent <indent>
Set json indent.

Default `4`

fund

Select XinFin Fund transaction builder.

```
swap xinfund fund [OPTIONS]
```

Options

- sh, --secret-hash** <secret_hash>
Required Set secret 256 hash.
- ra, --recipient-address** <recipient_address>
Required Set XinFin recipient address.
- sa, --sender-address** <sender_address>
Required Set XinFin sender address.
- e, --endtime** <endtime>
Required Set Expiration block timestamp.
- am, --amount** <amount>
Required Set XinFin fund amount.
- u, --unit** <unit>
Set XinFin fund amount unit.
Default Wei
- ca, --contract-address** <contract_address>
Set XinFin HTLC contract address. [default: None]
- ta, --token-address** <token_address>
Set XinFin XRC20 token address. [default: None]
- n, --network** <network>
Set XinFin network.
Default mainnet
- x20, --xrc20** <xrc20>
Set Enable XinFin XRC20 token contract.
Default False

htlc

Select XinFin Hash Time Lock Contract (HTLC) builder.

```
swap xinfund htlc [OPTIONS]
```

Options

- ca, --contract-address** <contract_address>
Set XinFin HTLC contact address. [default: None]
- n, --network** <network>
Set XinFin network.
Default mainnet
- x20, --xrc20** <xrc20>
Set Enable XinFin XRC20 token contract.
Default False
- i, --indent** <indent>
Set json indent.
Default 4

refund

Select XinFin Refund transaction builder.

```
swap xinfm refund [OPTIONS]
```

Options

- th, --transaction-hash** <transaction_hash>
Required Set XinFin funded transaction hash/id.
- a, --address** <address>
Required Set XinFin sender address.
- ca, --contract-address** <contract_address>
Set XinFin HTLC contact address. [default: None]
- n, --network** <network>
Set XinFin network.
Default mainnet
- x20, --xrc20** <xrc20>
Set Enable XinFin XRC20 token contract.
Default False

sign

Select XinFin Transaction raw signer.

```
swap xinfm sign [OPTIONS]
```

Options

- xpk, --xprivate-key <xprivate_key>**
Required Set XinFin root xprivate key.
- tr, --transaction-raw <transaction_raw>**
Required Set XinFin unsigned transaction raw.
- ac, --account <account>**
Set XinFin derivation from account.
Default 0
- ch, --change <change>**
Set XinFin derivation from change.
Default False
- ad, --address <address>**
Set XinFin derivation from address.
Default 0
- p, --path <path>**
Set XinFin derivation from path. [default: None]

submit

Select XinFin Transaction raw submitter.

```
swap xinfm submit [OPTIONS]
```

Options

- tr, --transaction-raw <transaction_raw>**
Required Set signed XinFin transaction raw.

withdraw

Select XinFin Withdraw transaction builder.

```
swap xinfm withdraw [OPTIONS]
```

Options

- th, --transaction-hash <transaction_hash>**
Required Set XinFin funded transaction hash/id.
- a, --address <address>**
Required Set XinFin recipient address.
- sk, --secret-key <secret_key>**
Required Set secret password/passphrase.
- ca, --contract-address <contract_address>**
Set XinFin HTLC contract address. [default: None]

-n, --network <network>
Set XinFin network.

Default mainnet

-x20, --xrc20 <xrc20>
Set Enable XinFin XRC20 token contract.

Default False

UTILS

class swap.utils.NestedNamespace(dictionary, **kwargs)

swap.utils.generate_passphrase(length: int = 32) → str
Generate entropy hex string.

Parameters length (int) – Passphrase length, default to 32.

Returns str – Passphrase hex string.

```
>>> from swap.utils import generate_passphrase
>>> generate_passphrase(length=32)
"N39rPfa3QvF2Tm2nPyoBpXNiBFXJywTz"
```

swap.utils.generate_entropy(strength: int = 128) → str
Generate entropy hex string.

Parameters strength (int) – Entropy strength, default to 128.

Returns str – Entropy hex string.

```
>>> from swap.utils import generate_entropy
>>> generate_entropy(strength=128)
"ee535b143b0d9d1f87546f9df0d06b1a"
```

swap.utils.generate_mnemonic(language: str = 'english', strength: int = 128) → str
Generate mnemonic words.

Parameters

- **language** (str) – Mnemonic language, default to english.
- **strength** (int) – Entropy strength, default to 128.

Returns str – Mnemonic words.

```
>>> from swap.utils import generate_mnemonic
>>> generate_mnemonic(language="french")
"sceptre capter sequence girafe absolu relatif fleur zoologie muscle sirop saboter_
↳parure"
```

swap.utils.get_current_timestamp(plus: int = 0) → int
Get current timestamp.

Parameters plus (int) – Add seconds on current time, default to 0.

Returns int – Current timestamp.

```
>>> from swap.utils import get_current_timestamp
>>> get_current_timestamp()
1623869258
```

`swap.utils.is_entropy(entropy: str) → bool`
Check entropy hex string.

Parameters `entropy (str)` – Mnemonic words.

Returns `bool` – Entropy valid/invalid.

```
>>> from swap.utils import is_entropy
>>> is_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
True
```

`swap.utils.is_mnemonic(mnemonic: str, language: Optional[str] = None) → bool`
Check mnemonic words.

Parameters

- `mnemonic (str)` – Mnemonic words.
- `language (str)` – Mnemonic language, default to None.

Returns `bool` – Mnemonic valid/invalid.

```
>>> from swap.utils import is_mnemonic
>>> is_mnemonic(mnemonic="sceptre capter sequence girafe absolu relatif fleur,
↳zoologie muscle sirop saboter parure")
True
```

`swap.utils.get_entropy_strength(entropy: str) → int`
Get entropy strength.

Parameters `entropy (str)` – Entropy hex string.

Returns `int` – Entropy strength.

```
>>> from swap.utils import get_entropy_strength
>>> get_entropy_strength(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
128
```

`swap.utils.get_mnemonic_strength(mnemonic: str, language: Optional[str] = None) → int`
Get mnemonic strength.

Parameters

- `mnemonic (str)` – Mnemonic words.
- `language (str)` – Mnemonic language, default to None.

Returns `int` – Mnemonic strength.

```
>>> from swap.utils import get_mnemonic_strength
>>> get_mnemonic_strength(mnemonic="sceptre capter sequence girafe absolu relatif,
↳fleur zoologie muscle sirop saboter parure")
128
```

`swap.utils.get_mnemonic_language(mnemonic: str) → str`
Get mnemonic language.

Parameters `mnemonic (str)` – Mnemonic words.

Returns `str` – Mnemonic language.

```
>>> from swap.utils import get_mnemonic_language
>>> get_mnemonic_language(mnemonic="sceptre capter sequence girafe absolu relatif_
↪ fleur zoologie muscle sirop saboter parure")
"french"
```

`swap.utils.entropy_to_mnemonic(entropy: str, language: str = 'english') → str`
Get mnemonic from entropy hex string.

Parameters

- **entropy (str)** – Entropy hex string.
- **language (str)** – Mnemonic language, default to english.

Returns `str` – Mnemonic words.

```
>>> from swap.utils import entropy_to_mnemonic
>>> entropy_to_mnemonic(entropy="ee535b143b0d9d1f87546f9df0d06b1a", language="korean
↪ ")
"
```

`swap.utils.mnemonic_to_entropy(mnemonic: str, language: Optional[str] = None) → str`
Get entropy from mnemonic words.

Parameters

- **mnemonic (str)** – Mnemonic words.
- **language (str)** – Mnemonic language, default to english.

Returns `str` – Entropy hex string.

```
>>> from swap.utils import mnemonic_to_entropy
>>> mnemonic_to_entropy(mnemonic=" ", language="korean")
"ee535b143b0d9d1f87546f9df0d06b1a"
```

`swap.utils.sha256(data: Union[str, bytes]) → str`
SHA256 hash.

Parameters `data (str, bytes)` – Any string/bytes data.

Returns `str` – SHA256 hash.

```
>>> from swap.utils import sha256
>>> sha256(data="Hello Meheret!")
"3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb"
```

`swap.utils.double_sha256(data: Union[str, bytes]) → str`
Double SHA256 hash.

Parameters `data (str, bytes)` – Any string/bytes data.

Returns `str` – Double SHA256 hash.

```
>>> from swap.utils import double_sha256
>>> double_sha256(data="Hello Meheret!")
"821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e0158"
```

`swap.utils.clean_transaction_raw(transaction_raw: str) → str`
Clean transaction raw.

Parameters `transaction_raw (str)` – Any transaction raw.

Returns `str` – Cleaned transaction raw.

```
>>> from swap.utils import clean_transaction_raw
>>> clean_transaction_raw(transaction_raw=
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcTluZlseDAyc3lmd2Q3bnBlaGZ4ejRsZGRoenFzdmUyZnU2dmM3
↳ ")
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcTluZlseDAyc3lmd2Q3bnBlaGZ4ejRsZGRoenFzdmUyZnU2dmM3
↳ "
```

BITCOIN

Bitcoin is a Cryptocurrency. It is a decentralized digital currency without a central bank or single administrator that can be sent from user to user on the peer-to-peer bitcoin network without the need for intermediaries.

For more <https://bitcoin.org>

6.1 Wallet

The implementation of Hierarchical Deterministic (HD) wallets generator for Bitcoin blockchain.

class `swap.providers.bitcoin.wallet.Wallet`(*network: str = 'mainnet', use_default_path: bool = False*)
Bitcoin hierarchical deterministic wallet.

Parameters

- **network** (*str*) – Bitcoin network, defaults to `mainnet`.
- **use_default_path** (*bool*) – Use default derivation path, defaults to `False`.

Returns `Wallet` – Bitcoin wallet instance.

Note: Bitcoin has only two networks, `mainnet` and `testnet`.

from_entropy(*entropy: str, language: str = 'english', passphrase: Optional[str] = None*) →
swap.providers.bitcoin.wallet.Wallet
Master from Entropy.

Parameters

- **entropy** (*str*) – Bitcoin entropy.
- **language** (*str*) – Bitcoin language, default to `english`.
- **passphrase** (*str*) – Bitcoin passphrase, default to `None`.

Returns `Wallet` – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_mnemonic(*mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None*) →
swap.providers.bitcoin.wallet.Wallet
Master from Mnemonic.

Parameters

- **mnemonic** (*str*) – Bitcoin mnemonic.
- **language** (*str*) – Bitcoin language, default to english.
- **passphrase** (*str*) – Bitcoin passphrase, default to None.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park,
↳ clown build renew illness fault")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_seed(*seed: str*) → *swap.providers.bitcoin.wallet.Wallet*

Master from Seed.

Parameters **seed** (*str*) – Bitcoin seed.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_seed(seed=
↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ ")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_xprivate_key(*xprivate_key: str, strict: bool = True*) → *swap.providers.bitcoin.wallet.Wallet*

Master from Root XPrivate Key.

Parameters

- **xprivate_key** (*str*) – Bitcoin root xprivate key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv
↳ ")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_xpublic_key(*xpublic_key: str, strict: bool = True*) → *swap.providers.bitcoin.wallet.Wallet*

Master from Root XPublic Key.

Parameters

- **xpublic_key** (*str*) – Bitcoin root xpublic key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xpublic_key(xpublic_key=
↳ "tpubD6NzVbkrYhZ4XpK9BpGhJuvfHJMeAggFcHCZH3NKsSbcetttiJnp184yx2cp2uJyapPQLt7LGTUzvnKWbdgKBk
↳ ")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_wif(wif: str) → *swap.providers.bitcoin.wallet.Wallet*

Master from Wallet Important Format (WIF).

Parameters **wif** (str) – Bitcoin wallet important format.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_wif(wif="cS6utJFQYTQEAY455hRQ5nardhCCoc2yf4M45P71ve5Dx44ag7qg")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_private_key(private_key) → *swap.providers.bitcoin.wallet.Wallet*

Master from Private Key.

Parameters **private_key** (str) – Bitcoin private key.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_private_key(private_key=
↳ "86e4296c4b8804b952933ddf9b786a0bad1049c1d5b372e43f9336eb4ac2fcb6")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_path(path: str) → *swap.providers.bitcoin.wallet.Wallet*

Drive Bitcoin wallet from path.

Parameters **path** (str) – Bitcoin derivation path.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet", use_default_path=False)
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/1'/0'/0/0")
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

from_index(index: int, hardened: bool = False) → *swap.providers.bitcoin.wallet.Wallet*

Drive Bitcoin from index.

Parameters

- **index** (int) – Bitcoin derivation index.
- **hardened** (bool) – Use harden, default to False.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet", use_default_path=False)
```

(continues on next page)

(continued from previous page)

```
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_index(index=44, hardened=True)
>>> wallet.from_index(index=1, hardened=True)
>>> wallet.from_index(index=0, hardened=True)
>>> wallet.from_index(index=0)
>>> wallet.from_index(index=0)
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
```

clean_derivation() → *swap.providers.bitcoin.wallet.Wallet*
Clean derivation Bitcoin.

Returns Wallet – Bitcoin wallet instance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.path()
"m/44'/1'/0'/0/0"
>>> wallet.clean_derivation()
<swap.providers.bitcoin.wallet.Wallet object at 0x040DA268>
>>> wallet.path()
None
```

strength() → Optional[int]
Get Bitcoin strength.

Returns int – Bitcoin strength.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.strength()
128
```

entropy() → Optional[str]
Get Bitcoin entropy.

Returns str – Bitcoin entropy.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park_
↳ clown build renew illness fault")
>>> wallet.entropy()
"ed0802d701a033776811601dd6c5c4a9"
```

mnemonic() → Optional[str]
Get Bitcoin mnemonic.

Returns str – Bitcoin mnemonic.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.mnemonic()
"unfair divorce remind addict add roof park clown build renew illness fault"
```

passphrase() → Optional[str]

Get Bitcoin passphrase.

Returns str – Bitcoin passphrase.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9", passphrase="meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]

Get Bitcoin language.

Returns str – Bitcoin language.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.language()
"english"
```

seed() → Optional[str]

Get Bitcoin seed.

Returns str – Bitcoin seed.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.seed()

"1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea..."
```

root_xprivate_key(encoded: bool = True) → Optional[str]

Get Bitcoin root xprivate key.

Parameters **encoded** (bool) – Encoded root xprivate key, default to True.

Returns str – Bitcoin root xprivate key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xprivate_key()

"tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv..."
```

root_xpublic_key(encoded: bool = True) → Optional[str]

Get Bitcoin root xpublic key.

Parameters `encoded` (*bool*) – Encoded root xpublic key, default to True.

Returns `str` – Bitcoin root xpublic key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xpublic_key()

↪ "tpubD6NzVbkrYhZ4XpK9BpGhJuvfHJMeAggFcHCZH3NKsSbcetttiJnp184yx2cp2uJyapPQLt7LGTLUZvnKWbdgKBk
↪ "
```

xprivate_key(*encoded: bool = True*) → `Optional[str]`

Get Bitcoin root xprivate key.

Parameters `encoded` (*bool*) – Encoded root xprivate key, default to True.

Returns `str` – Bitcoin root xprivate key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.xprivate_key()

↪ "tprv8kqWVfMdSgo9WhUAxbmL6GNW4ivePvEZBu8QiiRfMXbVDgnHx16vndnAsv7Uds4iFvjMpdJiB6q6hhh753fRb89
↪ "
```

xpublic_key(*encoded: bool = True*) → `Optional[str]`

Get Bitcoin xpublic key.

Parameters `encoded` (*bool*) – Encoded xprivate key, default to True.

Returns `str` – Bitcoin xpublic key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.xpublic_key()

↪ "tpubDHYXe5Psb4UpQAVxrFRvVg2cdkSaZFRTmCjC1ETxmoPt4B34aPvWy8Q343tUsTaCQCisJJVpzgyP1NQ3mffY7oF6
↪ "
```

uncompressed(*compressed: Optional[str] = None*) → `str`

Get Bitcoin Uncompressed public key.

Parameters `compressed` (*str*) – Compressed public key, default to None.

Returns `str` – Bitcoin Uncompressed public key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.uncompressed()

↪ "f4206f9c6d35f50b3b05edc13118ab64d27959d0b7412638bfea5d132b3fb36c6d9515384318aab7fc4d15d5a1e
↪ "
```

(continues on next page)

(continued from previous page)

compressed(*uncompressed: Optional[str] = None*) → str

Get Bitcoin Compressed public key.

Parameters **uncompressed** (*str*) – Uncompressed public key, default to None.**Returns** *str* – Bitcoin Compressed public key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.compressed()
"02f4206f9c6d35f50b3b05edc13118ab64d27959d0b7412638bfea5d132b3fb36c"
```

chain_code() → str

Get Bitcoin chain code.

Returns *str* – Bitcoin chain code.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.chain_code()
"cbe00345c4cfa83dc315e52b1d5acaf2c6fce1bc8760f02696c05c3a94171304"
```

private_key() → str

Get Bitcoin private key.

Returns *str* – Bitcoin private key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.private_key()
"86e4296c4b8804b952933ddf9b786a0bad1049c1d5b372e43f9336eb4ac2fcb6"
```

public_key(*compressed: bool = True, private_key: Optional[str] = None*) → str

Get Bitcoin Public key.

Parameters

- **compressed** (*bool*) – Compressed public key, default to True.
- **private_key** (*str*) – Private key hex string, default to None.

Returns *str* – Bitcoin public key.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.public_key()
"02f4206f9c6d35f50b3b05edc13118ab64d27959d0b7412638bfea5d132b3fb36c"
```

path() → Optional[str]

Get Bitcoin path.

Returns str – Bitcoin path.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.path()
"m/44'/1'/0'/0/0"
```

wif() → str

Get Bitcoin important format (WIF).

Returns str – Bitcoin important format.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.wif()
"cS6utJFQYTQEAY455hRQ5nardhCCoc2yf4M45P71ve5Dx44ag7qg"
```

hash(private_key: Optional[str] = None) → str

Get Bitcoin public key/address hash.

Returns str – Bitcoin public key/address hash.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.hash()
"e00ff2a640b7ce2d336860739169487a57f84b15"
```

address() → str

Get Bitcoin address.

Returns str – Bitcoin address.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.address()
"n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a"
```

balance(unit: str = 'Satoshi') → Union[int, float]

Get Bitcoin balance.

Parameters unit (str) – Bitcoin unit, default to Satoshi.

Returns int, float – Bitcoin balance.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.balance(unit="BTC")
0.2
```

utxos(*limit: int = 15*) → list

Get Bitcoin unspent transaction output (UTXO's).

Parameters **limit** (*int*) – Limit of UTXO's, default is 15.

Returns list – Bitcoin unspent transaction outputs.

```
>>> from swap.providers.bitcoin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy("ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/1'/0'/0/0")
>>> wallet.utxos()
[{'index': 0, 'hash':
  ↳ '9d60a8b4dd16d4bf02835a21a3e9154e636ba06ad55368f36114eb7e930b35e8', 'output_
  ↳ index': 1, 'amount': 100000, 'script':
  ↳ '76a914e00ff2a640b7ce2d336860739169487a57f84b1588ac'}, {'index': 1, 'hash':
  ↳ '77ecb5ffe0f85454183bcab0cf1e15bfc62dc86cbdeaf374224ba03cb5cd7d29', 'output_
  ↳ index': 0, 'amount': 10000, 'script':
  ↳ '76a914e00ff2a640b7ce2d336860739169487a57f84b1588ac'}, {'index': 2, 'hash':
  ↳ 'e3ed50900a06990c123f3e87187009ce124cb65a46cd45eba5773fb0979fce43', 'output_
  ↳ index': 0, 'amount': 1797372, 'script':
  ↳ '76a914e00ff2a640b7ce2d336860739169487a57f84b1588ac'}]]
```

6.2 Hash Time Lock Contract (HTLC)

Bitcoin Hash Time Lock Contract (HTLC).

class swap.providers.bitcoin.htlc.**HTLC**(*network: str = 'mainnet', contract_address: Optional[str] = None*)

Bitcoin Hash Time Lock Contract (HTLC).

Parameters **network** (*str*) – Bitcoin network, defaults to mainnet.

Returns HTLC – Bitcoin HTLC instance.

Note: Bitcoin has only two networks, mainnet and testnet.

build_htlc(*secret_hash: str, recipient_address: str, sender_address: str, endtime: int*) → *swap.providers.bitcoin.htlc.HTLC*

Build Bitcoin Hash Time Lock Contract (HTLC).

Parameters

- **secret_hash** (*str*) – secret sha-256 hash.
- **recipient_address** (*str*) – Bitcoin recipient address.
- **sender_address** (*str*) – Bitcoin sender address.
- **endtime** (*int*) – Expiration block time (Seconds).

Returns HTLC – Bitcoin Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", sender_address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", endtime=1624687630)
<swap.providers.bitcoin.htlc.HTLC object at 0x0409DAF0>
```

from_opcode(opcode: str) → *swap.providers.bitcoin.htlc.HTLC*

Initiate Bitcoin Hash Time Lock Contract (HTLC) from opcode script.

Parameters opcode (str) – Bitcoin opcode script.

Returns HTLC – Bitcoin Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> opcode: str = "OP_IF OP_HASH256_
↳ 821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e0158 OP_
↳ EQUALVERIFY OP_DUP OP_HASH160 0a0a6590e6ba4b48118d21b86812615219ece76b OP_
↳ EQUALVERIFY OP_CHECKSIG OP_ELSE 0ec4d660 OP_CHECKLOCKTIMEVERIFY OP_DROP OP_
↳ DUP OP_HASH160 e00ff2a640b7ce2d336860739169487a57f84b15 OP_EQUALVERIFY OP_
↳ CHECKSIG OP_ENDIF"
>>> htlc.from_opcode(opcode=opcode)
<swap.providers.bitcoin.htlc.HTLC object at 0x0409DAF0>
```

from_bytecode(bytecode: str) → *swap.providers.bitcoin.htlc.HTLC*

Initialize Bitcoin Hash Time Lock Contract (HTLC) from bytecode.

Parameters bytecode (str) – Bitcoin bytecode.

Returns HTLC – Bitcoin Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4f
↳ "
>>> htlc.from_bytecode(bytecode=bytecode)
<swap.providers.bitcoin.htlc.HTLC object at 0x0409DAF0>
```

bytecode() → str

Get Bitcoin Hash Time Lock Contract (HTLC) bytecode.

Returns str – Bitcoin HTLC bytecode.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a",
↳ 1624687630)
>>> htlc.bytecode()

↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4f
↳ "
```

(continues on next page)

(continued from previous page)

opcode() → str

Get Bitcoin Hash Time Lock Contract (HTLC) OP_Code.

Returns str – Bitcoin HTLC opcode.

```

>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
    ↪ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a",
    ↪ 1624687630)
>>> htlc.opcode()
"OP_IF OP_HASH256
    ↪ 821124b554d13f247b1e5d10b84e44fb1296f18f38bba1bea34a12c843e0158 OP_
    ↪ EQUALVERIFY OP_DUP OP_HASH160 0a0a6590e6ba4b48118d21b86812615219ece76b OP_
    ↪ EQUALVERIFY OP_CHECKSIG OP_ELSE 0ec4d660 OP_CHECKLOCKTIMEVERIFY OP_DROP OP_
    ↪ DUP OP_HASH160 e00ff2a640b7ce2d336860739169487a57f84b15 OP_EQUALVERIFY OP_
    ↪ CHECKSIG OP_ENDIF"

```

hash() → str

Get Bitcoin HTLC hash.

Returns str – Bitcoin Hash Time Lock Contract (HTLC) hash.

```

>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
    ↪ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a",
    ↪ 1624687630)
>>> htlc.hash()
"a914c8c77a9b43ee2bdf1a07c48699833d7668bf264c87"

```

contract_address() → str

Get Bitcoin Hash Time Lock Contract (HTLC) address.

Returns str – Bitcoin HTLC address.

```

>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
    ↪ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a",
    ↪ 1624687630)
>>> htlc.contract_address()
"2NBYr6gvh4ujsRwKKjDrrRr2vGonazzX6Z6"

```

balance(unit: str = 'Satoshi') → Union[int, float]

Get Bitcoin HTLC balance.

Parameters unit (str) – Bitcoin unit, default to Satoshi.**Returns** int, float – Bitcoin wallet balance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a",
↳ 1624687630)
>>> htlc.balance(unit="BTC")
0.001
```

utxos(*limit: int = 15*) → list

Get Bitcoin HTLC unspent transaction output (UTXO's).

Parameters **limit** (*int*) – Limit of UTXO's, default is 15.

Returns list – Bitcoin unspent transaction outputs.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(sha256("Hello Meheret!"),
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", "n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a",
↳ 1624687630)
>>> htlc.utxos()
[{'index': 0, 'hash':
↳ 'a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31', 'output_
↳ index': 0, 'amount': 1000000, 'script':
↳ 'a914c8c77a9b43ee2bdf1a07c48699833d7668bf264c87'}]
```

6.3 Transaction

Bitcoin transaction in blockchain network.

class swap.providers.bitcoin.transaction.**Transaction**(*network: str = 'mainnet', version: int = 2*)
Bitcoin Transaction.

Parameters

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **version** (*int*) – Bitcoin transaction version, defaults to 2.

Returns Transaction – Bitcoin transaction instance.

Note: Bitcoin has only two networks, mainnet and testnet.

fee(*unit: str = 'Satoshi'*) → Union[int, float]

Get Bitcoin transaction fee.

Parameters **unit** (*str*) – Bitcoin unit, default to Satoshi.

Returns int, float – Bitcoin transaction fee.

```
>>> from swap.providers.bitcoin.transaction import WithdrawTransaction
>>> withdraw_transaction = WithdrawTransaction("testnet")
```

(continues on next page)

(continued from previous page)

```
>>> withdraw_transaction.build_transaction("mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V",
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> withdraw_transaction.fee(unit="Satoshi")
576
```

hash() → str

Get Bitcoin transaction hash.

Returns str – Bitcoin transaction id/hash.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.providers.bitcoin.transaction import FundTransaction
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", sender_address=
↳ "n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a", endtime=1624687630)
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a", htlc=htlc, amount=0.001, unit="BTC")
>>> fund_transaction.hash()
"9cc0524fb8e7b2c5fecae4eb91d43a3dc5cc18e9906abcb35a5732ff52efcc7"
```

json() → dict

Get Bitcoin transaction json format.

Returns dict – Bitcoin transaction json format.

```
>>> from swap.providers.bitcoin.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="testnet")
>>> refund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcNfAtJmes8YhpvtDzdNhDY5a", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> refund_transaction.json()
{"hex":
↳ "020000000012c392217483906f902e73c4bc132864de58153772d79268960998162266634be0100000000fffff",
↳ ", "txid": "9cc0524fb8e7b2c5fecae4eb91d43a3dc5cc18e9906abcb35a5732ff52efcc7",
↳ "hash": "9cc0524fb8e7b2c5fecae4eb91d43a3dc5cc18e9906abcb35a5732ff52efcc7",
↳ "size": 117, "vsize": 117, "version": 2, "locktime": 0, "vin": [{"txid":
↳ "be346626628199608926792d775381e54d8632c14b3ce702f90639481722392c", "vout": 1,
↳ "scriptSig": {"asm": "", "hex": ""}, "sequence": "4294967295"}], "vout": [{
↳ "value": "0.00001000", "n": 0, "scriptPubKey": {"asm": "OP_HASH160",
↳ "971894c58d85981c16c2059d422bcde0b156d044 OP_EQUAL", "hex":
↳ "a914971894c58d85981c16c2059d422bcde0b156d04487", "type": "p2sh", "address":
↳ "2N729UBGZB3xjsGFRgKivy4bSjkaJGMVSpB"}}, {"value": "0.00010662", "n": 1,
↳ "scriptPubKey": {"asm": "OP_DUP OP_HASH160",
↳ "6bce65e58a50b97989930e9a4ff1ac1a77515ef1 OP_EQUALVERIFY OP_CHECKSIG", "hex":
↳ "76a9146bce65e58a50b97989930e9a4ff1ac1a77515ef188ac", "type": "p2pkh",
↳ "address": "mqLyrNDjpENRMZAoDpspH7kR9RtgvhWzYE"}]}}
```

raw() → str

Get Bitcoin main transaction raw.

Returns str – Bitcoin transaction raw.

```

>>> from swap.providers.bitcoin.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction("testnet")
>>> withdraw_transaction.build_transaction(address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> withdraw_transaction.raw()

↳ "020000000012c392217483906f902e73c4bc132864de58153772d79268960998162266634be0100000000fffff"
↳ "

```

`type()` → str

Get Bitcoin signature transaction type.

Returns str – Bitcoin signature transaction type.

```

>>> from swap.providers.bitcoin.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "testnet")
>>> withdraw_transaction.build_transaction(address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> withdraw_transaction.type()
"bitcoin_withdraw_unsigned"

```

6.3.1 NormalTransaction

`class swap.providers.bitcoin.transaction.NormalTransaction(network: str = 'mainnet', version: int = 2)`

Bitcoin Normal transaction.

Parameters

- **network** (str) – Bitcoin network, defaults to mainnet.
- **version** (int) – Bitcoin transaction version, defaults to 2.

Returns NormalTransaction – Bitcoin normal transaction instance.

Warning: Do not forget to build transaction after initialize normal transaction.

`build_transaction(address: str, recipients: dict, unit: str = 'Satoshi', locktime: int = 0) → swap.providers.bitcoin.transaction.NormalTransaction`

Build Bitcoin normal transaction.

Parameters

- **address** (str) – Bitcoin sender address.
- **recipients** (dict) – Recipients Bitcoin address and amount.
- **unit** (str) – Bitcoin unit, default to Satoshi.
- **locktime** (int) – Bitcoin transaction lock time, defaults to 0.

Returns NormalTransaction – Bitcoin normal transaction instance.

```
>>> from swap.providers.bitcoin.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC", recipients={
↳ "2N6kHwQy6Ph5EdKNgzGrcW2WhGHKGfmP5ae": 10000000}, locktime=0)
<swap.providers.bitcoin.transaction.NormalTransaction object at 0x0409DAF0>
```

sign(*solver*: swap.providers.bitcoin.solver.NormalSolver) →
swap.providers.bitcoin.transaction.NormalTransaction
 Sign Bitcoin normal transaction.

Parameters *solver* (bitcoin.solver.NormalSolver) – Bitcoin normal solver.

Returns NormalTransaction – Bitcoin normal transaction instance.

```
>>> from swap.providers.bitcoin.transaction import NormalTransaction
>>> from swap.providers.bitcoin.solver import NormalSolver
>>> from swap.providers.bitcoin.wallet import Wallet, DEFAULT_PATH
>>> sender_wallet: Wallet = Wallet(network="testnet").from_entropy(entropy=
↳ "72fee73846f2d1a5807dc8c953bf79f1").from_path(path=DEFAULT_PATH)
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_wallet.root_
↳ xprivate_key())
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet
↳ ").build_transaction(address=sender_wallet.address(), recipients={
↳ "2N6kHwQy6Ph5EdKNgzGrcW2WhGHKGfmP5ae": 10000000})
>>> normal_transaction.sign(solver=normal_solver)
<swap.providers.bitcoin.transaction.NormalTransaction object at 0x0409DAF0>
```

transaction_raw() → str
 Get Bitcoin normal transaction raw.

Returns str – Bitcoin normal transaction raw.

```
>>> from swap.providers.bitcoin.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC", recipients={
↳ "2N6kHwQy6Ph5EdKNgzGrcW2WhGHKGfmP5ae": 10000000})
>>> normal_transaction.transaction_raw()

↳ "eyJmZWUiOiA2NzgsICJyYXciOiAiMDIwMDAwMDAwMTJjMzkyMjE3NDgzOTA2ZjkwMmU3M2M0YmMxMzI4NjRkZTU4MTUz"
↳ "
```

6.3.2 FundTransaction

class swap.providers.bitcoin.transaction.FundTransaction(*network*: str = 'mainnet', *version*: int = 2)
 Bitcoin Fund transaction.

Parameters

- **network** (str) – Bitcoin network, defaults to mainnet.
- **version** (int) – Bitcoin transaction version, defaults to 2.

Returns FundTransaction – Bitcoin fund transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(*address: str, htlc: swap.providers.bitcoin.htlc.HTLC, amount: Optional[Union[int, float]], unit: str = 'Satoshi', locktime: int = 0*) → *swap.providers.bitcoin.transaction.FundTransaction*

Build Bitcoin fund transaction.

Parameters

- **address** (*str*) – Bitcoin sender address.
- **htlc** (*bitcoin.htlc.HTLC*) – Bitcoin HTLC instance.
- **amount** (*int, float*) – Bitcoin amount, default to None.
- **unit** (*str*) – Bitcoin unit, default to Satoshi.
- **locktime** (*int*) – Bitcoin transaction lock time, defaults to 0.

Returns *FundTransaction* – Bitcoin fund transaction instance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.providers.bitcoin.transaction import FundTransaction
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", sender_address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", endtime=1624687630)
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", htlc=htlc, amount=0.001, unit="BTC")
<swap.providers.bitcoin.transaction.FundTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.bitcoin.solver.FundSolver*) → *swap.providers.bitcoin.transaction.FundTransaction*

Sign Bitcoin fund transaction.

Parameters **solver** (*bitcoin.solver.FundSolver*) – Bitcoin fund solver.

Returns *FundTransaction* – Bitcoin fund transaction instance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.providers.bitcoin.transaction import FundTransaction
>>> from swap.providers.bitcoin.solver import FundSolver
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", sender_address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", endtime=1624687630)
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", htlc=htlc, amount=0.001, unit="BTC")
>>> fund_solver = FundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv
↳ ")
>>> fund_transaction.sign(solver=fund_solver)
<swap.providers.bitcoin.transaction.FundTransaction object at 0x0409DAF0>
```


sign(*solver*: swap.providers.bitcoin.solver.WithdrawSolver) →
swap.providers.bitcoin.transaction.WithdrawTransaction
 Sign Bitcoin withdraw transaction.

Parameters **solver** (*bitcoin.solver.WithdrawSolver*) – Bitcoin withdraw solver.

Returns *WithdrawTransaction* – Bitcoin withdraw transaction instance.

```
>>> from swap.providers.bitcoin.transaction import WithdrawTransaction
>>> from swap.providers.bitcoin.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction("testnet")
>>> withdraw_transaction.build_transaction(address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4f
↳ "
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPf949JcuVFLXPJ5m4VKe33gVX3FYVZYVHr2dChU8K66aEQcPdHpUgAcq5GQu81Z4e3QN1vxCrV4p
↳ ", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_transaction.sign(solver=withdraw_solver)
<swap.providers.bitcoin.transaction.WithdrawTransaction object at 0x0409DAF0>
```

transaction_raw() → str
 Get Bitcoin withdraw transaction raw.

Returns str – Bitcoin withdraw transaction raw.

```
>>> from swap.providers.bitcoin.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction("testnet")
>>> withdraw_transaction.build_transaction(address=
↳ "mgS3WMHp9nvdUPeDJxr5iCF2P5HuFZSR3V", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> withdraw_transaction.transaction_raw()

↳ "eyJmZWUiOiA1NzYsICJyYXciOiAiMDIwMDAwMDAwMTMxZmI3NmEwYzM4ZDU3MzgxeYjMxMTBlNGY1ZWU5YjUyODFkY2Yy
↳ "
```

6.3.4 RefundTransaction

class swap.providers.bitcoin.transaction.**RefundTransaction**(*network*: str = 'mainnet', *version*: int = 2)

Bitcoin Refund transaction.

Parameters

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **version** (*int*) – Bitcoin transaction version, defaults to 2.

Returns *RefundTransaction* – Bitcoin refund transaction instance.

Warning: Do not forget to build transaction after initialize refund transaction.

build_transaction(*address: str, transaction_hash: str, locktime: int = 0*) →
swap.providers.bitcoin.transaction.RefundTransaction
 Build Bitcoin refund transaction.

Parameters

- **address** (*str*) – Bitcoin sender address.
- **transaction_hash** (*str*) – Bitcoin funded transaction hash/id.
- **locktime** (*int*) – Bitcoin transaction lock time, defaults to 0.

Returns *RefundTransaction* – Bitcoin refund transaction instance.

```
>>> from swap.providers.bitcoin.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction("testnet")
>>> refund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
<swap.providers.bitcoin.transaction.RefundTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.bitcoin.solver.RefundSolver*) →
swap.providers.bitcoin.transaction.RefundTransaction
 Sign Bitcoin refund transaction.

Parameters **solver** (*bitcoin.solver.RefundSolver*) – Bitcoin refund solver.

Returns *RefundTransaction* – Bitcoin refund transaction instance.

```
>>> from swap.providers.bitcoin.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction("testnet")
>>> refund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bba1bea34a12c843e01588876a9140a0a6590e6ba4f"
↳ ""
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTV"
↳ "", bytecode=bytecode, endtime=1624687630)
>>> refund_transaction.sign(solver=refund_solver)
<swap.providers.bitcoin.transaction.RefundTransaction object at 0x0409DAF0>
```

transaction_raw() → *str*
 Get Bitcoin refund transaction raw.

Returns *str* – Bitcoin refund transaction raw.

```
>>> from swap.providers.bitcoin.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction("testnet")
>>> refund_transaction.build_transaction(address=
↳ "n1wgm6kkzMcnfAtJmes8YhpvtDzdNhDY5a", transaction_hash=
↳ "a211d21110756b266925fee2fbf2dc81529beef5e410311b38578dc3a076fb31")
>>> refund_transaction.transaction_raw()

↳ "eyJmZWUiOiA1NzYsICJyYXciOiAiMDIwMDAwMDAwMTMxZmI3NmEwYzY4ZDU3MzgxYjMxMTBlNGY1ZWU5YjUyODFkY2Y"
↳ ""
```

6.4 Solver

Bitcoin solver.

6.4.1 NormalSolver

```
class swap.providers.bitcoin.solver.NormalSolver(xprivate_key: str, account: int = 0, change: bool = False, address: int = 0, path: Optional[str] = None, strict: bool = True)
```

Bitcoin Normal solver.

Parameters

- **xprivate_key** (*str*) – Bitcoin sender root xprivate key.
- **account** (*int*) – Bitcoin derivation account, defaults to 0.
- **change** (*bool*) – Bitcoin derivation change, defaults to False.
- **address** (*int*) – Bitcoin derivation address, defaults to 0.
- **path** (*str*) – Bitcoin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns NormalSolver – Bitcoin normal solver instance.

```
>>> from swap.providers.bitcoin.solver import NormalSolver
>>> sender_xprivate_key =
↳ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB
↳ "
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_xprivate_key)
<swap.providers.bitcoin.solver.NormalSolver object at 0x03FCCA60>
```

6.4.2 FundSolver

```
class swap.providers.bitcoin.solver.FundSolver(xprivate_key: str, account: int = 0, change: bool = False, address: int = 0, path: Optional[str] = None, strict: bool = True)
```

Bitcoin Fund solver.

Parameters

- **xprivate_key** (*str*) – Bitcoin sender root xprivate key.
- **account** (*int*) – Bitcoin derivation account, defaults to 0.
- **change** (*bool*) – Bitcoin derivation change, defaults to False.
- **address** (*int*) – Bitcoin derivation address, defaults to 0.
- **path** (*str*) – Bitcoin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns FundSolver – Bitcoin fund solver instance.

```
>>> from swap.providers.bitcoin.solver import FundSolver
>>> sender_xprivate_key: str =
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTvgVQvI
↳ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=sender_xprivate_key, path="m/
↳ 44'/1'/0'/0'/0")
<swap.providers.bitcoin.solver.FundSolver object at 0x03FCCA60>
```

6.4.3 WithdrawSolver

```
class swap.providers.bitcoin.solver.WithdrawSolver(xprivate_key: str, secret_key: str, bytecode: str,
                                                    account: int = 0, change: bool = False, address:
                                                    int = 0, path: Optional[str] = None, strict: bool =
                                                    True)
```

Bitcoin Withdraw solver.

Parameters

- **xprivate_key** (*str*) – Bitcoin recipient root xprivate key.
- **secret_key** (*str*) – Secret password/passphrase.
- **bytecode** (*str*) – Bitcoin witness HTLC bytecode.
- **account** (*int*) – Bitcoin derivation account, defaults to 0.
- **change** (*bool*) – Bitcoin derivation change, defaults to False.
- **address** (*int*) – Bitcoin derivation address, defaults to 0.
- **path** (*str*) – Bitcoin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns WithdrawSolver – Bitcoin withdraw solver instance.

```
>>> from swap.providers.bitcoin.solver import WithdrawSolver
>>> recipient_xprivate_key: str =
↳ "tprv8ZgxMBicQKsPf949JcuVFLXPJ5m4VKe33gVX3FYVZYVHr2dChU8K66aEQcPdHpUgACq5GQu81Z4e3QN1vxCrV4pxcUcX
↳ "
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140e259e08f2ec9fc99a
↳ "
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=recipient_
↳ xprivate_key, secret_key="Hello Meheret!", bytecode=bytecode, path="m/44'/1'/0'/0/
↳ 0")
<swap.providers.bitcoin.solver.WithdrawSolver object at 0x03FCCA60>
```

6.4.4 RefundSolver

class `swap.providers.bitcoin.solver.RefundSolver`(*xprivate_key*: *str*, *bytecode*: *str*, *endtime*: *int*, *account*: *int* = 0, *change*: *bool* = False, *address*: *int* = 0, *path*: *Optional[str]* = None, *strict*: *bool* = True)

Bitcoin Refund solver.

Parameters

- **xprivate_key** (*str*) – Bitcoin sender root xprivate key.
- **bytecode** (*str*) – Bitcoin witness HTLC bytecode..
- **endtime** (*int*) – Bitcoin witness expiration block timestamp.
- **account** (*int*) – Bitcoin derivation account, defaults to 0.
- **change** (*bool*) – Bitcoin derivation change, defaults to False.
- **address** (*int*) – Bitcoin derivation address, defaults to 0.
- **path** (*str*) – Bitcoin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns RefundSolver – Bitcoin refund solver instance.

```
>>> from swap.providers.bitcoin.solver import RefundSolver
>>> sender_xprivate_key: str =
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTvgVQvI
↳ "
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140e259e08f2ec9fc99a
↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=sender_xprivate_key,
↳ bytecode=bytecode, endtime=1000, path="m/44'/1'/0'/0/0")
<swap.providers.bitcoin.solver.RefundSolver object at 0x03FCCA60>
```

6.5 Signature

Bitcoin signature.

class `swap.providers.bitcoin.signature.Signature`(*network*: *str* = 'mainnet', *version*: *int* = 2)

Bitcoin Signature.

Parameters

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **version** (*int*) – Bitcoin transaction version, defaults to 2.

Returns Signature – Bitcoin signature instance.

Note: Bitcoin has only two networks, mainnet and testnet.

fee(*unit*: *str* = 'Satoshi') → Union[int, float]

Get Bitcoin transaction fee.

Parameters **unit** (*str*) – Bitcoin unit, default to Satoshi.

(continued from previous page)

```

{"hex":
  ↳ "020000000010825e00ba596ab11126cd89203b882bce60a7db019e51217056c471f510cfd850000000000006b48304500",
  ↳ ", "txid": "29c7ac0ec049687e1b952cefdaf2f1f52957e6f42f35826af21ec6bd3edf60ce",
  ↳ "hash": "29c7ac0ec049687e1b952cefdaf2f1f52957e6f42f35826af21ec6bd3edf60ce",
  ↳ "size": 224, "vsize": 224, "version": 2, "locktime": 0, "vin": [{"txid":
  ↳ "85fd0c511f476c051712e519b07d0ae6bc82b80392d86c1211ab96a50be02508", "vout": 0,
  ↳ "scriptSig": {"asm":
  ↳ "30450221009ac6afb68728eee53050ea7a301b6fb836e13b782cd52c29be2f8b0cc71f4427022069671a0a3df14
  ↳ 02065e8cb5fa76699079860a450bddd0e37e0ad3dbf2ddfd01d7b600231e6cde8e", "hex":
  ↳ "4830450221009ac6afb68728eee53050ea7a301b6fb836e13b782cd52c29be2f8b0cc71f4427022069671a0a3df
  ↳ "}, "sequence": "4294967295"}], "vout": [{"value": "0.00010000", "n": 0,
  ↳ "scriptPubKey": {"asm": "OP_HASH160 4695127b1d17c454f4bae9c41cb8e3cdb5e89d24
  ↳ OP_EQUAL", "hex": "a9144695127b1d17c454f4bae9c41cb8e3cdb5e89d2487", "type":
  ↳ "p2sh", "address": "2MygRsRs6En1RCj8a88FfsK1QBeissBTswL"}}, {"value": "0.
  ↳ 00089322", "n": 1, "scriptPubKey": {"asm": "OP_DUP OP_HASH160
  ↳ 33ecab3d67f0e2bde43e52f41ec1ecbdc73f11f8 OP_EQUALVERIFY OP_CHECKSIG", "hex":
  ↳ "76a91433ecab3d67f0e2bde43e52f41ec1ecbdc73f11f888ac", "type": "p2pkh",
  ↳ "address": "mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC"}]}]}

```

raw() → str

Get Bitcoin main transaction raw.

Returns str – Bitcoin signature transaction raw.

```

>>> from swap.providers.bitcoin.signature import Signature
>>> from swap.providers.bitcoin.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
  ↳ "eyJmZWUiOiAiNzYsICJyYXciOiAiMDIwMDAwMDAwMTMxZmI3NmEwYzM4ZDU3MzgxYjMxMTBlNGY1ZWU5YjUyODFkY2Yy
  ↳ "
>>> bytecode: str =
  ↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4
  ↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
  ↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTV
  ↳ ", bytecode=bytecode, endtime=1624687630)
>>> signature: Signature = Signature(network="testnet")
>>> signature.sign(transaction_raw=unsigned_refund_transaction_raw,
  ↳ solver=refund_solver)
>>> signature.raw()

  ↳ "020000000011823f39a8c5f6f27845dd13a65e03fe2ef5108d235e7a36edb6eb267b0459c5a00000000ca4730440
  ↳ "

```

type() → str

Get Bitcoin signature transaction type.

Returns str – Bitcoin signature transaction type.

```

>>> from swap.providers.bitcoin.signature import Signature
>>> from swap.providers.bitcoin.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
  ↳ "eyJmZWUiOiAiXMTIyLCAicmF3IjogIjAyMDAwMDAwMDIzMWZiNzZhMGMzOGQ1NzM4MWIzMTEwZTRmNWVlOWI1MjgxZGZ
  ↳ "

```

(continues on next page)

(continued from previous page)

```

>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv")
↳ ")
>>> signature: Signature = Signature(network="testnet")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.type()
"bitcoin_fund_signed"

```

sign(*transaction_raw*: str, *solver*: Union[swap.providers.bitcoin.solver.NormalSolver, swap.providers.bitcoin.solver.FundSolver, swap.providers.bitcoin.solver.WithdrawSolver, swap.providers.bitcoin.solver.RefundSolver]) → Union[swap.providers.bitcoin.signature.NormalSignature, swap.providers.bitcoin.signature.FundSignature, swap.providers.bitcoin.signature.WithdrawSignature, swap.providers.bitcoin.signature.RefundSignature]

Sign unsigned transaction raw.

Parameters

- **transaction_raw** (str) – Bitcoin unsigned transaction raw.
- **solver** (bitcoin.solver.NormalSolver, bitcoin.solver.FundSolver, bitcoin.solver.WithdrawSolver, bitcoin.solver.RefundSolver) – Bitcoin solver

Returns NormalSignature, FundSignature, WithdrawSignature, RefundSignature – Bitcoin signature instance.

```

>>> from swap.providers.bitcoin.signature import Signature
>>> from swap.providers.bitcoin.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiAxMTIyLCAicmF3IjogIjAyMDAwMDAwMDIzMWZiNzZhMGMzOGQ1NzM4MWIzMTEwZTRmNWVlOWI1MjgxZGNa"
↳ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv")
↳ ")
>>> signature: Signature = Signature(network="testnet")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
<swap.providers.bitcoin.signature.FundSignature object at 0x0409DAF0>

```

transaction_raw() → str

Get Bitcoin transaction raw.

Returns str – Bitcoin transaction raw.

```

>>> from swap.providers.bitcoin.signature import Signature
>>> from swap.providers.bitcoin.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiAxMTIyLCAicmF3IjogIjAyMDAwMDAwMDIzMWZiNzZhMGMzOGQ1NzM4MWIzMTEwZTRmNWVlOWI1MjgxZGNa"
↳ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv")
↳ ")
>>> signature: Signature = Signature(network="testnet")

```

(continues on next page)

(continued from previous page)

```

>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.transaction_raw()

↳ "eyJyYXciOiAiMDIwMDAwMDAwMTA4MjVlMDBiYTU5NmFiMTExMjZjZDg5MjAzYjg4MmJjZTYwYTdkYjAxOWU1MTIxNzA"
↳ ""

```

6.5.1 NormalSignature

class swap.providers.bitcoin.signature.**NormalSignature**(*network: str = 'mainnet', version: int = 2*)

Bitcoin Normal signature.

Parameters

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **version** (*int*) – Bitcoin transaction version, defaults to 2.

Returns NormalSignature – Bitcoin normal signature instance.

sign(*transaction_raw: str, solver: swap.providers.bitcoin.solver.NormalSolver*) →

swap.providers.bitcoin.signature.NormalSignature

Sign unsigned normal transaction raw.

Parameters

- **transaction_raw** (*str*) – Bitcoin unsigned normal transaction raw.
- **solver** (*bitcoin.solver.NormalSolver*) – Bitcoin normal solver.

Returns NormalSignature – Bitcoin normal signature instance.

```

>>> from swap.providers.bitcoin.signature import NormalSignature
>>> from swap.providers.bitcoin.solver import NormalSolver
>>> unsigned_normal_transaction_raw =
↳ "eyJmZWUiOiA2NzgsICJyYXciOiAiMDIwMDAwMDAwMTA4MjVlMDBiYTU5NmFiMTExMjZjZDg5MjAzYjg4MmJjZTYwYTdkYjAxOWU1MTIxNzA"
↳ ""
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv"
↳ ")
>>> normal_signature: NormalSignature = NormalSignature(network="testnet")
>>> normal_signature.sign(transaction_raw=unsigned_normal_transaction_raw,
↳ solver=normal_solver)
<swap.providers.bitcoin.signature.NormalSignature object at 0x0409DAF0>

```

6.5.2 FundSignature

class swap.providers.bitcoin.signature.**FundSignature**(*network: str = 'mainnet', version: int = 2*)

Bitcoin Fund signature.

Parameters

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **version** (*int*) – Bitcoin transaction version, defaults to 2.

Returns FundSignature – Bitcoin fund signature instance.

sign(transaction_raw: str, solver: swap.providers.bitcoin.solver.FundSolver) →
 swap.providers.bitcoin.signature.FundSignature
 Sign unsigned fund transaction raw.

Parameters

- **transaction_raw** (str) – Bitcoin unsigned fund transaction raw.
- **solver** (bitcoin.solver.FundSolver) – Bitcoin fund solver.

Returns FundSignature – Bitcoin fund signature instance.

```
>>> from swap.providers.bitcoin.signature import FundSignature
>>> from swap.providers.bitcoin.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiAxMTIyLCAicmF3IjogIjAyMDAwMDAwMDIzMWZiNzZhMGMzOGQ1NzM4MWIzMTEwZTRmNWVlOWI1MjgxZGZm"
↳ ""
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv"
↳ ")
>>> fund_signature: FundSignature = FundSignature(network="testnet")
>>> fund_signature.sign(transaction_raw=unsigned_fund_transaction_raw,
↳ solver=fund_solver)
<swap.providers.bitcoin.signature.FundSignature object at 0x0409DAF0>
```

6.5.3 WithdrawSignature

class swap.providers.bitcoin.signature.WithdrawSignature(network: str = 'mainnet', version: int = 2)
 Bitcoin Withdraw signature.

Parameters

- **network** (str) – Bitcoin network, defaults to mainnet.
- **version** (int) – Bitcoin transaction version, defaults to 2.

Returns WithdrawSignature – Bitcoin withdraw signature instance.

sign(transaction_raw: str, solver: swap.providers.bitcoin.solver.WithdrawSolver) →
 swap.providers.bitcoin.signature.WithdrawSignature
 Sign unsigned withdraw transaction raw.

Parameters

- **transaction_raw** (str) – Bitcoin unsigned withdraw transaction raw.
- **solver** (bitcoin.solver.WithdrawSolver) – Bitcoin withdraw solver.

Returns WithdrawSignature – Bitcoin withdraw signature instance.

```
>>> from swap.providers.bitcoin.signature import WithdrawSignature
>>> from swap.providers.bitcoin.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiAxNzYsICJyYXciOiAiMDIwMDAwMDAwMTMxZmI3NmEwYzYzM4ZDU3MzgxeYjMxMTBlNGY1ZWU5YjUyODFkY2Y"
↳ ""
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4"
↳ ""
```

(continues on next page)

(continued from previous page)

```

>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPf949JcuVFLXPJ5m4VKe33gVX3FYVZYVHr2dChU8K66aEQcPdHpUgACq5GQu81Z4e3QN1vxCrV4p
↳ ", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_signature: WithdrawSignature = WithdrawSignature(network="testnet")
>>> withdraw_signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
<swap.providers.bitcoin.signature.WithdrawSignature object at 0x0409DAF0>

```

6.5.4 RefundSignature

class swap.providers.bitcoin.signature.**RefundSignature**(network: str = 'mainnet', version: int = 2)
Bitcoin Refund signature.

Parameters

- **network** (str) – Bitcoin network, defaults to mainnet.
- **version** (int) – Bitcoin transaction version, defaults to 2.

Returns RefundSignature – Bitcoin refund signature instance.

sign(transaction_raw: str, solver: swap.providers.bitcoin.solver.RefundSolver) →
swap.providers.bitcoin.signature.RefundSignature
Sign unsigned refund transaction raw.

Parameters

- **transaction_raw** (str) – Bitcoin unsigned refund transaction raw.
- **solver** (bitcoin.solver.RefundSolver) – Bitcoin refund solver.

Returns RefundSignature – Bitcoin refund signature instance.

```

>>> from swap.providers.bitcoin.signature import Signature
>>> from swap.providers.bitcoin.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
↳ "eyJmZWUiOiA1NzYsICJyYXciOiAiMDIwMDAwMDAwMTMxZmI3NmEwYzY4ZDU3Mzg5YjMxMTB1NGY1ZWU5YjUyODFkY2Y
↳ "
>>> bytecode: str =
↳ "63aa20821124b554d13f247b1e5d10b84e44fb1296f18f38bbaa1bea34a12c843e01588876a9140a0a6590e6ba4
↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "tprv8ZgxMBicQKsPeMHMJAc6uWGYiGqi1MVM2ybmzXL2TAoDpQe85uyDpdT7mv7Nhdu5rTCBEKLZsd9KyP2LQZJzZTv
↳ ", bytecode=bytecode, endtime=1624687630)
>>> refund_signature: RefundSignature = RefundSignature(network="testnet")
>>> refund_signature.sign(transaction_raw=unsigned_refund_transaction_raw,
↳ solver=refund_solver)
<swap.providers.bitcoin.signature.RefundSignature object at 0x0409DAF0>

```

6.6 Remote Procedure Call (RPC)

Bitcoin remote procedure call.

```
swap.providers.bitcoin.rpc.get_balance(address: str, network: str = 'mainnet', headers: dict = {'accept':
                                         'application/json', 'content-type': 'application/json;
                                         charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout:
                                         int = 60) → int
```

Get Bitcoin balance.

Parameters

- **address** (*str*) – Bitcoin address.
- **network** (*str*) – Bitcoin network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common-headers`.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *int* – Bitcoin balance (Satoshi amount).

```
>>> from swap.providers.bitcoin.rpc import get_balance
>>> get_balance(address="n1wgm6kkzMcnFAtJmes8YhpvtDzdNhDY5a", network="testnet")
1394238
```

```
swap.providers.bitcoin.rpc.get_utxos(address: str, network: str = 'mainnet', include_script: bool = True,
                                     limit: int = 15, headers: dict = {'accept': 'application/json',
                                     'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap
                                     User-Agent 0.5.0'}, timeout: int = 60) → list
```

Get Bitcoin unspent transaction outputs (UTXO's).

Parameters

- **address** (*str*) – Bitcoin address.
- **network** (*str*) – Bitcoin network, defaults to `mainnet`.
- **include_script** (*bool*) – Bitcoin include script, defaults to `True`.
- **limit** (*int*) – Bitcoin utxo's limit, defaults to 15.
- **headers** (*dict*) – Request headers, default to `common-headers`.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *list* – Bitcoin unspent transaction outputs (UTXO's).

```
>>> from swap.providers.bitcoin.rpc import get_utxos
>>> get_utxos(address="mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC", network="testnet")
[{'tx_hash': '98c6a3d4e136d32d0848126e08325c94da2e8217593e92236471b11b42ee7999',
  ↳ 'block_height': 1890810, 'tx_input_n': -1, 'tx_output_n': 1, 'value': 67966, 'ref_
  ↳ balance': 146610, 'spent': False, 'confirmations': 5278, 'confirmed': '2020-11-
  ↳ 09T08:53:01Z', 'double_spend': False, 'script':
  ↳ '76a91433ecab3d67f0e2bde43e52f41ec1ecbdc73f11f888ac'}]
```

```
swap.providers.bitcoin.rpc.get_transaction(transaction_hash: str, network: str = 'mainnet', headers:
                                         dict = {'accept': 'application/json', 'content-type':
                                         'application/json; charset=utf-8', 'user-agent': 'Swap
                                         User-Agent 0.5.0'}, timeout: int = 60) → dict
```

Get Bitcoin transaction detail.

Parameters

- **transaction_hash** (*str*) – Bitcoin transaction hash/id.
- **network** (*str*) – Bitcoin network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common-headers`.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Bitcoin transaction detail.

```
>>> from swap.providers.bitcoin.rpc import get_transaction
>>> get_transaction(transaction_hash=
↳ "4e91bca76db112d3a356c17366df93e364a4922993414225f65390220730d0c1", network=
↳ "testnet")
{'block_hash': '0000000000000006fb2aec57209181feb54750319e47263c48eca24369bdbee86',
↳ 'block_height': 1890810, 'block_index': 37, 'hash':
↳ '98c6a3d4e136d32d0848126e08325c94da2e8217593e92236471b11b42ee7999', 'addresses': [
↳ '2N1NiQVBaSXmdZVATeST9sMQWVPeL5oA8Ks', 'mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC'],
↳ 'total': 77966, 'fees': 678, 'size': 224, 'preference': 'low', 'relayed_by': '104.
↳ 197.28.151:18333', 'confirmed': '2020-11-09T08:53:01Z', 'received': '2020-11-
↳ 09T08:47:10.889Z', 'ver': 2, 'double_spend': False, 'vin_sz': 1, 'vout_sz': 2,
↳ 'confirmations': 5279, 'confidence': 1, 'inputs': [{'prev_hash':
↳ '9825b68e57c8a924285828dde37869c2eca3bb3784b171353962f0d7e7577da1', 'output_index
↳ ': 1, 'script':
↳ '483045022100e906ed456dc5d2c2546e70385b028dbbe62e9abc94324e9477c1374f8355501e02201072a242ebae589
↳ ', 'output_value': 78644, 'sequence': 4294967295, 'addresses': [
↳ 'mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC'], 'script_type': 'pay-to-pubkey-hash', 'age
↳ ': 1837431}], 'outputs': [{'value': 10000, 'script':
↳ 'a914592ba3ba46263dc5c976ede5a6f91f75e5b6f69f87', 'addresses': [
↳ '2N1NiQVBaSXmdZVATeST9sMQWVPeL5oA8Ks'], 'script_type': 'pay-to-script-hash'}, {
↳ 'value': 67966, 'script': '76a91433ecab3d67f0e2bde43e52f41ec1ecbdc73f11f888ac',
↳ 'addresses': ['mkFWGt4hT11XS8dJKzzRFsTrqjjAwZfQAC'], 'script_type': 'pay-to-
↳ pubkey-hash'}]}}
```

`swap.providers.bitcoin.rpc.find_p2sh_utxo(transaction: dict) → Optional[dict]`
Find Bitcoin pay to script hash UTXO info's.

Parameters **transaction** (*dict*) – Bitcoin transaction detail.

Returns dict – Pay to Script Hash (P2SH) UTXO info's.

```
>>> from swap.providers.bitcoin.rpc import find_p2sh_utxo, get_transaction
>>> find_p2sh_utxo(transaction=get_transaction(
↳ "868f81fd172b8f1d24e0c195af011489c3a7948513521d4b6257b8b5fb2ef409", "testnet"))
{'value': 10050780, 'script': 'a9149418feed4647e156d6663db3e0cef7c050d0386787',
↳ 'addresses': ['2N6kHwQy6Ph5EdKNgzGrcW2WhGHKGfmP5ae'], 'script_type': 'pay-to-
↳ script-hash'}}
```

`swap.providers.bitcoin.rpc.decode_raw(raw: str, network: str = 'mainnet', offline: bool = True, headers: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → dict`

Decode original Bitcoin raw.

Parameters

- **raw** (*str*) – Bitcoin transaction raw.

- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **offline** (*bool*) – Offline decode, defaults to True.
- **headers** (*dict*) – Request headers, default to common-headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Bitcoin decoded transaction raw.

```
>>> from swap.providers.bitcoin.rpc import decode_raw
>>> decode_raw(raw=
→ "020000000011823f39a8c5f6f27845dd13a65e03fe2ef5108d235e7a36edb6eb267b0459c5a0100000006a473044022070
→ ", network="testnet")
{'hex':
→ '020000000011823f39a8c5f6f27845dd13a65e03fe2ef5108d235e7a36edb6eb267b0459c5a0100000006a473044022070
→ ', 'txid': '6e5c80f600f45acda3c3101128bb3075bf2cf7af4bab0d99c9d856ebfb4b0953',
→ 'hash': '6e5c80f600f45acda3c3101128bb3075bf2cf7af4bab0d99c9d856ebfb4b0953', 'size
→ ': 223, 'vsize': 223, 'version': 2, 'locktime': 0, 'vin': [{'txid':
→ '5a9c45b067b26edb6ea3e735d20851efe23fe0653ad15d84276f5f8c9af32318', 'vout': 1,
→ 'scriptSig': {'asm':
→ '304402207018b7fd1ba6624fe9bb0f16cd65fa243d202e32fdff452699f56465b61ab648022009f0dc1a0a63109246c4
→ 027f0dc0894bd690635412af782d05e4f79d3d40bf568978c650f3f1ca1a96cf36', 'hex':
→ '47304402207018b7fd1ba6624fe9bb0f16cd65fa243d202e32fdff452699f56465b61ab648022009f0dc1a0a63109246c4
→ '}, 'sequence': '4294967295']], 'vout': [{'value': '0.00010000', 'n': 0,
→ 'scriptPubKey': {'asm': 'OP_HASH160 9418feed4647e156d6663db3e0cef7c050d03867 OP_
→ EQUAL', 'hex': 'a9149418feed4647e156d6663db3e0cef7c050d0386787', 'type': 'p2sh',
→ 'address': '2N6kHwQy6Ph5EdKNgzGrcW2WhGHKGfmP5ae'}}], {'value': '0.00078644', 'n': 1,
→ 1, 'scriptPubKey': {'asm': 'OP_DUP OP_HASH160
→ 33ecab3d67f0e2bde43e52f41ec1ecbdc73f11f8 OP_EQUALVERIFY OP_CHECKSIG', 'hex':
→ '76a91433ecab3d67f0e2bde43e52f41ec1ecbdc73f11f888ac', 'type': 'p2pkh', 'address':
→ 'mkFWGt4hT11XS8dJKzRFsTrqjjAwZfQAC'}}]}
```

```
swap.providers.bitcoin.rpc.submit_raw(raw: str, network: str = 'mainnet', endpoint: str = 'sochain',
headers: dict = {'accept': 'application/json', 'content-type':
'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent
0.5.0'}, timeout: int = 60) → str
```

Submit original Bitcoin raw into blockchain.

Parameters

- **raw** (*str*) – Bitcoin transaction raw.
- **network** (*str*) – Bitcoin network, defaults to mainnet.
- **endpoint** (*str*) – Bitcoin transaction submitter endpoint api name, defaults to sochain.
- **headers** (*dict*) – Request headers, default to common-headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Bitcoin submitted transaction id/hash.

```
>>> from swap.providers.bitcoin.rpc import submit_raw
>>> submit_raw(raw=
→ "020000000011823f39a8c5f6f27845dd13a65e03fe2ef5108d235e7a36edb6eb267b0459c5a0100000006a473044022070
→ ", network="testnet")
"167faa4043ff622e7860ee5228d1ad6d763c5a6cfce79dbc3b9b5fc7bded6394"
```

6.7 Utils

Bitcoin Utils.

`swap.providers.bitcoin.utils.fee_calculator(transaction_input: int = 1, transaction_output: int = 1) → int`

Bitcoin fee calculator.

Parameters

- **transaction_input** (*int*) – transaction input numbers, defaults to 1.
- **transaction_output** (*int*) – transaction output numbers, defaults to 1.

Returns *int* – Bitcoin fee (Satoshi amount).

```
>>> from swap.providers.bitcoin.utils import fee_calculator
>>> fee_calculator(transaction_input=2, transaction_output=9)
1836
```

`swap.providers.bitcoin.utils.get_address_type(address: str) → str`

Get Bitcoin address type.

Parameters **address** (*str*) – Bitcoin address.

Returns *str* – Bitcoin address type (P2PKH, P2SH).

```
>>> from swap.providers.bitcoin.utils import get_address_type
>>> get_address_type(address="mrmtGq2HmMqAogSsGDjCtXUpXrb7rHThFH")
"p2pkh"
```

`swap.providers.bitcoin.utils.is_network(network: str) → bool`

Check Bitcoin network.

Parameters **network** (*str*) – Bitcoin network.

Returns *bool* – Bitcoin valid/invalid network.

```
>>> from swap.providers.bitcoin.utils import is_network
>>> is_network(network="testnet")
True
```

`swap.providers.bitcoin.utils.is_address(address: str, network: Optional[str] = None, address_type: Optional[str] = None) → bool`

Check Bitcoin address.

Parameters

- **address** (*str*) – Bitcoin address.
- **network** (*str*) – Bitcoin network, defaults to None.
- **address_type** (*str*) – Bitcoin address type, defaults to None.

Returns *bool* – Bitcoin valid/invalid address.

```
>>> from swap.providers.bitcoin.utils import is_address
>>> is_address(address="mrmtGq2HmMqAogSsGDjCtXUpXrb7rHThFH", network="testnet")
True
```

Check Bitcoin transaction raw.

Returns bool – Bitcoin valid/invalid transaction raw.

True

Bitcoin amount unit converter

- **amount** (*Union[int, float]*) – Bitcoin any amount.
- **unit_from** (*str*) – Bitcoin unit convert from symbol, default to Satoshi2BTC.

0.1

Decode Bitcoin transaction raw.

- **transaction_raw** (*str*) – Bitcoin transaction raw.
- **offline** (*bool*) – Offline decode, defaults to True.
- **headers** (*dict*) – Request headers, default to common-headers.
- **timeout** (*int*) – Request timeout, default to 60.

```
'scriptSig': {'asm': '', 'hex': '', 'sequence': '4294967295'}], 'vo
```

(continued from previous page)

```
swap.providers.bitcoin.utils.submit_transaction_raw(transaction_raw: str, endpoint: str = 'sochain',
                                                    headers: dict = {'accept': 'application/json',
                                                                    'content-type': 'application/json; charset=utf-8',
                                                                    'user-agent': 'Swap User-Agent 0.5.0'}, timeout:
                                                    int = 60) → dict
```

Submit transaction raw to Bitcoin blockchain.

Parameters

- **transaction_raw** (*str*) – Bitcoin transaction raw.
- **endpoint** (*str*) – Bitcoin transaction submitter endpoint api name, defaults to sochain.
- **headers** (*dict*) – Request headers, default to common-headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Bitcoin submitted transaction id, fee, type and date.

```
>>> from swap.providers.bitcoin.utils import submit_transaction_raw
>>> transaction_raw =
↪ "eyJmZWUiOiA2NzgsICJyYXciOiAiMDIwMDAwMDAwMTg4OGJlN2VjMDY1MDk3ZDk1NjY0NzYzZjI3NmQ0MjU1NTJkNmM1ZmIz"
↪ "
>>> submit_transaction_raw(transaction_raw=transaction_raw)
{'fee': '...', 'type': '...', 'transaction_id': '...', 'network': '...', 'date': '..
↪ .'}
```

```
swap.providers.bitcoin.utils.get_address_hash(address: str, script: bool = False) → Union[str,
                                                btcpy.structs.script.P2pkhScript,
                                                btcpy.structs.script.P2shScript]
```

Get Bitcoin address hash.

Parameters

- **address** (*str*) – Bitcoin address.
- **script** (*bool*) – Return script (P2pkhScript, P2shScript), default to False.

Returns str – Bitcoin address hash.

```
>>> from swap.providers.bitcoin.utils import get_address_hash
>>> get_address_hash(address="mrmtGq2HMMqAogSsGDjCtXUpXrb7rHThFH", script=False)
"7b7c4431a43b612a72f8229935c469f1f6903658"
```

Bytom is a protocol of multiple byte assets. Heterogeneous byte-assets operate in different forms on the Bytom Blockchain and atomic assets (warrants, securities, dividends, bonds, intelligence information, forecasting information and other information that exist in the physical world) can be registered, exchanged, gambled via Bytom.

For more <https://bytom.io>

7.1 Wallet

The implementation of Hierarchical Deterministic (HD) wallets generator for Bytom blockchain.

class `swap.providers.bytom.wallet.Wallet`(*network: str = 'mainnet'*)

Bytom Wallet class.

Parameters *network* (*str*) – Bytom network, defaults to mainnet.

Returns `Wallet` – Bytom wallet instance.

Note: Bytom has only two networks, `mainnet`, `solonet` and `testnet`.

from_entropy(*entropy: str, language: str = 'english', passphrase: Optional[str] = None*) →
swap.providers.bytom.wallet.Wallet

Initiate Bytom wallet from entropy.

Parameters

- **entropy** (*str*) – Bytom entropy hex string.
- **language** (*str*) – Bytom wallet language, default to `english`.
- **passphrase** (*str*) – Bytom wallet passphrase, default to `None`.

Returns `Wallet` – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_mnemonic(*mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None*) →
swap.providers.bytom.wallet.Wallet

Initialize Bytom wallet from mnemonic.

Parameters

- **mnemonic** (*str*) – Bytom mnemonic words.
- **language** (*str*) – Bytom wallet language, default to english.
- **passphrase** (*str*) – Bytom wallet passphrase, default to None.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park,
↳ clown build renew illness fault")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_seed(*seed: str*) → *swap.providers.bytom.wallet.Wallet*
Initialize Bytom wallet from seed.

Parameters **seed** (*str*) – Bytom Seed hex string.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_seed(seed=
↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ ")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_xprivate_key(*xprivate_key: str*) → *swap.providers.bytom.wallet.Wallet*
Initiate Bytom wallet from xprivate key.

Parameters **xprivate_key** (*str*) – Bytom XPrivate key.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_private_key(*private_key: str*) → *swap.providers.bytom.wallet.Wallet*
Initialize Bytom wallet from private key.

Parameters **private_key** (*str*) – Bytom Private key.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_private_key(private_key=
↳ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512
↳ ")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_path(*path: str*) → *swap.providers.bytom.wallet.Wallet*
Drive Bytom wallet from path.

Parameters **path** (*str*) – Bytom derivation path.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_indexes(*indexes: List[str]*) → *swap.providers.bytom.wallet.Wallet*

Drive Bytom wallet from indexes.

Parameters *indexes* (*list*) – Bytom derivation indexes.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> wallet.from_indexes(indexes=["2c000000", "99000000", "01000000", "00000000",
↳ "01000000"])
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

from_index(*index: int, hardened: bool = False*) → *swap.providers.bytom.wallet.Wallet*

Drive Bytom wallet from index.

Parameters

- **index** (*int*) – Bytom wallet index.
- **hardened** (*bool*) – Use hardened, default to False.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_index(index=44)
>>> wallet.from_index(index=153)
>>> wallet.from_index(index=1)
>>> wallet.from_index(index=0)
>>> wallet.from_index(index=1)
<swap.providers.bytom.wallet.Wallet object at 0x040DA268>
```

clean_derivation() → *swap.providers.bytom.wallet.Wallet*

Clean derivation Bytom wallet.

Returns Wallet – Bytom wallet instance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.indexes()
["2c000000", "99000000", "01000000", "00000000", "01000000"]
>>> wallet.path()
```

(continues on next page)

(continued from previous page)

```
"m/44/153/1/0/1"
>>> wallet.clean_derivation()
>>> wallet.indexes()
[]
>>> wallet.path()
None
```

strength() → Optional[int]
Get Bytom wallet strength.

Returns int – Bytom wallet strength.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.strength()
128
```

entropy() → Optional[str]
Get Bytom wallet entropy.

Returns str – Bytom wallet entropy.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.entropy()
"ed0802d701a033776811601dd6c5c4a9"
```

mnemonic() → Optional[str]
Get Bytom wallet mnemonic.

Returns str – Bytom wallet mnemonic.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.mnemonic()
"unfair divorce remind addict add roof park clown build renew illness fault"
```

passphrase() → Optional[str]
Get Bytom wallet passphrase.

Returns str – Bytom wallet passphrase.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9", passphrase=
↳ "meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]
Get Bytom wallet language.

Returns str – Bytom wallet language.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.language()
"english"
```

seed() → Optional[str]
Get Bytom wallet seed.

Returns str – Bytom wallet seed.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.seed()

↪ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea"
↪ "
```

path() → Optional[str]
Get Bytom wallet derivation path.

Returns str – Bytom derivation path.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet = Wallet(network="mainnet", change=True, address=3)
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.path()
"m/44/153/1/0/1"
```

indexes() → list
Get Bytom wallet derivation indexes.

Returns list – Bytom derivation indexes.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.indexes()
['2c000000', '99000000', '01000000', '00000000', '01000000']
```

xprivate_key() → Optional[str]
Get Bytom wallet xprivate key.

Returns str – Bytom xprivate key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.xprivate_key()

↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↪ "
```

xpublic_key() → Optional[str]

Get Bytom wallet xpublic key.

Returns str – Bytom xpublic key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.xpublic_key()

↪ "f80a401807fde1ee5727ae032ee144e4b757e69431e68e6cd732eda3c8cd3936daedfdd0fd8f8df14e2084c7e8d
↪ "
```

expand_xprivate_key() → Optional[str]

Get Bytom wallet expand xprivate key.

Returns str – Bytom expand xprivate key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.expand_xprivate_key()

↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e465c68d75d8a29eb3ffd7e8213808
↪ "
```

child_xprivate_key() → Optional[str]

Get Bytom child wallet xprivate key.

Returns str – Bytom child xprivate key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.child_xprivate_key()

↪ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512
↪ "
```

child_xpublic_key() → Optional[str]

Get Bytom child wallet xpublic key.

Returns str – Bytom child xpublic key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.child_xpublic_key()

↪ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212db35f71c405fd5948ecffa2c512
↪ "
```

guid() → Optional[str]

Get Bytom wallet Blockcenter GUID.

Returns str – Bytom Blockcenter GUID.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.guid()
"9ed61a9b-e7b6-4cb7-94fb-932b738e4f66"
```

private_key() → str

Get Bytom wallet private key.

Returns str – Bytom private key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.private_key()
↪ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512"
↪ "
```

public_key() → str

Get Bytom wallet public key.

Returns str – Bytom public key.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.public_key()
"fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212"
```

program() → str

Get Bytom wallet control program.

Returns str – Bytom control program.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.program()
"0014b1592acbb917f13937166c2a9b6ce973296ebb60"
```

address(network: Optional[str] = None) → str

Get Bytom wallet address.

Parameters network (str) – Bytom network, defaults to mainnet.

Returns str – Bytom wallet address.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_indexes(indexes=["2c000000", "99000000", "01000000", "00000000",
↪ "01000000"])
```

(continues on next page)

(continued from previous page)

```
>>> wallet.address(network="mainnet")
"bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx"
```

```
balance(asset: Union[str, swap.providers.bytom.assets.AssetNamespace] =
    'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA', unit: str = 'NEU') → Union[int, float]
    Get Bytom wallet balance.
```

Parameters

- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM asset.
- **unit** (*str*) – Bytom unit, default to NEU.

Returns int, float – Bytom wallet balance.

```
>>> from swap.providers.bytom.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.balance(unit="BTM")
2.0
```

```
utxos(asset: Union[str, swap.providers.bytom.assets.AssetNamespace] =
    '0000000000000000000000000000000000000000000000000000000000000000', limit: int = 15) → list
    Get Bytom wallet unspent transaction output (UTXO's).
```

Parameters

- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM asset.
- **limit** (*int*) – Limit of UTXO's, default is 15.

Returns list – Bytom unspent transaction outputs.

[illegible]

7.2 Hash Time Lock Contract (HTLC)

Bytom Hash Time Lock Contract (HTLC).

```
class swap.providers.bytom.htlc.HTLC(network: str = 'mainnet', contract_address: Optional[str] = None)
    Bytom Hash Time Lock Contract (HTLC).
```

Parameters `network (str)` – Bytom network, defaults to mainnet.

Returns HTLC – Bytom HTLC instance.

Note: Bytom has only three networks, mainnet, solonet and testnet.

build_htlc(*secret_hash: str, recipient_public_key: str, sender_public_key: str, endblock: int, use_script: bool = False*) → *swap.providers.bytom.htlc.HTLC*

Build Bytom Hash Time Lock Contract (HTLC).

Parameters

- **secret_hash** (*str*) – secret sha-256 hash.
- **recipient_public_key** (*str*) – Bytom recipient public key.
- **sender_public_key** (*str*) – Bytom sender public key.
- **endblock** (*int*) – Bytom expiration block height.
- **use_script** (*bool*) – Initialize HTLC by using script, default to False.

Returns HTLC – Bytom Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.rpc import get_current_block_height
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=get_current_block_height(plus=100), use_script=False)
<swap.providers.bytom.htlc.HTLC object at 0x0409DAF0>
```

from_bytecode(*bytecode: str*) → *swap.providers.bytom.htlc.HTLC*

Initialize Bytom Hash Time Lock Contract (HTLC) from bytecode.

Parameters **bytecode** (*str*) – Bytom bytecode.

Returns HTLC – Bytom Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> htlc.from_bytecode(bytecode=bytecode)
<swap.providers.bytom.htlc.HTLC object at 0x0409DAF0>
```

bytecode() → *str*

Get Bytom Hash Time Lock Contract (HTLC) bytecode.

Returns *str* – Bytom HTLC bytecode.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
```

(continues on next page)

(continued from previous page)

```
>>> htlc.bytecode()
↪ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↪ "
```

opcode() → Optional[str]

Get Bytom Hash Time Lock Contract (HTLC) OP_Code.

Returns str – Bytom HTLC opcode.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=679208)
>>> htlc.opcode()
"0x285d0a 0xfe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212
↪ 0x3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e
↪ 0x3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb DEPTH
↪ 0x547a6416000000557aa888537a7cae7cac631f000000537acd9f6972ae7cac FALSE
↪ CHECKPREDICATE"
```

hash() → str

Get Bytom Hash Time Lock Contract (HTLC) hash.

Returns str – Bytom HTLC hash.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=679208)
>>> htlc.hash()
"e7f4a9815f3a36c616c5666b97fb7fdacd3720c117d078c429494d1b617fe7d4"
```

contract_address() → str

Get Bytom Hash Time Lock Contract (HTLC) address.

Returns str – Bytom HTLC address.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=679208)
>>> htlc.contract_address()
"bm1qul62nq2l8gmvv9k9ve4e07mlmtxnxgxpzlg833pff9x3kctlul2q727jyy"
```

balance(asset: Union[str, swap.providers.bytom.assets.AssetNamespace] =
 'ff', unit: str = 'NEU') → Union[int, float]
 Get Bytom HTLC balance.

Parameters

- **asset** (str, bytom.assets.AssetNamespace, bytom.assets.AssetNamespace)
 – Bytom asset id, defaults to BTM.
- **unit** (str) – Bytom unit, default to NEU.

Returns int, float – Bytom HTLC balance.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
>>> htlc.balance(asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
0.1
```

utxos(asset: Union[str, swap.providers.bytom.assets.AssetNamespace] =
 'ff', limit: int = 15) → list
 Get Bytom HTLC unspent transaction output (UTXO's).

Parameters

- **asset** (str, bytom.assets.AssetNamespace) – Bytom asset id, defaults to BTM.
- **limit** (int) – Limit of UTXO's, default is 15.

Returns list – Bytom unspent transaction outputs.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
>>> htlc.utxos(asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
[{'hash': '1aaf7df33c1d41bc6108c93d8b6da6af1d7f68632f54516408a03ff86494a1f0',
↳ 'asset': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'amount': 100000000}]
```

7.3 Transaction

Bytom transaction in blockchain network.

class `swap.providers.bytom.transaction.Transaction(network: str = 'mainnet')`
Bytom Transaction.

Parameters `network` (`str`) – Bytom network, defaults to mainnet.

Returns Transaction – Bytom transaction instance.

Note: Bytom has only three networks, mainnet, solonet and mainnet.

fee(`unit: str = 'NEU'`) → Union[int, float]

Get Bytom transaction fee.

Parameters `unit` (`str`) – Bytom unit, default to NEU.

Returns int, float – Bytom transaction fee.

```
>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.fee(unit="NEU")
509000
```

hash() → str

Get Bytom transaction hash.

returns str – Bytom transaction id/hash.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_transaction.hash()
"a3078af0810c68a7bb6f2f42cd67dce9dea3d77028ca0c527224e4524038abc4"
```

json() → dict

Get Bytom transaction json format.

Returns dict – Bytom transaction json format.

```
>>> from swap.providers.bytom.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> refund_transaction.json()
{"tx_id": "1722aa930f6f93b4c87788ea55f49055f26f86821bcd11a64d42bcb9e3b8a96d",
↳ "version": 1, "size": 179, "time_range": 0, "inputs": [{"type": "spend",
↳ "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↳ "asset_definition": {}, "amount": 100000000, "control_program":
↳ "0020e7f4a9815f3a36c616c5666b97fb7fdacd3720c117d078c429494d1b617fe7d4",
↳ "address": "bm1qul62nq2l8gmvv9k9ve4e07mlmtxnwgxpzlg833pff9x3kctlul2q727jyy",
↳ "spent_output_id":
↳ "1aaf7df33c1d41bc6108c93d8b6da6af1d7f68632f54516408a03ff86494a1f0", "input_id
↳ ": "6ccb3abb96d713fcdf27548ed76dadcd695259fb7570b38ab9cde23f7ec261d60",
↳ "witness_arguments": null, "sign_data":
↳ "cc78c1fb648f8826e4dd4f85f885ac75866c0233b0af6581753d858304b8e04b"}], "outputs
↳ ": [{"type": "control", "id":
↳ "6f831e2f958252a20b8d5aa9242c7bda229cb0e35bd2101978ea7df6cd7cc728", "position
↳ ": 0, "asset_id":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", "asset_
↳ definition": {}, "amount": 9491000, "control_program":
↳ "0014b1592acbb917f13937166c2a9b6ce973296ebb60", "address":
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx"}], "fee": 509000}
```

raw() → str

Get Bytom transaction raw.

Returns str – Bytom transaction raw.

```
>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.raw()
↳ "07010001016b0169f7df4d06a3fe3c8ac6438f25f9c97744a10455357857775526c3e6c752fb69eaaffffffff"
↳ "
```

type() → str

Get Bytom signature transaction type.

Returns str – Bytom signature transaction type.

```
>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
```

(continued from previous page)

```
>>> withdraw_transaction.type()
"bytom_withdraw_unsigned"
```

unsigned_datas(*detail: bool = False*) → List[dict]

Get Bytom transaction unsigned datas(messages) with instruction.

Parameters **detail** (*bool*) – Bytom unsigned datas to see detail, defaults to False.

Returns list – Bytom transaction unsigned datas.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
    ↪ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
    ↪ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
    ↪ sender_public_key=
    ↪ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
    ↪ endblock=679208)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
    ↪ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9grufx", htlc=htlc, amount=0.1, asset=
    ↪ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
    ↪ ")
>>> fund_transaction.unsigned_datas()
[{"datas": [{"f42a2b6e15585b88da8b34237c7a6fd83af12ee6971813d66cf794a63ebcc16f"}],
  ↪ "public_key":
  ↪ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212", "network
  ↪ ": "mainnet", "path": "m/44/153/1/0/1"}]}
```

signatures() → List[List[str]]

Get Bytom transaction signatures(signed datas).

Returns list – Bytom transaction signatures.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> from swap.providers.bytom.solver import FundSolver
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
    ↪ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
    ↪ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
    ↪ sender_public_key=
    ↪ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
    ↪ endblock=679208)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
    ↪ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9grufx", htlc=htlc, amount=0.1, asset=
    ↪ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
    ↪ ")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
    ↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
    ↪ ", path="m/44/153/1/0/1")
>>> fund_transaction.sign(solver=fund_solver)
```

(continues on next page)

(continued from previous page)

```
>>> fund_transaction.signatures()
[[
  ↳ 'b82e97abc4b70f7ffe7f783254c63e61436d6a7ad15da89b1fb791f91d1d6aa0bab7ff86328eabd2959f5475dde
  ↳ ']]
```

7.3.1 NormalTransaction

class swap.providers.bytom.transaction.**NormalTransaction**(*network: str = 'mainnet'*)

Bytom Normal transaction.

Parameters **network** (*str*) – Bytom network, defaults to mainnet.

Returns NormalTransaction – Bytom normal transaction instance.

Warning: Do not forget to build transaction after initialize normal transaction.

build_transaction(*address: str, recipients: dict, asset: Union[str, swap.providers.bytom.assets.AssetNamespace] = 'ff', unit: str = 'NEU'*) → *swap.providers.bytom.transaction.NormalTransaction*

Build Bytom normal transaction.

Parameters

- **address** (*str*) – Bytom sender wallet address.
- **recipients** (*dict*) – Recipients Bytom address and amount.
- **asset** (*str, bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM.
- **unit** (*str*) – Bytom unit, default to NEU.

Returns NormalTransaction – Bytom normal transaction instance.

```
>>> from swap.providers.bytom.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="mainnet")
>>> normal_transaction.build_transaction(address=
  ↳ "bm1q9ndylx02syfwd7npehfxz4lddhzqsve2fu6vc7", recipients={
  ↳ "bm1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8": 10000000},
  ↳ asset="ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.bytom.transaction.NormalTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.bytom.solver.NormalSolver*) →

swap.providers.bytom.transaction.NormalTransaction

Sign Bytom normal transaction.

Parameters **solver** (*bytom.solver.NormalSolver*) – Bytom normal solver.

Returns NormalTransaction – Bytom normal transaction instance.

```
>>> from swap.providers.bytom.transaction import NormalTransaction
>>> from swap.providers.bytom.solver import NormalSolver
>>> from swap.providers.bytom.wallet import Wallet, DEFAULT_PATH
>>> sender_wallet: Wallet = Wallet(network="mainnet").from_entropy(entropy=
  ↳ "72fee73846f2d1a5807dc8c953bf79f1").from_path(path=DEFAULT_PATH)
```

(continues on next page)

(continued from previous page)

```

>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_wallet.
↳ xprivate_key())
>>> normal_transaction: NormalTransaction = NormalTransaction(network="mainnet")
>>> normal_transaction.build_transaction(address=sender_wallet.address(),
↳ recipients={"bm1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8":
↳ 100000000}, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> normal_transaction.sign(solver=normal_solver)
<swap.providers.bytom.transaction.NormalTransaction object at 0x0409DAF0>

```

transaction_raw() → str

Get Bytom normal transaction raw.

Returns str – Bytom normal transaction raw.

```

>>> from swap.providers.bytom.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction("mainnet")
>>> normal_transaction.build_transaction(address=
↳ "bm1q9ndylx02syfwd7npehfxz4lddhzqsve2fu6vc7", recipients={
↳ "bm1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8": 100000000},
↳ asset="ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> normal_transaction.transaction_raw()

↳ "eyJmZWUiOiA2NzgsICJyYXciOiAiMDIwMDAwMDAwMTJjMzkyMjE3NDgzOTA2ZjkwMmU3M2M0YmMxMzI4NjRkZTU4MTU"
↳ ""

```

7.3.2 FundTransaction

class swap.providers.bytom.transaction.**FundTransaction**(network: str = 'mainnet')

Bytom Fund transaction.

Parameters **network** (str) – Bytom network, defaults to mainnet.

Returns FundTransaction – Bytom fund transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(address: str, htlc: swap.providers.bytom.htlc.HTLC, amount: Union[int, float], asset: Union[str, swap.providers.bytom.assets.AssetNamespace] = 'AA', unit: str = 'NEU') → swap.providers.bytom.transaction.FundTransaction

Build Bytom fund transaction.

Parameters

- **address** (str) – Bytom sender wallet address.
- **htlc** (str) – Bytom Hash Time Lock Contract (HTLC) instance.
- **amount** (int, float) – Bytom amount to fund.
- **asset** (str, bytom.assets.AssetNamespace) – Bytom asset id, defaults to BTM.
- **unit** (str) – Bytom unit, default to NEU.

Returns FundTransaction – Bytom fund transaction instance.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
<swap.providers.bytom.transaction.FundTransaction object at 0x0409DAF0>
```

sign(solver: swap.providers.bytom.solver.FundSolver) → swap.providers.bytom.transaction.FundTransaction
Sign Bytom fund transaction.

Parameters solver (bytom.solver.FundSolver) – Bytom fund solver.

Returns FundTransaction – Bytom fund transaction instance.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> from swap.providers.bytom.solver import FundSolver
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ", path="m/44/153/1/0/1")
>>> fund_transaction.sign(solver=fund_solver)
<swap.providers.bytom.transaction.FundTransaction object at 0x0409DAF0>
```

transaction_raw() → str
Get Bytom fund transaction raw.

Returns str – Bytom fund transaction raw.

```
>>> from swap.providers.bytom.htlc import HTLC
>>> from swap.providers.bytom.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=679208)
```

(continued from previous page)

```
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9grufx", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_transaction.transaction_raw()
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcTlxcnV"
↳ "
```

7.3.3 WithdrawTransaction

```
class swap.providers.bytom.transaction.WithdrawTransaction(network: str = 'mainnet')
    Bytom Withdraw transaction.
```

Parameters **network** (*str*) – Bytom network, defaults to mainnet.

Returns WithdrawTransaction – Bytom withdraw transaction instance.

Warning: Do not forget to build transaction after initialize withdraw transaction.

```

build_transaction(address: str, transaction_hash: str, asset: Union[str,
    swap.providers.bytom.assets.AssetNamespace] =
    'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA') →
    swap.providers.bytom.transaction.WithdrawTransaction

```

Build Bytom withdraw transaction.

Parameters

- **address** (*str*) – Bytom recipient wallet address.
- **transaction_hash** (*str*) – Bytom funded transaction hash/id.
- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM.

Returns WithdrawTransaction – Bytom withdraw transaction instance.

```
>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.bytom.transaction.WithdrawTransaction object at 0x0409DAF0>
```

```
sign(solver: swap.providers.bytom.solver.WithdrawSolver) →  
    swap.providers.bytom.transaction.WithdrawTransaction  
Sign Bytom withdraw transaction.
```

Parameters `solver` (`bytom.solver.WithdrawSolver`) – Bytom withdraw solver.

Returns WithdrawTransaction – Bytom withdraw transaction instance.

```

>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> from swap.providers.bytom.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ "", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_transaction.sign(solver=withdraw_solver)
<swap.providers.bytom.transaction.WithdrawTransaction object at 0x0409DAF0>

```

transaction_raw() → str

Get Bytom withdraw transaction raw.

Returns str – Bytom withdraw transaction raw.

```

>>> from swap.providers.bytom.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "bm1q3plwvmvy4qhjmp5zffzmk50aagpujt6f5je85p", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.transaction_raw()

↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHud2d4cHpsZz"
↳ ""

```

7.3.4 RefundTransaction

class swap.providers.bytom.transaction.**RefundTransaction**(network: str = 'mainnet')

Bytom Refund transaction.

Parameters **network** (str) – Bytom network, defaults to mainnet.

Returns RefundTransaction – Bytom refund transaction instance.

Warning: Do not forget to build transaction after initialize refund transaction.

build_transaction(address: str, transaction_hash: str, asset: Union[str, swap.providers.bytom.assets.AssetNamespace] = 'ffffffffffffffffffffffffffffffff') → swap.providers.bytom.transaction.RefundTransaction

Build Bytom refund transaction.

Parameters

- **address** (*str*) – Bytom sender wallet address.
- **transaction_hash** (*str*) – Bytom funded transaction hash/id
- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM.

Returns *RefundTransaction* – Bytom refund transaction instance.

```
>>> from swap.providers.bytom.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.bytom.transaction.RefundTransaction object at 0x0409DAF0>
```

sign(*solver*: *swap.providers.bytom.solver.RefundSolver*) →
swap.providers.bytom.transaction.RefundTransaction
 Sign Bytom refund transaction.

Parameters *solver* (*bytom.solver.RefundSolver*) – Bytom refund solver.

Returns *RefundTransaction* – Bytom refund transaction instance.

```
>>> from swap.providers.bytom.transaction import RefundTransaction
>>> from swap.providers.bytom.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "58775359b7b3588dcdb1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ "", bytecode=bytecode)
>>> refund_transaction.sign(solver=refund_solver)
<swap.providers.bytom.transaction.RefundTransaction object at 0x0409DAF0>
```

transaction_raw() → *str*
 Get Bytom refund transaction raw.

Returns *str* – Bytom refund transaction raw.

```
>>> from swap.providers.bytom.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", transaction_hash=
↳ "59b1e43b57cba1afa5834eb9886e4a9fba031c9880ce7ae29d32c36f6b47496f", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> refund_transaction.transaction_raw()
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHh2d2d4cHpsZzY"
↳ ""
```

7.4 Solver

Bytom solver.

7.4.1 NormalSolver

```
class swap.providers.bytom.solver.NormalSolver(xprivate_key: str, account: int = 1, change: bool =
False, address: int = 1, path: Optional[str] = None,
indexes: Optional[List[str]] = None)
```

Bytom Normal solver.

Parameters

- **xprivate_key** (*str*) – Bytom sender xprivate key.
- **account** (*int*) – Bytom derivation account, defaults to 1.
- **change** (*bool*) – Bytom derivation change, defaults to False.
- **address** (*int*) – Bytom derivation address, defaults to 1.
- **path** (*str*) – Bytom derivation path, defaults to None.
- **indexes** (*list*) – Bytom derivation indexes, defaults to None.

Returns NormalSolver – Bytom normal solver instance.

```
>>> from swap.providers.bytom.solver import NormalSolver
>>> sender_xprivate_key =
↪ "205b15f70e253399da90b127b074ea02904594be9d54678207872ec1ba31ee51ef4490504bd2b6f99711367189245883
↪ "
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_xprivate_key)
<swap.providers.bytom.solver.NormalSolver object at 0x03FCCA60>
```

7.4.2 FundSolver

```
class swap.providers.bytom.solver.FundSolver(xprivate_key: str, account: int = 1, change: bool = False,
address: int = 1, path: Optional[str] = None, indexes:
Optional[List[str]] = None)
```

Bytom Fund solver.

Parameters

- **xprivate_key** (*str*) – Bytom sender xprivate key.
- **account** (*int*) – Bytom derivation account, defaults to 1.
- **change** (*bool*) – Bytom derivation change, defaults to False.
- **address** (*int*) – Bytom derivation address, defaults to 1.
- **path** (*str*) – Bytom derivation path, defaults to None.
- **indexes** (*list*) – Bytom derivation indexes, defaults to None.

Returns FundSolver – Bytom fund solver instance.

```
>>> from swap.providers.bytom.solver import FundSolver
>>> sender_xprivate_key: str =
↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8df4701
↪ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=sender_xprivate_key)
<swap.providers.bytom.solver.FundSolver object at 0x03FCCA60>
```

7.4.3 WithdrawSolver

```
class swap.providers.bytom.solver.WithdrawSolver(xprivate_key: str, secret_key: str, bytecode: str,
                                                  account: int = 1, change: bool = False, address: int
                                                  = 1, path: Optional[str] = None, indexes:
                                                  Optional[List[str]] = None)
```

Bytom Withdraw solver.

Parameters

- **xprivate_key** (*str*) – Bytom sender xprivate key.
- **secret_key** (*str*) – Secret password/passphrase.
- **bytecode** (*str*) – Bytom witness HTLC bytecode.
- **account** (*int*) – Bytom derivation account, defaults to 1.
- **change** (*bool*) – Bytom derivation change, defaults to False.
- **address** (*int*) – Bytom derivation address, defaults to 1.
- **path** (*str*) – Bytom derivation path, defaults to None.
- **indexes** (*list*) – Bytom derivation indexes, defaults to None.

Returns WithdrawSolver – Bytom withdraw solver instance.

```
>>> from swap.providers.bytom.solver import WithdrawSolver
>>> recipient_xprivate_key: str =
↪ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9f650b
↪ "
>>> bytecode: str =
↪ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa031d45
↪ "
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=recipient_
↪ xprivate_key, secret_key="Hello Meheret!", bytecode=bytecode)
<swap.providers.bytom.solver.WithdrawSolver object at 0x03FCCA60>
```

7.4.4 RefundSolver

```
class swap.providers.bytom.solver.RefundSolver(xprivate_key: str, bytecode: str, account: int = 1,
                                                change: bool = False, address: int = 1, path:
                                                Optional[str] = None, indexes: Optional[List[str]] =
                                                None)
```

Bytom Refund solver.

Parameters

- **xprivate_key** (*str*) – Bytom sender xprivate key.

- **bytecode** (*str*) – Bytom witness HTLC bytecode.
- **account** (*int*) – Bytom derivation account, defaults to 1.
- **change** (*bool*) – Bytom derivation change, defaults to False.
- **address** (*int*) – Bytom derivation address, defaults to 1.
- **path** (*str*) – Bytom derivation path, defaults to None.
- **indexes** (*list*) – Bytom derivation indexes, defaults to None.

Returns RefundSolver – Bytom refund solver instance.

```
>>> from swap.providers.bytom.solver import RefundSolver
>>> sender_xprivate_key: str =
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8df4701
↳ "
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa031d45
↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=sender_xprivate_key,
↳ bytecode=bytecode)
<swap.providers.bytom.solver.RefundSolver object at 0x03FCCA60>
```

7.5 Signature

Bytom signature.

class swap.providers.bytom.signature.**Signature**(*network: str = 'mainnet'*)
Bytom Signature.

Parameters **network** (*str*) – Bytom network, defaults to mainnet.

Returns Signature – Bytom signature instance.

Note: Bytom has only three networks, mainnet, solonet and testnet.

fee(*unit: str = 'NEU'*) → Union[int, float]
Get Bytom transaction fee.

Parameters **unit** (*str*) – Bytom unit, default to NEU.

Returns int, float – Bytom transaction fee.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGRrZHM0ZmttOGZ3djVrYXdtcTlxcnV
↳ "
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
```

(continues on next page)

(continued from previous page)

```
>>> signature.fee(unit="NEU")
449000
```

hash() → str

Get Bytom signature transaction hash.

Returns str – Bytom signature transaction hash or transaction id.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHhud2d4cHpsZzg"
↳ ""
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ ", secret_key='Hello Meheret!", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
>>> signature.hash()
"d1e84c37f41056f4df398523f84ecf079377fd85e4561c10ec03818cd4db7ec0"
```

json() → dict

Get Bytom signature transaction json format.

Returns dict – Bytom signature transaction json format.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXF0XzQNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdtcTlxcnV"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.json()
{"tx_id": "a3078af0810c68a7bb6f2f42cd67dce9dea3d77028ca0c527224e4524038abc4",
↳ "version": 1, "size": 275, "time_range": 0, "inputs": [{"type": "spend",
↳ "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↳ "asset_definition": {}, "amount": 189551000, "control_program":
↳ "0014b1592acbb917f13937166c2a9b6ce973296ebb60", "address":
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", "spent_output_id":
↳ "dd12e69d28a3e581b8b4501f9979ad39ba1e6b7e2163fe112a54a81fc2e8d6e3", "input_id
↳ ": "e55412ce943b72860ea06f7bc4c7ca4d9913b3dd736f8915279741c9a8c3bb2d",
↳ "witness_arguments": [
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212"], "sign_
↳ data": "f42a2b6e15585b88da8b34237c7a6fd83af12ee6971813d66cf794a63ebcc16f"}],
↳ "outputs": [{"type": "control", "id":
↳ "ecbd05faf0c2bec7706fb1d5230768d86eddc4d65fae2e4b0f995e6aa278c278", "position
↳ ": 0, "asset_id":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", "asset_
↳ definition": {}, "amount": 10000000, "control_program":
↳ "0020e7f4a9815f3a36c616c5666b97fb7fdacd3720c117d078c429494d1b617fe7d4",
↳ "address": "bm1qul62nq218gmvv9k9ve4e07mlmtxngwpzlg833pff9x3kctlul2q727jyy"},
↳ {"type": "control", "id":
```

(continues on next page)

(continued from previous page)

raw() → str

Get Bytom signature transaction raw.

Returns str – Bytom signature transaction raw.

```

>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
    ↪ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGRrZHM0ZmttOGZ3djVrYXdtcTlxcnVn"
    ↪ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
    ↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
    ↪ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
    ↪ solver)
>>> signature.raw()

    ↪ "07010001015f015df7df4d06a3fe3c8ac6438f25f9c97744a10455357857775526c3e6c752fb69eaaffffffff"
    ↪ ""

```

type() → str

Get Bytom signature transaction type.

Returns str – Bytom signature transaction type.

```

>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
    ↪ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHhud2d4cHpsZzgy"
    ↪ ""
>>> bytecode: str =
    ↪ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
    ↪ ""
>>> signature: Signature = Signature(network="mainnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
    ↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
    ↪ ", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_refund_transaction_raw,
    ↪ solver=refund_solver)
>>> signature.type()
    ↪ "bytom_refund_signed"

```

sign(transaction_raw: str, solver: Union[swap.providers.bytom.solver.NormalSolver, swap.providers.bytom.solver.FundSolver, swap.providers.bytom.solver.WithdrawSolver, swap.providers.bytom.solver.RefundSolver]) → Union[swap.providers.bytom.signature.NormalSignature, swap.providers.bytom.signature.FundSignature, swap.providers.bytom.signature.WithdrawSignature, swap.providers.bytom.signature.RefundSignature]

Sign unsigned transaction raw.

Parameters

- **transaction_raw** (str) – Bytom unsigned transaction raw.

- **solver** (bytom.solver.NormalSolver, bytom.solver.FundSolver, bytom.solver.WithdrawSolver, bytom.solver.RefundSolver) – Bytom solver

Returns NormalSignature, FundSignature, WithdrawSignature, RefundSignature – Bytom signature instance.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdteTlxcnVn"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
<swap.providers.bytom.signature.FundSignature object at 0x0409DAF0>
```

unsigned_datas() → List[dict]

Get Bytom transaction unsigned datas with instruction.

Returns list – Bytom transaction unsigned datas.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHhud2d4cHpsZzgy"
↳ ""
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ ", secret_key='Hello Meheret!", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
>>> signature.unsigned_datas()
[{"datas": ["45d1746a1ec0695d3e06059c413872040d24f86499ddafab48177368e5c72883"],
↳ "network": "mainnet", "path": null}]
```

signatures() → List[List[str]]

Get Bytom transaction signatures(signed datas).

Returns list – Bytom transaction signatures.

```
>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdteTlxcnVn"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
```

(continues on next page)

(continued from previous page)

```

>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.signatures()
[[
↳ "b82e97abc4b70f7ffe7f783254c63e61436d6a7ad15da89b1fb791f91d1d6aa0bab7ff86328eabd2959f5475dde
↳ "]]

```

transaction_raw() → str

Get Bytom signed transaction raw.

Returns str – Bytom signed transaction raw.

```

>>> from swap.providers.bytom.signature import Signature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcTlxcnV
↳ "
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.transaction_raw()

↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcTlxcnV
↳ "

```

7.5.1 NormalSignature

class swap.providers.bytom.signature.**NormalSignature**(network: str = 'mainnet')

Bytom Normal signature.

Parameters **network** (str) – Bytom network, defaults to mainnet.**Returns** NormalSignature – Bytom normal signature instance.**sign**(transaction_raw: str, solver: swap.providers.bytom.solver.NormalSolver) →*swap.providers.bytom.signature.NormalSignature*

Sign unsigned normal transaction raw.

Parameters

- **transaction_raw** (str) – Bytom unsigned normal transaction raw.
- **solver** (bytom.solver.NormalSolver) – Bytom normal solver.

Returns NormalSignature – Bytom normal signature instance.

```

>>> from swap.providers.bytom.signature import NormalSignature
>>> from swap.providers.bytom.solver import NormalSolver
>>> unsigned_normal_transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZlJlc3MiOiAiYm0xcTluZlhlseDAyc3lmd2Q3bnBlcGZ4ejRsZGRoenFzdmUyZnU
↳ "
>>> sender_xprivate_key: str =
↳ "205b15f70e253399da90b127b074ea02904594be9d54678207872ec1ba31ee510f4498504bd2b6f997113671892
↳ "

```

(continues on next page)

(continued from previous page)

```
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_xprivate_key)
>>> normal_signature: NormalSignature = NormalSignature(network="mainnet")
>>> normal_signature.sign(transaction_raw=unsigned_normal_transaction_raw,
↳ solver=normal_solver)
<swap.providers.bytom.signature.NormalSignature object at 0x0409DAF0>
```

7.5.2 FundSignature

```
class swap.providers.bytom.signature.FundSignature(network: str = 'mainnet')
    Bytom Fund signature.
```

Parameters **network** (*str*) – Bytom network, defaults to mainnet.

Returns FundSignature – Bytom fund signature instance.

```
sign(transaction_raw: str, solver: swap.providers.bytom.solver.FundSolver) →  
    swap.providers.bytom.signature.FundSignature  
Sign unsigned fund transaction raw.
```

Parameters

- **transaction_raw**(*str*) – Bytom unsigned fund transaction raw.
- **solver**([bytom.solver.FundSolver](#)) – Bytom fund solver.

Returns FundSignature – Bytom fund signature instance.

```
>>> from swap.providers.bytom.signature import FundSignature
>>> from swap.providers.bytom.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogImJtMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcTlxcnV"
↳ ""
>>> fund_signature: FundSignature = FundSignature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dcdc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> fund_signature.sign(transaction_raw=unsigned_fund_transaction_raw,
↳ solver=fund_solver)
<swap.providers.bytom.signature.FundSignature object at 0x0409DAF0>
```

7.5.3 WithdrawSignature

```
class swap.providers.bytom.signature.WithdrawSignature(network: str = 'mainnet')
    Bytom Withdraw signature.
```

Parameters `network` (*str*) – Bytom network, defaults to mainnet.

Returns WithdrawSignature – Bytom withdraw signature instance.

```
sign(transaction_raw: str, solver: swap.providers.bytom.solver.WithdrawSolver) →  
    swap.providers.bytom.signature.WithdrawSignature  
Sign unsigned withdraw transaction raw.
```

Parameters

- **transaction_raw** (*str*) – Bytom unsigned withdraw transaction raw.

- **solver** (`bytom.solver.WithdrawSolver`) – Bytom withdraw solver.

Returns `WithdrawSignature` – Bytom withdraw signature instance.

```
>>> from swap.providers.bytom.signature import WithdrawSignature
>>> from swap.providers.bytom.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZlJlc3MiOiAiYm0xcTNwbHd2bXZ5NHFoam1wNXpmZnptazUwYWFnchHVqdDZmNWp"
↳ ""
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> withdraw_signature: WithdrawSignature = WithdrawSignature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ "", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
<swap.providers.bytom.signature.WithdrawSignature object at 0x0409DAF0>
```

7.5.4 RefundSignature

class `swap.providers.bytom.signature.RefundSignature` (`network: str = 'mainnet'`)

Bytom Refund signature.

Parameters **network** (`str`) – Bytom network, defaults to `mainnet`.

Returns `RefundSignature` – Bytom withdraw signature instance.

sign (`transaction_raw: str, solver: swap.providers.bytom.solver.RefundSolver`) →

`swap.providers.bytom.signature.RefundSignature`

Sign unsigned refund transaction raw.

Parameters

- **transaction_raw** (`str`) – Bytom unsigned refund transaction raw.
- **solver** (`bytom.solver.RefundSolver`) – Bytom refund solver.

Returns `RefundSignature` – Bytom refund signature instance.

```
>>> from swap.providers.bytom.signature import RefundSignature
>>> from swap.providers.bytom.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHhhd2d4cHpsZzg"
↳ ""
>>> bytecode: str =
↳ "03285d0a20fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa03"
↳ ""
>>> refund_signature: RefundSignature = RefundSignature(network="mainnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ "", bytecode=bytecode)
>>> refund_signature.sign(transaction_raw=unsigned_refund_transaction_raw,
↳ solver=refund_solver)
<swap.providers.bytom.signature.RefundSignature object at 0x0409DAF0>
```

7.6 Remote Procedure Call (RPC)

Bytom remote procedure call.

```
swap.providers.bytom.rpc.get_balance(address: str, asset: Union[str,
swap.providers.bytom.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffff', network: str =
'mainnet', headers: dict = {'accept': 'application/json',
'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap
User-Agent 0.5.0'}, timeout: int = 60) → int
```

Get Bytom balance.

Parameters

- **address** (*str*) – Bytom address.
- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset, default to BTM.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *int* – Bytom asset balance (NEU amount).

```
>>> from swap.providers.bytom.rpc import get_balance
>>> from swap.providers.bytom.assets import BTM as ASSET
>>> get_balance(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", asset=ASSET,
↪network="mainnet")
71560900
```

```
swap.providers.bytom.rpc.get_utxos(program: str, network: str = 'mainnet', asset: Union[str,
swap.providers.bytom.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffff', limit: int = 15, by: str =
'amount', order: str = 'desc', headers: dict = {'accept':
'application/json', 'content-type': 'application/json; charset=utf-8',
'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → list
```

Get Bytom unspent transaction outputs (UTXO's).

Parameters

- **program** (*str*) – Bytom control program.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, defaults to BTM.
- **limit** (*int*) – Bytom utxo's limit, defaults to 15.
- **by** (*str*) – Sort by, defaults to amount.
- **order** (*str*) – Sort order, defaults to desc.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *list* – Bytom unspent transaction outputs (UTXO's).

```
>>> from swap.providers.bytom.rpc import get_utxos
>>> get_utxos(program="00142cda4f99ea8112e6fa61cdd26157ed6dc408332a", network=
↳ "mainnet")
[{'hash': '7c1e20e6ff719176a3ed6f5332ec3ff665ab28754d2511950e591267e0e675df', 'asset':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 71510800}, {'hash':
↳ '01b07c3523085b75f1e047be3a73b263635d0b86f9b751457a51b26c5a97a110', 'asset':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 500000}, {'hash': 'e46cfecc1f1a26413172ce81c78affb19408e613915642fa5fb04d3b0a4ffa65
↳ ', 'asset': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'amount': 100}]
```

```
swap.providers.bytom.rpc.estimate_transaction_fee(address: str, amount: int, asset: Union[str,
swap.providers.bytom.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
confirmations: int = 1, network: str = 'mainnet',
headers: dict = {'accept': 'application/json',
'content-type': 'application/json; charset=utf-8',
'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int
= 60) → int
```

Estimate Bytom transaction fee.

Parameters

- **address** (*str*) – Bytom address.
- **amount** (*int*) – Bytom amount (NEU amount).
- **asset** (*str*, *bytom.assets.AssetNamespace*) – Bytom asset id, default to BTM.
- **confirmations** (*int*) – Bytom confirmations, default to 1.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – request timeout, default to 60.

Returns *str* – Estimated transaction fee (NEU amount).

```
>>> from swap.providers.bytom.rpc import estimate_transaction_fee
>>> from swap.providers.bytom.assets import BTM as ASSET
>>> estimate_transaction_fee(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx",
↳ asset=ASSET, amount=100_000, confirmations=100, network="mainnet")
449000
```

```
swap.providers.bytom.rpc.account_create(xpublic_key: str, label: str = '1st address', account_index: int =
1, network: str = 'mainnet', headers: dict = {'accept':
'application/json', 'content-type': 'application/json;
charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout:
int = 60) → dict
```

Create account in blockcenter.

Parameters

- **xpublic_key** (*str*) – Bytom xpublic key.
- **label** (*str*) – Bytom limit, defaults to 1st address.
- **account_index** (*str*) – Account index, defaults to 1.

- **network** (*str*) – Bytom network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common` headers.
- **timeout** (*int*) – request timeout, default to `60`.

Returns dict – Bytom blockcenter guid, address and label.

```
>>> from swap.providers.bytom.rpc import account_create
>>> account_create(xpublic_key=
↳ "f80a401807fde1ee5727ae032ee144e4b757e69431e68e6cd732eda3c8cd3936daedfdd0fd8f8df14e2084c7e8df4701
↳ ", network="mainnet")
{"guid": "9ed61a9b-e7b6-4cb7-94fb-932b738e4f66", "address":
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9grufx", "label": "1st address"}
```

```
swap.providers.bytom.rpc.build_transaction(address: str, transaction: dict, network: str = 'mainnet',
                                             headers: dict = {'accept': 'application/json', 'content-type':
                                             'application/json; charset=utf-8', 'user-agent': 'Swap
                                             User-Agent 0.5.0'}, timeout: int = 60) → dict
```

Build Bytom transaction.

Parameters

- **address** (*str*) – Bytom address.
- **transaction** (*dict*) – Bytom transaction (inputs, outputs, fee, confirmations & forbid_chain_tx).
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Bytom built transaction.

```
>>> from swap.providers.bytom.rpc import build_transaction
>>> build_transaction(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx",
↳ transaction={"fee": "0.1", "confirmations": 1, "inputs": [{"type": "spend_wallet",
↳ "amount": "0.0001", "asset":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"}, {"type": "control_address", "amount": "0.0001", "asset":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", "address":
↳ "bm1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8"}]], network=
↳ "mainnet")
{'tx': {'hash': '5d4ae68487953863783599045f99eb8740b5745376ed8d8926d68de695e72476',
↳ 'status': True, 'size': 404, 'submission_timestamp': 0, 'memo': '', 'inputs': [{
↳ 'script': '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
↳ 'bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx', 'asset': {'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'decimals': 0,
↳ 'unit': 'BTM'}, 'amount': '0.0005', 'type': 'spend'}, {'script':
↳ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
↳ 'bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx', 'asset': {'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'decimals': 0,
↳ 'unit': 'BTM'}, 'amount': '0.715108', 'type': 'spend'}], 'outputs': [{'utxo_id':
↳ '0d5c097b8e75f711765ff63017fe8a4a987d8b50f7ca3a5d1873120af5f46116', 'script':
↳ '00204f8f0e88d0a44b3d884b07b6dd4536518ffccb596a91ca0e6b2f37e96463bbfc', 'address
↳ ': 'bm1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8', 'asset': {
↳ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'decimals': 0, 'unit': 'BTM'}, 'amount': '0.0001', 'type': 'control' (continues on next page)
↳ ': 'c49da44ef15d227ca978191e91d5d8915a3f92baf6b5778b7377deb2bddca554', 'script':
```

(continued from previous page)

```
swap.providers.bytom.rpc.get_transaction(transaction_hash: str, network: str = 'mainnet', headers: dict
                                         = {'accept': 'application/json', 'content-type':
                                             'application/json; charset=utf-8', 'user-agent': 'Swap
                                             User-Agent 0.5.0'}, timeout: int = 60) → dict
```

Get Bytom transaction detail.

Parameters

- **transaction_hash** (*str*) – Bytom transaction hash/id.
- **network** (*str*) – Bytom network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common` headers.
- **timeout** (*int*) – Request timeout, default to `60`.

Returns *dict* – Bytom transaction detail.

```
>>> from swap.providers.bytom.rpc import get_transaction
>>> get_transaction(transaction_hash=
    ↳ "bc935995cb3408b51aa3d05e7e77226840eb68340b229f9c561edae31ebc8b95", network=
    ↳ "mainnet")
{'id': 'bc935995cb3408b51aa3d05e7e77226840eb68340b229f9c561edae31ebc8b95',
 ↳ 'timestamp': 1524765978, 'block_height': 3487, 'trx_amount': 41249562600, 'trx_fee
 ↳ ': 437400, 'status_fail': False, 'coinbase': False, 'size': 332, 'chain_status':
 ↳ 'mainnet', 'time_range': 0, 'index_id': 3489, 'mux_id':
 ↳ '305a28d8d34b40c65936810f9e9c1f8bc9c793ec2e72c70f9203fbbbeb0a56db9', 'inputs': [{
 ↳ 'txtype': 'spend', 'asset_id':
 ↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 4
 ↳ 1250000000, 'control_program': '0014151df3db084d909ccb55d45d4e59db2e17e5f237',
 ↳ 'address': 'bm1qz5wl8kcgfkgfej6463w5ukwm9ct7tu3ht8p7te', 'spent_output_id':
 ↳ '6def8e6a7c29ccff4c5596a37a6698b71f392bf713bc67bb3fa0af54bf50f815', 'input_id':
 ↳ 'fb226e3ad39e38341f0d232c910065b76ef7c267faa3ea4e49a31836405b6747', 'witness_
 ↳ arguments': [
 ↳ '1848cb550620b971fd244eb625ccf4507ccd9944da65b47674550397c983247e1bd3ff880782beca963a81c34c17c8e
 ↳ ', '268402537b02d91fafdcdeb6eda3aa542548d77cd6cccb38ecd7ea7ce8a22cf7'], 'asset_
 ↳ name': 'BTM', 'asset_definition': '{}', 'cross_chain_asset': False, 'asset_
 ↳ decimals': 8}], 'outputs': [{ 'txtype': 'control', 'id':
 ↳ 'a8a7b5363379dee8ff77da7c4acf63dc3469a79ebaade079fc1842543964c6e9', 'asset_id':
 ↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 4
 ↳ 1239562600, 'control_program': '0014fc22634a713acle6f831c56184f847b7546fbda4',
 ↳ 'address': 'bm1qls3xxjn38tq7d7p3c4scf7z8ka2xl0dyppj52k', 'asset_name': 'BTM',
 ↳ 'asset_definition': '{}', 'cross_chain_asset': False, 'position': 0, 'asset_
 ↳ decimals': 8}, { 'txtype': 'control', 'id':
 ↳ '84287fb5b2b461dbd3b937a9013d89c0d54a21768e31fb8345b02d57a7992533', 'asset_id':
 ↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 1
 ↳ 00000000, 'control_program': '00140e43a92a9e8aca788eb1551c316448c2e3f78215',
 ↳ 'address': 'bm1qpep6j2573t983r4325wrzezgct3l0qs4q04pem', 'asset_name': 'BTM',
 ↳ 'asset_definition': '{}', 'cross_chain_asset': False, 'position': 1, 'asset_
 ↳ decimals': 8}], 'confirmations': 558509}
```

```
swap.providers.bytom.rpc.get_current_block_height(plus: int = 0, network: str = 'mainnet', headers: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → int
```

Get Bytom transaction detail.

Parameters

- **plus** (*int*) – Add block number on current block height, default to 0.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *int* – Bytom current block height.

```
>>> from swap.providers.bytom.rpc import get_current_block_height
>>> get_current_block_height(plus=0)
678722
```

```
swap.providers.bytom.rpc.find_p2wsh_utxo(transaction: dict) → Optional[dict]
```

Find Bytom pay to witness script hash UTXO info's.

Parameters **transaction** (*dict*) – Bytom transaction detail.

Returns *dict* – Pay to Witness Script Hash (P2WSH) UTXO info's.

```
>>> from swap.providers.bytom.rpc import find_p2wsh_utxo, get_transaction
>>> find_p2wsh_utxo(transaction=get_transaction(transaction_hash=
→ "b6d12407bbd238938941246fd0dd3e5234f1e3c370bef3fcbc1f60ebee022e76", network=
→ "mainnet"))
{'txtype': 'control', 'id':
→ 'alc5cce9df9343a10dafa582dea04e61c402ee8398b5268ba5c9c3aefd58017a', 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 1.
→ 10499000, 'control_program':
→ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcbb596a91ca0e6b2f37e96463bbfc', 'address
→ ': 'bm1qf78sazxs539nmzqtq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8', 'asset_name
→ ': 'BTM', 'asset_definition': '{}', 'cross_chain_asset': False, 'position': 0,
→ 'asset_decimals': 8}
```

```
swap.providers.bytom.rpc.decode_raw(raw: str, network: str = 'mainnet', headers: dict = {'accept':
→ 'application/json', 'content-type': 'application/json; charset=utf-8',
→ 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → dict
```

Decode original Bytom raw.

Parameters

- **raw** (*str*) – Bytom transaction raw.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *dict* – Bytom decoded transaction raw.

```

>>> from swap.providers.bytom.rpc import decode_raw
>>> decode_raw(raw=
→ "07010002015e015c88650475abf87eb364f93c608db879ad71643fbc7725ded246e8883e79c75a78fffffffffffffffffffff
→ ", network="testnet")
{'tx_id': '5d4ae68487953863783599045f99eb8740b5745376ed8d8926d68de695e72476',
→ 'version': 1, 'size': 404, 'time_range': 0, 'inputs': [{'type': 'spend', 'asset_id
→ ': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
→ definition': {}, 'amount': 50000, 'control_program':
→ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
→ 'bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx', 'spent_output_id':
→ '01b07c3523085b75f1e047be3a73b263635d0b86f9b751457a51b26c5a97a110', 'input_id':
→ 'de193c78772c93356f81a5061a90d8dcfba84d03ae4d78b2a57a9201f88c38af', 'witness_
→ arguments': ['91ff7f525ff40874c4f47f0cab42e46e3bf53adad59adef9558ad1b6448f22e2'],
→ 'sign_data': 'a5da2ae06bfaea9854423fe9cc544d775854cf57827c8c2ab606418452d30209'},
→ {'type': 'spend', 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
→ definition': {}, 'amount': 71510800, 'control_program':
→ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
→ 'bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx', 'spent_output_id':
→ '7cle20e6ff719176a3ed6f5332ec3ff665ab28754d2511950e591267e0e675df', 'input_id':
→ 'de2c7bcf9caf00f78ca8e316cf37cf88c86b0457e47cf58e2465d783151abd0e', 'witness_
→ arguments': ['91ff7f525ff40874c4f47f0cab42e46e3bf53adad59adef9558ad1b6448f22e2'],
→ 'sign_data': '3e44203712c4e981783810875fa67f2efe0afda38afe229fd09da0d113c3d885'}],
→ 'outputs': [{'type': 'control', 'id':
→ '0d5c097b8e75f711765ff63017fe8a4a987d8b50f7ca3a5d1873120af5f46116', 'position': 0,
→ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
→ 'asset_definition': {}, 'amount': 10000, 'control_program':
→ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'address
→ ': 'bm1qf78sazxs539nmzqtq7md63fk2x8lew6ed2gu5rnt9um7jerrh07q3yf5q8'}, {'type':
→ 'control', 'id': 'c49da44ef15d227ca978191e91d5d8915a3f92baf6b5778b7377deb2bddca554
→ ', 'position': 1, 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
→ definition': {}, 'amount': 61590800, 'control_program':
→ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
→ 'bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx'}], 'fee': 9960000}

```

```

swap.providers.bytom.rpc.submit_raw(address: str, raw: str, signatures: list, network: str = 'mainnet',
→ headers: dict = {'accept': 'application/json', 'content-type':
→ 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent
→ 0.5.0'}, timeout: int = 60) → str

```

Submit original Bytom raw into blockchain.

Parameters

- **address** (*str*) – Bytom address.
- **raw** (*str*) – Bytom transaction raw.
- **signatures** (*list*) – Bytom signed message datas.
- **network** (*str*) – Bytom network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common` headers.
- **timeout** (*int*) – Request timeout, default to `60`.

Returns str – Bytom submitted transaction id/hash.

```
>>> from swap.providers.bytom.rpc import submit_raw
>>> submit_raw(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", raw=
↳ "07010002015e015c88650475abf87eb364f93c608db879ad71643fbc7725ded246e8883e79c75a78fffffffffffffffffffff
↳ ", signatures=[[
↳ "f8466336a79d166e47fb5d64f1e7ec01b203b59b3ee86686492bd1e4d0bdd642dfe4a575049071a052a441635c33670
↳ "], [
↳ "ebf33fbda5c2f3d144e90c3b763b1e7e42d501e595216fcd2b310b089918bae2ef4c7b8a2e1f650ee741578aba796070
↳ "]]], network="mainnet")
"2993414225f65390220730d0c1a356c14e91bca76db112d37366df93e364a492"
```

7.7 Utils

Bytom Utils.

swap.providers.bytom.utils.get_address_type(address: str) → Optional[str]
Get Bytom address type.

Parameters address (str) – Bytom address.

Returns str – Bytom address type (P2WPKH, P2WSH).

```
>>> from swap.providers.bytom.utils import get_address_type
>>> get_address_type(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx")
"p2wpkh"
```

swap.providers.bytom.utils.is_network(network: str) → bool
Check Bytom network.

Parameters network (str) – Bytom network.

Returns bool – Bytom valid/invalid network.

```
>>> from swap.providers.bytom.utils import is_network
>>> is_network(network="solonet")
True
```

swap.providers.bytom.utils.is_address(address: str, network: Optional[str] = None, address_type: Optional[str] = None) → bool
Check Bytom address.

Parameters

- address (str) – Bytom address.
- network (str) – Bytom network, defaults to None.
- address_type (str) – Bytom address type, defaults to None.

Returns bool – Bytom valid/invalid address.

```
>>> from swap.providers.bytom.utils import is_address
>>> is_address(address="bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", network=
↳ "mainnet")
True
```

`swap.providers.bytom.utils.is_transaction_raw(transaction_raw: str) → bool`
 Check Bytom transaction raw.

Parameters `transaction_raw (str)` – Bytom transaction raw.

Returns `bool` – Bytom valid/invalid transaction raw.

```
>>> from swap.providers.bytom.utils import is_transaction_raw
>>> transaction_raw =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHZJLc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTbwenAyNGRrZmZlbHlzM0Hpjd3l1"
↳ ""
>>> is_transaction_raw(transaction_raw=transaction_raw)
True
```

`swap.providers.bytom.utils.amount_unit_converter(amount: Union[int, float], unit_from: str = 'NEU2BTM') → Union[int, float]`

Bytom amount unit converter

Parameters

- **amount** (`int`, `float`) – Bytom any amount.
- **unit_from** (`str`) – Bytom unit convert from symbol, default to NEU2BTM.

Returns `int`, `float` – BTM asset amount.

```
>>> from swap.providers.bytom.utils import amount_unit_converter
>>> amount_unit_converter(amount=10_000_000, unit_from="NEU2BTM")
0.1
```

`swap.providers.bytom.utils.estimate_endblock(endtime: int, network: str = 'mainnet', headers: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → int`

Estimate Bytom expiration block height.

Parameters

- **endtime** (`int`) – Expiration block timestamp.
- **network** (`str`) – Bytom network, defaults to mainnet.
- **headers** (`dict`) – Request headers, default to common-headers.
- **timeout** (`int`) – Request timeout, default to 60.

Returns `str` – Estimated Vapor endblock.

```
>>> from swap.providers.bytom.utils import estimate_endblock
>>> from swap.utils import get_current_timestamp
>>> estimate_endblock(endtime=get_current_timestamp(plus=3600))
680854
```

`swap.providers.bytom.utils.decode_transaction_raw(transaction_raw: str, headers: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → dict`

Decode Bytom transaction raw.

Parameters

- **transaction_raw** (`str`) – Bytom transaction raw.

- **headers** (*dict*) – Request headers, default to `common-headers`.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *dict* – Decoded Bytom transaction raw.

```
>>> from swap.providers.bytom.utils import decode_transaction_raw
>>> transaction_raw =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTbwenAyNGRrZmZlbHlzOHpjd311"
↳ ""
>>> decode_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'address': '...', 'transaction': {...}, 'unsigned_datas
↳ ': [...], 'signatures': [...], 'network': '...'}
```

`swap.providers.bytom.utils.submit_transaction_raw`(*transaction_raw: str*, *headers: dict* = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, *timeout: int* = 60) → *dict*

Submit Bytom transaction raw.

Parameters

- **transaction_raw** (*str*) – Bytom transaction raw.
- **headers** (*dict*) – Request headers, default to `common-headers`.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *dict* – Bytom submitted transaction id, fee, type and date.

```
>>> from swap.providers.bytom.utils import submit_transaction_raw
>>> transaction_raw =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTbwenAyNGRrZmZlbHlzOHpjd311"
↳ ""
>>> submit_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'transaction_hash': '...', 'network': '...', 'date': '..
↳ :.'}
```

ETHEREUM

Ethereum is a decentralized, open-source blockchain with smart contract functionality. Ether is the native cryptocurrency of the platform. After Bitcoin, it is the second-largest cryptocurrency by market capitalization. Ethereum is the most actively used blockchain.

For more <https://ethereum.org>

8.1 Wallet

The implementation of Hierarchical Deterministic (HD) wallets generator for Ethereum blockchain.

```
class swap.providers.ethereum.wallet.Wallet(network: str = 'mainnet', provider: str = 'http', token:
Optional[str] = None)
```

Ethereum Wallet class.

Parameters

- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns Wallet – Ethereum wallet instance.

Note: Ethereum has only five networks, `mainnet`, `ropsten`, `kovan`, `rinkeby` and `testnet`.

```
from_entropy(entropy: str, language: str = 'english', passphrase: Optional[str] = None) →
swap.providers.ethereum.wallet.Wallet
```

Master from Entropy.

Parameters

- **entropy** (*str*) – Ethereum wallet entropy.
- **language** (*str*) – Ethereum wallet language, default to `english`.
- **passphrase** (*str*) – Ethereum wallet passphrase, default to `None`.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_mnemonic(mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None) → *swap.providers.ethereum.wallet.Wallet*

Master from Mnemonic.

Parameters

- **mnemonic** (str) – Ethereum wallet mnemonic.
- **language** (str) – Ethereum wallet language, default to english.
- **passphrase** (str) – Ethereum wallet passphrase, default to None.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park_
↳ clown build renew illness fault")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_seed(seed: str) → *swap.providers.ethereum.wallet.Wallet*

Master from Seed.

Parameters **seed** (str) – Ethereum wallet seed.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_seed(seed=
↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ ")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_xprivate_key(xprivate_key: str, strict: bool = True) → *swap.providers.ethereum.wallet.Wallet*

Master from Root XPrivate Key.

Parameters

- **xprivate_key** (str) – Ethereum root xprivate key.
- **strict** (bool) – Strict for must be root xprivate key, default to True.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUUbHly3kQf
↳ ")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_xpublic_key(xpublic_key: str, strict: bool = True) → *swap.providers.ethereum.wallet.Wallet*

Master from Root XPublic Key.

Parameters

- **xpublic_key** (*str*) – Ethereum root xpublic key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xpublic_key(xpublic_key=
↳ "xpub661MyMwAqRbcG28HjdHc6zbHxBFzBJBC4ecFvVKBXxqiucEB5wirgQ9hzY2WQMjnurVjJbTjMWRskHi7jnSRkJ
↳ ")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_wif(*wif: str*) → *swap.providers.ethereum.wallet.Wallet*

Master from Wallet Important Format (WIF).

Parameters **wif** (*str*) – Ethereum wallet important format.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_wif(wif="L4AfqFc8aoBWYNTKU6PkiFbP9kbXRfVHXZWde6SpAdTewwJMc5VZ")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_private_key(*private_key*) → *swap.providers.ethereum.wallet.Wallet*

Master from Private Key.

Parameters **private_key** (*str*) – Ethereum private key.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_private_key(private_key=
↳ "cf4c2fb2b88a556c211d5fe79335dcee6dd11403bbbc5b47a530e9cf56ee3aee")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_path(*path: str*) → *swap.providers.ethereum.wallet.Wallet*

Drive Ethereum wallet from path.

Parameters **path** (*str*) – Ethereum wallet path.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

from_index(*index: int, hardened: bool = False*) → *swap.providers.ethereum.wallet.Wallet*

Drive Ethereum wallet from index.

Parameters

- **index** (*int*) – Ethereum wallet index.
- **hardened** (*bool*) – Use hardened index, default to False.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_index(44, harden=True)
>>> wallet.from_index(60, harden=True)
>>> wallet.from_index(0, harden=True)
>>> wallet.from_index(0)
>>> wallet.from_index(0)
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
```

clean_derivation() → *swap.providers.ethereum.wallet.Wallet*

Clean derivation Ethereum wallet.

Returns Wallet – Ethereum wallet instance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/60'/0'/0/0")
>>> wallet.path()
"m/44'/60'/0'/0/0"
>>> wallet.clean_derivation()
<swap.providers.ethereum.wallet.Wallet object at 0x040DA268>
>>> wallet.path()
None
```

strength() → Optional[int]

Get Ethereum wallet strength.

Returns int – Ethereum wallet strength.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.strength()
128
```

entropy() → Optional[str]

Get Ethereum wallet entropy.

Returns str – Ethereum wallet entropy.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.entropy()
"ed0802d701a033776811601dd6c5c4a9"
```

mnemonic() → Optional[str]

Get Ethereum wallet mnemonic.

Returns str – Ethereum wallet mnemonic.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.mnemonic()
"unfair divorce remind addict add roof park clown build renew illness fault"
```

passphrase() → Optional[str]

Get Ethereum wallet passphrase.

Returns str – Ethereum wallet passphrase.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9", passphrase=
↳ "meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]

Get Ethereum wallet language.

Returns str – Ethereum wallet language.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.language()
"english"
```

seed() → Optional[str]

Get Ethereum wallet seed.

Returns str – Ethereum wallet seed.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.seed()

↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ "
```

root_xprivate_key(encoded: bool = True) → Optional[str]

Get Ethereum wallet root xprivate key.

Parameters **encoded** (bool) – Encoded root xprivate key, default to True.**Returns** str – Ethereum wallet root xprivate key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xprivate_key()

↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHly3kQf
↳ "
```

root_xpublic_key(*encoded: bool = True*) → Optional[str]

Get Ethereum wallet root xpublic key.

Parameters **encoded** (*bool*) – Encoded root xprivate key, default to True.

Returns str – Ethereum wallet root xpublic key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xpublic_key()

↪ "xpub661MyMwAqRbcG28HjdHc6zbHxBFzBJBC4ecFvVKBXxiucEBE5wirgQ9hzY2WQMjnurVjJbTjMWRskHi7jnSRkJo"
↪ "
```

xprivate_key(*encoded=True*) → Optional[str]

Get Ethereum wallet xprivate key.

Parameters **encoded** (*bool*) – Encoded xprivate key, default to True.

Returns str – Ethereum wallet xprivate key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.xprivate_key()

↪ "xprvA3xrxQQVw6Kvc786WAccK4H7dLHhnb9XRsmUMqU3bJoZf5bWxtD5VePTNnn854tEbvV57ggjqkGHXc2u4Jx2veJ"
↪ "
```

xpublic_key(*encoded: bool = True*) → Optional[str]

Get Ethereum wallet xpublic key.

Parameters **encoded** (*bool*) – Encoded xprivate key, default to True.

Returns str – Ethereum wallet xpublic key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.xpublic_key()

↪ "xpub6GxDMuwPmTtDpbCZcC9cgCDrBN8CC3sNo6H5ADsf9eLYXsvfWRwL3ShwE5u4gxbPPcZj1yJSDrvvLxsHEPdjtFHF"
↪ "
```

uncompressed() → str

Get Ethereum wallet uncompressed public key.

Returns str – Ethereum wallet uncompressed public key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.uncompressed()

↪ "e270f9d51cad2977c0a28182b9320bb5edc3c70e6d84ff5837f8d407ed9d676d417e195e1af5494d1a0c8dc310"
↪ "
```

(continued from previous page)

compressed() → str

Get Ethereum wallet compressed public key.

Returns str – Ethereum wallet compressed public key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.compressed()
"03e270f9d51cad2977c0a28182b9320bb5edc3c70e6d84ff5837f8d407ed9d676d"
```

chain_code() → str

Get Ethereum wallet chain code.

Returns str – Ethereum wallet chain code.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.chain_code()
"9e5c492fa0a5c5cc649922c34ac3468a08473f3b61f59bba61b52cce364d6b0c"
```

private_key() → str

Get Ethereum wallet private key.

Returns str – Ethereum wallet private key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.private_key()
"cf4c2fb2b88a556c211d5fe79335dcee6dd11403bbbc5b47a530e9cf56ee3aee"
```

public_key() → str

Get Ethereum wallet public key.

Returns str – Ethereum wallet public key.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.public_key()
"03e270f9d51cad2977c0a28182b9320bb5edc3c70e6d84ff5837f8d407ed9d676d"
```

path() → Optional[str]

Get Ethereum wallet path.

Returns str – Ethereum wallet path.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.path()
"m/44'/60'/0'/0/0"
```

address() → str

Get Ethereum wallet address.

Returns str – Ethereum wallet address.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.address()
"0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C"
```

wif() → str

Get Ethereum wallet important format (WIF).

Returns str – Ethereum wallet important format.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.wif()
"L4AfqFc8aoBWYNTKU6PkiFbP9kbXRfVHXZWde6SpAdTewwJMc5VZ"
```

hash(private_key: Optional[str] = None) → str

Get Ethereum wallet public key/address hash.

Returns str – Ethereum wallet public key/address hash.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.hash()
"184847379abdde6617e8438fd4ff0d8fdf512cc2"
```

balance(unit: str = 'Wei') → Union[Wei, int, float]

Get Ethereum wallet balance.

Parameters unit (str) – Ethereum unit, default to Wei.

Returns Wei, int, float – Ethereum wallet balance.

```
>>> from swap.providers.ethereum.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/60'/0'/0/0")
>>> wallet.balance(unit="Ether")
96.96263982
```


Parameters `private_key` (*str*) – Ethereum private key.

Returns HTLC – Ethereum HTLC instance.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C")
>>> htlc.sign_transaction(private_key=
↳ "cf4c2fb2b88a556c211d5fe79335dcee6dd11403bbbc5b47a530e9cf56ee3aee")
<swap.providers.ethereum.htlc.HTLC object at 0x0409DAF0>
```

fee(*unit: str = 'Wei'*) → Union[Wei, int, float]

Get Ethereum HTLC transaction fee.

Parameters `unit` (*str*) – Ethereum unit, default to Wei.

Returns Wei, int, float – Ethereum transaction fee.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C")
>>> htlc.fee(unit="Wei")
1532774
```

hash() → Optional[str]

Get Ethereum HTLC transaction hash.

Returns str – Ethereum transaction hash.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C")
>>> htlc.sign_transaction(private_key=
↳ "cf4c2fb2b88a556c211d5fe79335dcee6dd11403bbbc5b47a530e9cf56ee3aee")
>>> htlc.hash()
"0x500953d43ff95604f5ffeb8f6c0e565d9080aa6aa31d8924a8d9df78c1f27879"
```

json() → dict

Get Ethereum HTLC transaction json.

Returns dict – Ethereum transaction json.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C")
>>> htlc.json()
{'chainId': 1337, 'from': '0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C', 'value': 0, 'nonce': 6, 'gas': 1532774, 'gasPrice': 20000000000, 'data': '0x608060405234801561001057600080fd5b50611ae9806100206000396000f3fe608060405260004361061003f5', 'to': b''}
```

raw() → Optional[str]

Get Ethereum HTLC transaction raw.

Returns str – Ethereum transaction raw.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C")
>>> htlc.sign_transaction(private_key=
↳ "cf4c2fb2b88a556c211d5fe79335dcee6dd11403bbbc5b47a530e9cf56ee3aee")
>>> htlc.raw()

↳ "0xf91b5e058504a817c800831763668080b91b09608060405234801561001057600080fd5b50611ae9806100206"
↳ "
```

contract_address() → ChecksumAddress

Get Ethereum HTLC contract address.

Returns ChecksumAddress – Ethereum HTLC contract address.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(contract_address=
↳ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
>>> htlc.contract_address()
"0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40"
```

build_htlc(secret_hash: str, recipient_address: str, sender_address: str, endtime: int, token_address: Optional[str] = None) → swap.providers.ethereum.htlc.HTLC

Build Ethereum Hash Time Lock Contract (HTLC).

Parameters

- **secret_hash** (str) – Secret sha-256 hash.
- **recipient_address** (str) – Ethereum recipient address.
- **sender_address** (str) – Ethereum sender address.
- **endtime** (int) – Expiration block timestamp.
- **token_address** (bool) – Ethereum ERC20 token address, default to None.

Returns HTLC – Ethereum HTLC instance.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
<swap.providers.ethereum.htlc.HTLC object at 0x0409DAF0>
```

abi() → list

Get Ethereum HTLC ABI.

Returns list – Ethereum HTLC ABI.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
```

(continues on next page)

(continued from previous page)

```

>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
>>> htlc.abi()
[{'anonymous': False, 'inputs': [{'indexed': True, 'internalType': 'bytes32',
↳ 'name': 'locked_contract_id', 'type': 'bytes32'}, {'indexed': False,
↳ 'internalType': 'bytes32', 'name': 'secret_hash', 'type': 'bytes32'}, {
↳ 'indexed': True, 'internalType': 'address', 'name': 'recipient', 'type':
↳ 'address'}, {'indexed': True, 'internalType': 'address', 'name': 'sender',
↳ 'type': 'address'}, {'indexed': False, 'internalType': 'uint256', 'name':
↳ 'endtime', 'type': 'uint256'}, {'indexed': False, 'internalType': 'uint256',
↳ 'name': 'amount', 'type': 'uint256'}], 'name': 'log_fund', 'type': 'event'}, {
↳ 'anonymous': False, 'inputs': [{'indexed': True, 'internalType': 'bytes32',
↳ 'name': 'locked_contract_id', 'type': 'bytes32'}], 'name': 'log_refund', 'type
↳ ': 'event'}, {'anonymous': False, 'inputs': [{'indexed': True, 'internalType
↳ ': 'bytes32', 'name': 'locked_contract_id', 'type': 'bytes32'}], 'name': 'log_
↳ withdraw', 'type': 'event'}, {'inputs': [{'internalType': 'bytes32', 'name':
↳ '_secret_hash', 'type': 'bytes32'}, {'internalType': 'address payable', 'name
↳ ': '_recipient', 'type': 'address'}, {'internalType': 'address payable', 'name
↳ ': '_sender', 'type': 'address'}, {'internalType': 'uint256', 'name': '_
↳ endtime', 'type': 'uint256'}], 'name': 'fund', 'outputs': [{'internalType':
↳ 'bytes32', 'name': 'locked_contract_id', 'type': 'bytes32'}], 'stateMutability
↳ ': 'payable', 'type': 'function'}, {'inputs': [{'internalType': 'bytes32',
↳ 'name': '_locked_contract_id', 'type': 'bytes32'}], 'name': 'get_locked_
↳ contract', 'outputs': [{'internalType': 'bytes32', 'name': 'id', 'type':
↳ 'bytes32'}, {'internalType': 'bytes32', 'name': 'secret_hash', 'type':
↳ 'bytes32'}, {'internalType': 'address', 'name': 'recipient', 'type': 'address
↳ '}, {'internalType': 'address', 'name': 'sender', 'type': 'address'}, {
↳ 'internalType': 'uint256', 'name': 'endtime', 'type': 'uint256'}, {
↳ 'internalType': 'uint256', 'name': 'amount', 'type': 'uint256'}, {
↳ 'internalType': 'bool', 'name': 'withdrawn', 'type': 'bool'}, {'internalType
↳ ': 'bool', 'name': 'refunded', 'type': 'bool'}, {'internalType': 'string',
↳ 'name': 'preimage', 'type': 'string'}], 'stateMutability': 'view', 'type':
↳ 'function'}, {'inputs': [{'internalType': 'bytes32', 'name': '_locked_
↳ contract_id', 'type': 'bytes32'}], 'name': 'refund', 'outputs': [{
↳ 'internalType': 'bool', 'name': '', 'type': 'bool'}], 'stateMutability':
↳ 'nonpayable', 'type': 'function'}, {'inputs': [{'internalType': 'bytes32',
↳ 'name': '_locked_contract_id', 'type': 'bytes32'}, {'internalType': 'string',
↳ 'name': '_preimage', 'type': 'string'}], 'name': 'withdraw', 'outputs': [{
↳ 'internalType': 'bool', 'name': '', 'type': 'bool'}], 'stateMutability':
↳ 'nonpayable', 'type': 'function'}]}

```

bytecode() → str

Get Ethereum HTLC bytecode.

Returns str – Ethereum HTLC bytecode.

```

>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))

```

(continues on next page)

(continued from previous page)

```
>>> htlc.bytecode()
```

```
↪ "608060405234801561001057600080fd5b50611ae9806100206000396000f3fe60806040526004361061003f5760"
↪ "
```

bytecode_runtime() → str

Get Ethereum HTLC bytecode runtime.

Returns str – Ethereum HTLC bytecode runtime.

```
>>> from swap.providers.ethereum.htlc import HTLC
```

```
>>> from swap.utils import sha256, get_current_timestamp
```

```
>>> htlc: HTLC = HTLC(contract_address=
```

```
↪ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
```

```
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
```

```
↪ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
```

```
↪ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
```

```
↪ timestamp(plus=3600))
```

```
>>> htlc.bytecode_runtime()
```

```
↪ "60806040526004361061003f5760003560e01c806306a53665146100445780637249fbb614610081578063cfd4b"
↪ "
```

opcode() → str

Get Ethereum HTLC opcode.

Returns str – Ethereum HTLC opcode.

```
>>> from swap.providers.ethereum.htlc import HTLC
```

```
>>> from swap.utils import sha256, get_current_timestamp
```

```
>>> htlc: HTLC = HTLC(contract_address=
```

```
↪ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
```

```
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
```

```
↪ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
```

```
↪ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
```

```
↪ timestamp(plus=3600))
```

```
>>> htlc.bytecode_runtime()
```

```
"PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0
```

```
↪ DUP1 REVERT JUMPDEST POP PUSH2 0x1AE9 DUP1 PUSH2 0x20 PUSH1 0x0 CODECOPY
```

```
↪ PUSH1 0x0 RETURN INVALID PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE
```

```
↪ LT PUSH2 0x3F JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0xE0 SHR DUP1 PUSH4
```

```
↪ 0x6A53665 EQ PUSH2 0x44 JUMPI DUP1 PUSH4 0x7249FBB6 EQ PUSH2 0x81 JUMPI DUP1
```

```
↪ PUSH4 0xCFD4B66E EQ PUSH2 0xBE JUMPI DUP1 PUSH4 0xF4FD3062 EQ PUSH2 0x103
```

```
↪ JUMPI JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO PUSH2
```

```
↪ 0x50 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0x6B PUSH1 0x4 DUP1
```

```
↪ CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2 0x66 SWAP2 SWAP1 PUSH2 0xF29 JUMP
```

```
↪ JUMPDEST PUSH2 0x133 JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x78 SWAP2 SWAP1
```

```
↪ PUSH2 0x12E1 JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN
```

```
↪ JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0x8D JUMPI PUSH1 0x0 DUP1 REVERT
```

```
↪ JUMPDEST POP PUSH2 0xA8 PUSH1 0x4 DUP1 CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2
```

```
↪ 0xA3 SWAP2 SWAP1 PUSH2 0xE74 JUMP JUMPDEST PUSH2 0x449 JUMP JUMPDEST PUSH1
```

```
↪ 0x40 MLOAD PUSH2 0xB5 SWAP2 SWAP1 PUSH2 0x12E1 JUMP JUMPDEST PUSH1 0x40 MLOAD
```

```
↪ DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0xCA JUMPI
```

```
↪ PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0xE5 PUSH1 0x4 DUP1 CALLDATASIZE SUB
```

```
↪ DUP2 ADD SWAP1 PUSH2 0xE0 SWAP2 SWAP1 PUSH2 0xE74 JUMP JUMPDEST PUSH2 0x73D
```

```
↪ JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0xFA SWAP10 SWAP9 SWAP8 SWAP7 SWAP6
```

```
↪ SWAP5 SWAP4 SWAP3 SWAP2 SWAP1 PUSH2 0x1317 JUMP JUMPDEST PUSH1 0x40 MLOAD
```

```
↪ DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH2 0x11D PUSH1 0x4 DUP1 CALLDATASIZE
```

```
↪ SUB DUP2 ADD SWAP1 PUSH2 0x118 SWAP2 SWAP1 PUSH2 0xEC6 JUMP JUMPDEST PUSH2
```

```
↪ 0x8E4 JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x12A SWAP2 SWAP1 PUSH2 0x12FC
```

```
↪ JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH1 0x0
```

8.2. Hash Time Lock Contract (HTLC)

117

(continued from previous page)

```
balance(unit: str = 'Wei') → Union[Wei, int, float]
```

Get Ethereum HTLC balance.

Parameters **unit** (*str*) – Ethereum unit, default to Ether.

Returns int, float – Ethereum HTLC balance.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> htlc: HTLC = HTLC(contract_address=
↳ "0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40", network="testnet")
>>> htlc.balance(unit="Ether")
1.56
```

$$\mathbf{erc20_balance}(token_address: str) \rightarrow \text{Tuple}[\text{int}, \text{str}, \text{str}, \text{int}, \text{str}]$$

Get Ethereum HTLC ERC20 balance.

Parameters `token_address` (*str*) – Ethereum ERC20 token address.

Returns tuple – Ethereum HTLC ERC20 balance and decimal.

[illegible]

8.3 Transaction

Ethereum transaction in blockchain network.

```
class swap.providers.ethereum.transaction.Transaction(network: str = 'mainnet', erc20: bool = False,
                                                       provider: str = 'http', token: Optional[str] =
                                                       None)
```

Ethereum Transaction.

Parameters

- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **erc20** (*bool*) – Transaction ERC20 token, default to `False`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns Transaction – Ethereum transaction instance.

Note: Ethereum has only five networks, mainnet, ropsten, kovan, rinkeby and testnet.

```
fee(unit: str = 'Wei') → Union[Wei, int, float]
```

Get Ethereum transaction fee.

Parameters `unit (str)` – Ethereum unit, default to Wei.

Returns Wei, int, float – Ethereum transaction fee.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.providers.ethereum.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", htlc=htlc, amount=100_000_000)
>>> fund_transaction.fee(unit="Wei")
1532774
```

hash() → Optional[str]

Get Ethereum transaction hash.

Returns str – Ethereum transaction hash.

```
>>> from swap.providers.ethereum.transaction import WithdrawTransaction
>>> from swap.providers.ethereum.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", secret_
↳ key="Hello Meheret!", address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C",
↳ contract_address="0x67324d402ffc103d061dAfa9096ff639f0676378")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUjt5frjwihGwMv1fQEDioiGZXWUubHly3kQf
↳ ", address=1)
>>> withdraw_transaction.sign(solver=withdraw_solver)
>>> withdraw_transaction.hash()
"0x9bbf83e56fea4cd9d23e000e8273551ba28317e4d3c311a49be919b305feb711"
```

json() → dict

Get Ethereum transaction fee.

Returns Wei, int, float – Ethereum transaction fee.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.providers.ethereum.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", htlc=htlc, amount=100_000_000)
>>> fund_transaction.json()
```

(continues on next page)

(continued from previous page)

```
{'chainId': 1337, 'from': '0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C', 'value': 3000000000000000000, 'nonce': 0, 'gas': 22488, 'gasPrice': 20000000000, 'to': '0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40', 'data': '0xf4fd30623a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb000000000000000000000000'} }
```

raw() → Optional[str]
Get Ethereum transaction hash.

Returns str – Ethereum transaction hash.

```
>>> from swap.providers.ethereum.transaction import RefundTransaction
>>> from swap.providers.ethereum.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", contract_address=
↳ "0x67324d402ffc103d061dAfA9096ff639f0676378")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmQTLhRgf86uZyCJk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHLY3kQf
↳ ", address=0)
>>> refund_transaction.sign(solver=refund_solver)
>>> refund_transaction.hash()
"0x9bbf83e56fea4cd9d23e000e8273551ba28317e4d3c311a49be919b305feb711"
```

type() → str
Get Ethereum transaction hash.

Returns str – Ethereum transaction hash.

```
>>> from swap.providers.ethereum.transaction import WithdrawTransaction
>>> from swap.providers.ethereum.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", secret_
↳ key="Hello Meheret!", address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C",
↳ contract_address="0x67324d402ffc103d061dAfA9096ff639f0676378")
>>> withdraw_transaction.type()
"ethereum_withdraw_unsigned"
```

signature() → dict
Get Ethereum transaction hash.

Returns str – Ethereum transaction hash.

```
>>> from swap.providers.ethereum.transaction import RefundTransaction
>>> from swap.providers.ethereum.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0xe49ff5077739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", contract_address=
↳ "0x67324d402ffc103d061dAfA9096ff639f0676378")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s2t1ZrQHf43K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwV1f0EDioiGZXWXUbHLY3kQf
↳ ", address=0)
(continues on next page)
```

(continued from previous page)

```
>>> refund_transaction.sign(solver=refund_solver)
>>> refund_transaction.signature()
{'hash': '0x120241e6e89b54d90dc3a3f73d6353f83818c3d404c991d3b74691f000583396',
  'rawTransaction':
  '0xf8f4018504a817c80083021cd094eaeac81da5e386e8ca4de1e64d40a10e468a5b408829a2241af62c0000b88',
  'r': '42223337416619984402386667584480976881779168344975798352755076934920973937908',
  's': '45155461792159514883067068644058913853180508583163102385805265017506142956847',
  'v': 2709}
```

transaction_raw() → str

Get Ethereum fund transaction raw.

Returns str – Ethereum fund transaction raw.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.providers.ethereum.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
  '0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f', sender_address=
  '0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C', endtime=get_current_
  timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
  '0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C', htlc=htlc, amount=100_000_000)
>>> fund_transaction.transaction_raw()

  'eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE'
  ''
```

8.3.1 NormalTransaction

class swap.providers.ethereum.transaction.**NormalTransaction**(network: str = 'mainnet', erc20: bool = False, provider: str = 'http', token: Optional[str] = None)

Ethereum Normal transaction.

Parameters

- **network** (str) – Ethereum network, defaults to mainnet.
- **erc20** (bool) – Normal transaction ERC20 token, default to False.
- **provider** (str) – Ethereum network provider, defaults to http.
- **token** (str) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns NormalTransaction – Ethereum normal transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(*address: str, recipient: dict, token_address: Optional[str] = None, unit: str = 'Wei'*) → *swap.providers.ethereum.transaction.NormalTransaction*

Build Ethereum normal transaction.

Parameters

- **address** (*str*) – Ethereum sender address.
- **recipient** (*dict*) – Recipients Ethereum address and amount.
- **token_address** (*bool*) – Ethereum ERC20 token address, default to None.
- **unit** (*str*) – Ethereum unit, default to Wei.

Returns *NormalTransaction* – Ethereum normal transaction instance.

```
>>> from swap.providers.ethereum.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", recipient={
↳ "0x1954C47a5D75bdDA53578CEe5D549bf84b8c6B94": 100_000_000})
<swap.providers.ethereum.transaction.FundTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.ethereum.solver.NormalSolver*) →

swap.providers.ethereum.transaction.NormalTransaction

Sign Ethereum normal transaction.

Parameters **solver** (*ethereum.solver.NormalSolver*) – Ethereum normal solver.

Returns *NormalTransaction* – Ethereum normal transaction instance.

```
>>> from swap.providers.ethereum.transaction import NormalTransaction
>>> from swap.providers.ethereum.solver import NormalSolver
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", recipient={
↳ "0x1954C47a5D75bdDA53578CEe5D549bf84b8c6B94": 100_000_000})
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUUbHly3kQf
↳ ", address=0)
>>> normal_transaction.sign(solver=normal_solver)
<swap.providers.ethereum.transaction.FundTransaction object at 0x0409DAF0>
```

8.3.2 FundTransaction

class *swap.providers.ethereum.transaction.FundTransaction*(*network: str = 'mainnet', erc20: bool = False, provider: str = 'http', token: Optional[str] = None*)

Ethereum Fund transaction.

Parameters

- **network** (*str*) – Ethereum network, defaults to mainnet.
- **erc20** (*bool*) – Fund transaction ERC20 token, default to False.
- **provider** (*str*) – Ethereum network provider, defaults to http.
- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns FundTransaction – Ethereum fund transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(address: str, htlc: swap.providers.ethereum.htlc.HTLC, amount: Union[Wei, int], unit: str = 'Wei') → swap.providers.ethereum.transaction.FundTransaction

Build Ethereum fund transaction.

Parameters

- **address** (str) – Ethereum sender address.
- **htlc** (ethereum.htlc.HTLC) – Ethereum HTLC instance.
- **amount** (Wei, int, float) – Ethereum amount or ERC20 amount.
- **unit** (str) – Ethereum unit, default to Wei.

Returns FundTransaction – Ethereum fund transaction instance.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.providers.ethereum.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(network="mainnet", erc20=False)
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", htlc=htlc, amount=100_000_000)
<swap.providers.ethereum.transaction.FundTransaction object at 0x0409DAF0>
```

sign(solver: swap.providers.ethereum.solver.FundSolver) → swap.providers.ethereum.transaction.FundTransaction

Sign Ethereum fund transaction.

Parameters solver (ethereum.solver.FundSolver) – Ethereum fund solver.

Returns FundTransaction – Ethereum fund transaction instance.

```
>>> from swap.providers.ethereum.htlc import HTLC
>>> from swap.providers.ethereum.transaction import FundTransaction
>>> from swap.providers.ethereum.solver import FundSolver
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "0xd77E0d2Eef905cfB39c3C4b952Ed278d58f96E1f", sender_address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", htlc=htlc, amount=100_000_000)
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWUubHLY3kQf",
↳ address=0)
```

(continues on next page)

(continued from previous page)

```
>>> fund_transaction.sign(solver=fund_solver)
<swap.providers.ethereum.transaction.FundTransaction object at 0x0409DAF0>
```

8.3.3 WithdrawTransaction

```
class swap.providers.ethereum.transaction.WithdrawTransaction(network: str = 'mainnet', erc20:
                                                                bool = False, provider: str = 'http',
                                                                token: Optional[str] = None)
```

Ethereum Withdraw transaction.

Parameters

- **network** (*str*) – Ethereum network, defaults to mainnet.
- **erc20** (*bool*) – Withdraw transaction ERC20 token, default to False.
- **provider** (*str*) – Ethereum network provider, defaults to http.
- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns WithdrawTransaction – Ethereum withdraw transaction instance.

Warning: Do not forget to build transaction after initialize withdraw transaction.

```
build_transaction(transaction_hash: str, address: str, secret_key: str, contract_address: Optional[str] =
None) → swap.providers.ethereum.transaction.WithdrawTransaction
```

Build Ethereum withdraw transaction.

Parameters

- **transaction_hash** (*str*) – Ethereum HTLC funded transaction hash.
- **address** (*str*) – Ethereum recipient address.
- **secret_key** (*str*) – Secret password/passphrase.
- **contract_address** (*str*) – Ethereum HTLC contract address, defaults to None.

Returns WithdrawTransaction – Ethereum withdraw transaction instance.

```
>>> from swap.providers.ethereum.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", secret_
↳ key="Hello Meheret!", address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C",
↳ contract_address="0x67324d402ffc103d061dAfa9096ff639f0676378")
<swap.providers.ethereum.transaction.WithdrawTransaction object at 0x0409DAF0>
```

```
sign(solver: swap.providers.ethereum.solver.WithdrawSolver) →
swap.providers.ethereum.transaction.WithdrawTransaction
Sign Ethereum withdraw transaction.
```

Parameters **solver** (`ethereum.solver.WithdrawSolver`) – Ethereum withdraw solver.

Returns WithdrawTransaction – Ethereum withdraw transaction instance.

```

>>> from swap.providers.ethereum.transaction import WithdrawTransaction
>>> from swap.providers.ethereum.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", secret_
↳ key="Hello Meheret!", address="0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C",
↳ contract_address="0x67324d402ffc103d061dAfa9096ff639f0676378")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ", address=1)
>>> withdraw_transaction.sign(solver=withdraw_solver)
<swap.providers.ethereum.transaction.WithdrawTransaction object at 0x0409DAF0>

```

8.3.4 RefundTransaction

class swap.providers.ethereum.transaction.**RefundTransaction**(network: str = 'mainnet', erc20: bool = False, provider: str = 'http', token: Optional[str] = None)

Ethereum Refund transaction.

Parameters

- **network** (str) – Ethereum network, defaults to mainnet.
- **erc20** (bool) – Refund transaction ERC20 token, default to False.
- **provider** (str) – Ethereum network provider, defaults to http.
- **token** (str) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns RefundTransaction – Ethereum refund transaction instance.

Warning: Do not forget to build transaction after initialize refund transaction.

build_transaction(transaction_hash: str, address: str, contract_address: Optional[str] = None) → swap.providers.ethereum.transaction.RefundTransaction

Build Ethereum refund transaction.

Parameters

- **transaction_hash** (str) – Ethereum HTLC funded transaction hash.
- **address** (str) – Ethereum sender address.
- **contract_address** (str) – Ethereum HTLC contract address, defaults to None.

Returns RefundTransaction – Ethereum refund transaction instance.

```

>>> from swap.providers.ethereum.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")

```

(continues on next page)

(continued from previous page)

```
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", contract_address=
↳ "0x67324d402ffc103d061dAfa9096ff639f0676378")
<swap.providers.ethereum.transaction.RefundTransaction object at 0x0409DAF0>
```

sign(solver: swap.providers.ethereum.solver.RefundSolver) →
 swap.providers.ethereum.transaction.RefundTransaction
 Sign Ethereum refund transaction.

Parameters **solver** (ethereum.solver.RefundSolver) – Ethereum refund solver.

Returns RefundTransaction – Ethereum refund transaction instance.

```
>>> from swap.providers.ethereum.transaction import RefundTransaction
>>> from swap.providers.ethereum.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0xe49ff507739f8d916ae2c9fd51dd63764658ffa42a5288a49d93bc70a933edc4", address=
↳ "0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C", contract_address=
↳ "0x67324d402ffc103d061dAfa9096ff639f0676378")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHLY3kQf
↳ ", address=0)
>>> refund_transaction.sign(solver=refund_solver)
<swap.providers.ethereum.transaction.RefundTransaction object at 0x0409DAF0>
```

8.4 Solver

Ethereum solver.

8.4.1 NormalSolver

class swap.providers.ethereum.solver.**NormalSolver**(xprivate_key: str, account: int = 0, change: bool = False, address: int = 0, path: Optional[str] = None, strict: bool = True)

Ethereum Normal solver.

Parameters

- **xprivate_key** (str) – Ethereum sender xprivate key.
- **account** (int) – Ethereum derivation account, defaults to 0.
- **change** (bool) – Ethereum derivation change, defaults to False.
- **address** (int) – Ethereum derivation address, defaults to 0.
- **path** (str) – Ethereum derivation path, defaults to None.
- **strict** (bool) – Strict for must be root xprivate key, default to True.

Returns NormalSolver – Ethereum normal solver instance.

```
>>> from swap.providers.ethereum.solver import NormalSolver
>>> sender_root_xprivate_key: str =
↪ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB
↪ "
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_root_xprivate_
↪ key)
<swap.providers.ethereum.solver.FundSolver object at 0x03FCCA60>
```

8.4.2 FundSolver

```
class swap.providers.ethereum.solver.FundSolver(xprivate_key: str, account: int = 0, change: bool =
False, address: int = 0, path: Optional[str] = None,
strict: bool = True)
```

Ethereum Fund solver.

Parameters

- **xprivate_key** (*str*) – Ethereum sender xprivate key.
- **account** (*int*) – Ethereum derivation account, defaults to 0.
- **change** (*bool*) – Ethereum derivation change, defaults to False.
- **address** (*int*) – Ethereum derivation address, defaults to 0.
- **path** (*str*) – Ethereum derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns FundSolver – Ethereum fund solver instance.

```
>>> from swap.providers.ethereum.solver import FundSolver
>>> sender_root_xprivate_key: str =
↪ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB
↪ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=sender_root_xprivate_key,
↪ path="m/44'/60'/0'/0/0")
<swap.providers.ethereum.solver.FundSolver object at 0x03FCCA60>
```

8.4.3 WithdrawSolver

```
class swap.providers.ethereum.solver.WithdrawSolver(xprivate_key: str, account: int = 0, change: bool
= False, address: int = 0, path: Optional[str] =
None, strict: bool = True)
```

Ethereum Withdraw solver.

Parameters

- **xprivate_key** (*str*) – Ethereum sender xprivate key.
- **account** (*int*) – Ethereum derivation account, defaults to 0.
- **change** (*bool*) – Ethereum derivation change, defaults to False.
- **address** (*int*) – Ethereum derivation address, defaults to 0.
- **path** (*str*) – Ethereum derivation path, defaults to None.

- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns WithdrawSolver – Ethereum withdraw solver instance.

```
>>> from swap.providers.ethereum.solver import WithdrawSolver
>>> recipient_root_xprivate_key: str =
↪ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB
↪ "
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=recipient_root_
↪ xprivate_key)
<swap.providers.ethereum.solver.WithdrawSolver object at 0x03FCCA60>
```

8.4.4 RefundSolver

```
class swap.providers.ethereum.solver.RefundSolver(xprivate_key: str, account: int = 0, change: bool =
False, address: int = 0, path: Optional[str] =
None, strict: bool = True)
```

Ethereum Refund solver.

Parameters

- **xprivate_key** (*str*) – Ethereum sender xprivate key.
- **account** (*int*) – Ethereum derivation account, defaults to 0.
- **change** (*bool*) – Ethereum derivation change, defaults to False.
- **address** (*int*) – Ethereum derivation address, defaults to 0.
- **path** (*str*) – Ethereum derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns RefundSolver – Ethereum refund solver instance.

```
>>> from swap.providers.ethereum.solver import RefundSolver
>>> sender_root_xprivate_key: str =
↪ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB
↪ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=sender_root_xprivate_
↪ key)
<swap.providers.ethereum.solver.RefundSolver object at 0x03FCCA60>
```

8.5 Signature

Ethereum signature.

```
class swap.providers.ethereum.signature.Signature(network: str = 'mainnet', erc20: bool = False,
provider: str = 'http', token: Optional[str] = None)
```

Ethereum Signature.

Parameters

- **network** (*str*) – Ethereum network, defaults to mainnet.
- **erc20** (*bool*) – Signature ERC20 token, default to False.
- **provider** (*str*) – Ethereum network provider, defaults to http.

- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns Signature – Ethereum signature instance.

Note: Ethereum has only five networks, mainnet, ropsten, kovan, rinkeby and testnet.

fee(*unit: str = 'Wei'*) → Union[Wei, int, float]

Get Ethereum signature fee.

Parameters **unit** (*str*) – Ethereum unit, default to Wie.

Returns Wei, int, float – Ethereum signature fee.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
>>> signature.fee(unit="Wei")
1532774
```

hash() → Optional[str]

Get Ethereum signature has.

Returns str – Ethereum signature hash.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
>>> signature.hash()
"0xe87b1aefec9fecbb7699e16d101e757e4825db157eb94d2e71ecfaf17fd3d75d"
```

json() → dict

Get Ethereum signature json.

Returns dict – Ethereum signature json.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
```

(continues on next page)

(continued from previous page)

```
>>> signature.json()
{'chainId': 1337, 'from': '0x69e04fe16c9A6A83076B3c2dc4b4Bc21b5d9A20C', 'value': 3000000000000000000, 'nonce': 1, 'gas': 138448, 'gasPrice': 20000000000, 'to': '0xeaEaC81da5E386E8Ca4De1e64d40a10E468A5b40', 'data': '0xf4fd30623a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb00000000000000000000'}
>>>
```

raw() → Optional[str]
Get Ethereum signature raw.

Returns str – Ethereum signature raw.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWXUbHLy3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiaY2hhaW5JZCI6IDZzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
>>> signature.raw()
↳ "0xf8f4018504a817c80083021cd094eaeac81da5e386e8ca4de1e64d40a10e468a5b408829a2241af62c0000b88
↳ "
```

type() → str
Get Ethereum signature type.

Returns str – Ethereum signature type.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHLy3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsia2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
>>> signature.type()
"ethereum_fund_signed"
```

```
sign(transaction_raw: str, solver: Union[swap.providers.ethereum.solver.NormalSolver,
swap.providers.ethereum.solver.FundSolver, swap.providers.ethereum.solver.WithdrawSolver,
swap.providers.ethereum.solver.RefundSolver]) →
Union[swap.providers.ethereum.signature.NormalSignature,
swap.providers.ethereum.signature.FundSignature,
swap.providers.ethereum.signature.WithdrawSignature,
swap.providers.ethereum.signature.RefundSignature]
Sign Ethereum unsigned transaction raw.
```

Parameters

- **transaction_raw**(str) – Ethereum unsigned transaction raw.

- **solver** (ethereum.solver.NormalSolver, ethereum.solver.FundSolver, ethereum.solver.WithdrawSolver, ethereum.solver.RefundSolver) – Ethereum solver.

Returns NormalSignature, FundSignature, WithdrawSignature, RefundSignature – Ethereum signature instance.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
<swap.providers.ethereum.signature.FundSignature object at 0x0409DAF0>
```

signature() → dict
Get Ethereum signature.

Returns dict – Ethereum signature.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
>>> signature.signature()
{'hash': '0xe87b1aefec9fecbb7699e16d101e757e4825db157eb94d2e71ecfaf17fd3d75d',
 'rawTransaction':
↳ '0xf8f4018504a817c80083021cd094eaeac81da5e386e8ca4de1e64d40a10e468a5b408829a2241af62c0000b88
↳ ', 'r':
↳ 49669210517760089961057755545670916457545361634072315135726343721882166945149,
↳ 's':
↳ 39373327445767756604614462296774202164268870502915897592346222361951457550066,
↳ 'v': 2709}
```

transaction_raw() → str
Get Ethereum signed transaction raw.

Returns str – Ethereum signed transaction raw.

```
>>> from swap.providers.ethereum.signature import Signature
>>> from swap.providers.ethereum.solver import FundSolver
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
```

(continues on next page)

(continued from previous page)

```
>>> signature.transaction_raw()
↪ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE"
↪ "
```

8.5.1 NormalSignature

```
class swap.providers.ethereum.signature.NormalSignature(network: str = 'mainnet', erc20: bool =
False, provider: str = 'http', token:
Optional[str] = None)
```

Ethereum Normal signature.

Parameters

- **network** (*str*) – Ethereum network, defaults to mainnet.
- **erc20** (*bool*) – Normal signature ERC20 token, default to False.
- **provider** (*str*) – Ethereum network provider, defaults to http.
- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns NormalSignature – Ethereum normal signature instance.

Note: Ethereum has only five networks, mainnet, ropsten, kovan, rinkeby and testnet.

```
sign(transaction_raw: str, solver: swap.providers.ethereum.solver.NormalSolver) →
swap.providers.ethereum.signature.NormalSignature
Sign Ethereum unsigned normal transaction raw.
```

Parameters

- **transaction_raw** (*str*) – Ethereum unsigned normal transaction raw.
- **solver** (*ethereum.solver.NormalSolver*) – Ethereum solver.

Returns NormalSignature – Ethereum normal signature instance.

```
>>> from swap.providers.ethereum.signature import NormalSignature
>>> from swap.providers.ethereum.solver import NormalSolver
>>> normal_signature: NormalSignature = NormalSignature(network="mainnet")
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=
↪ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWXUbHly3kQf"
↪ ")
>>> normal_signature.sign(transaction_raw=
↪ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE"
↪ ", solver=normal_solver)
<swap.providers.ethereum.signature.NormalSignature object at 0x0409DAF0>
```

8.5.2 FundSignature

```
class swap.providers.ethereum.signature.FundSignature(network: str = 'mainnet', erc20: bool = False,
                                                    provider: str = 'http', token: Optional[str] =
                                                    None)
```

Ethereum Fund signature.

Parameters

- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **erc20** (*bool*) – Fund signature ERC20 token, default to `False`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns `FundSignature` – Ethereum fund signature instance.

Note: Ethereum has only five networks, `mainnet`, `ropsten`, `kovan`, `rinkeby` and `testnet`.

```
sign(transaction_raw: str, solver: swap.providers.ethereum.solver.FundSolver) →
    swap.providers.ethereum.signature.FundSignature
Sign Ethereum unsigned fund transaction raw.
```

Parameters

- **transaction_raw** (*str*) – Ethereum unsigned fund transaction raw.
- **solver** (`ethereum.solver.FundSolver`) – Ethereum solver.

Returns `FundSignature` – Ethereum fund signature instance.

```
>>> from swap.providers.ethereum.signature import FundSignature
>>> from swap.providers.ethereum.solver import FundSolver
>>> fund_signature: FundSignature = FundSignature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> fund_signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiaY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=fund_solver)
<swap.providers.ethereum.signature.FundSignature object at 0x0409DAF0>
```

8.5.3 WithdrawSignature

```
class swap.providers.ethereum.signature.WithdrawSignature(network: str = 'mainnet', erc20: bool =
                                                         False, provider: str = 'http', token:
                                                         Optional[str] = None)
```

Ethereum Withdraw signature.

Parameters

- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **erc20** (*bool*) – Withdraw signature ERC20 token, default to `False`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.

- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns WithdrawSignature – Ethereum withdraw signature instance.

Note: Ethereum has only five networks, mainnet, ropsten, kovan, rinkeby and testnet.

sign(*transaction_raw: str, solver: swap.providers.ethereum.solver.WithdrawSolver*) → *swap.providers.ethereum.signature.WithdrawSignature*
Sign Ethereum unsigned withdraw transaction raw.

Parameters

- **transaction_raw** (*str*) – Ethereum unsigned withdraw transaction raw.
- **solver** (*ethereum.solver.WithdrawSolver*) – Ethereum withdraw solver.

Returns WithdrawSignature – Ethereum withdraw signature instance.

```
>>> from swap.providers.ethereum.signature import WithdrawSignature
>>> from swap.providers.ethereum.solver import WithdrawSolver
>>> withdraw_signature: WithdrawSignature = WithdrawSignature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHly3kQf
↳ ")
>>> withdraw_signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=withdraw_solver)
<swap.providers.ethereum.signature.WithdrawSignature object at 0x0409DAF0>
```

8.5.4 RefundSignature

class swap.providers.ethereum.signature.**RefundSignature**(*network: str = 'mainnet', erc20: bool = False, provider: str = 'http', token: Optional[str] = None*)

Ethereum Refund signature.

Parameters

- **network** (*str*) – Ethereum network, defaults to mainnet.
- **erc20** (*bool*) – Refund signature ERC20 token, default to False.
- **provider** (*str*) – Ethereum network provider, defaults to http.
- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns RefundSignature – Ethereum refund signature instance.

Note: Ethereum has only five networks, mainnet, ropsten, kovan, rinkeby and testnet.

sign(*transaction_raw: str, solver: swap.providers.ethereum.solver.RefundSolver*) → *swap.providers.ethereum.signature.RefundSignature*
Sign Ethereum unsigned refund transaction raw.

Parameters

- **transaction_raw** (*str*) – Ethereum unsigned refund transaction raw.
- **solver** (`ethereum.solver.RefundSolver`) – Ethereum refund solver.

Returns `RefundSignature` – Ethereum refund signature instance.

```
>>> from swap.providers.ethereum.signature import RefundSignature
>>> from swap.providers.ethereum.solver import RefundSolver
>>> refund_signature: RefundSignature = RefundSignature(network="mainnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWXUbHly3kQf
↳ ")
>>> refund_signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=refund_solver)
<swap.providers.ethereum.signature.RefundSignature object at 0x0409DAF0>
```

8.6 Remote Procedure Call (RPC)

Ethereum remote procedure call.

`swap.providers.ethereum.rpc.get_web3` (*network: str = 'mainnet', provider: str = 'http', token: Optional[str] = None*) → `web3.main.Web3`

Get Ethereum Web3 instance.

Parameters

- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns `Web3` – Ethereum Web3 instance.

```
>>> from swap.providers.ethereum.rpc import get_web3
>>> get_web3(network="testnet", provider="http", token="infura endpoint token ...")
<web3.main.Web3 object at 0x0000001DDECCD0640>
```

`swap.providers.ethereum.rpc.get_balance` (*address: str, network: str = 'mainnet', provider: str = 'http', token: Optional[str] = None*) → `Wei`

Get Ethereum balance.

Parameters

- **address** (*str*) – Ethereum address.
- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns `Wei` – Ethereum balance (Wei).

```
>>> from swap.providers.ethereum.rpc import get_balance
>>> get_balance(address="0xbaF2Fc3829B6D25739BeDC18a5A83bF519c6Fe8c", network=
↳ "testnet")
9993791576000000000000
```

```
swap.providers.ethereum.rpc.get_erc20_balance(address: str, token_address: str, network: str =
'mainnet', provider: str = 'http', token: Optional[str] =
None) → Tuple[int, str, str, int, str]
```

Get Ethereum ERC20 token balance.

Parameters

- **address** (*str*) – Ethereum address.
- **token_address** (*str*) – Ethereum ERC20 token address.
- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns tuple – Ethereum ERC20 token balance and decimals.

[illegible]

```
swap.providers.ethereum.rpc.get_erc20_decimals(token_address: str, network: str = 'mainnet', provider: str = 'http', token: Optional[str] = None) → int
```

Get Ethereum ERC20 token decimals.

Parameters

- **token_address** (*str*) – Ethereum ERC20 token address.
- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns int – Ethereum ERC20 token decimals.

```
>>> from swap.providers.ethereum.rpc import get_erc20_decimals
>>> get_erc20_decimals(token_address="0xDA6844e863bdfEE6AaFf888D2D34Bf1B7c37861",
↳ network="testnet")
18
```

```
swap.providers.ethereum.rpc.get_transaction(transaction_hash: str, network: str = 'mainnet', provider:
                                             str = 'http', token: Optional[str] = None) → dict
```

Get Ethereum transaction detail.

Parameters

- **transaction_hash** (*str*) – Ethereum transaction hash/id.
- **network** (*str*) – Ethereum network, defaults to mainnet.

Parameters

- **transaction_hash** (*str*) – Ethereum transaction hash/id.
- **timeout** (*int*) – Request timeout, default to 60.
- **network** (*str*) – Ethereum network, defaults to `mainnet`.
- **provider** (*str*) – Ethereum network provider, defaults to `http`.
- **token** (*str*) – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns dict – Ethereum transaction receipt.

[illegible]

`swap.providers.ethereum.rpc.decode_raw(raw: str) → dict`
Decode original Ethereum raw into blockchain.

Parameters **raw** (*str*) – Ethereum transaction raw.

Returns dict – Ethereum decoded transaction hash.

```
>>> from swap.providers.ethereum.rpc import decode_raw
>>> decode_raw(raw=
↳ "0xf86c02840ee6b280825208943e0a9b2ee8f8341a1ead3e7531d75f1e395f24b8901236efcbcb340000801ba03084
↳ ")
{'hash': '0x04b3bfb804f2b3329555c6f3a17a794b3f099b6435a9cf58c78609ed93853907', 'from
↳ ': '0x3769F63e3b694cD2e973e28af59bdFd751303273', 'to':
↳ '0x3e0a9B2Ee8F8341A1aEaD3E7531d75f1e395F24b', 'nonce': 2, 'gas': 21000, 'gas_price
↳ ': 2500000000, 'value': 2100000000000000000000, 'data': '0x', 'chain_id': -4, 'r':
↳ '0x3084982e4a9dd897d3cc1b2c8cc2d1b106b9d302eb23f6fae7d0e57e53e043f8', 's':
↳ '0x116f13f9ab385f6b53e7821b3335ced924a1ceb88303347cd0af4aa75e6bfb73', 'v': 27}
```

`swap.providers.ethereum.rpc.submit_raw(raw: str, network: str = 'mainnet', provider: str = 'http', token: Optional[str] = None) → str`
Submit original Ethereum raw into blockchain.

Parameters

- **raw** (*str*) – Ethereum transaction raw.
- **network** (*str*) – Ethereum network, defaults to mainnet.
- **provider** (*str*) – Ethereum network provider, defaults to http.

- **token** (*str*) – Infura API endpoint token, defaults to 4414fea5f7454211956b1627621450b4.

Returns *str* – Ethereum submitted transaction hash/id.

```
>>> from swap.providers.ethereum.rpc import submit_raw
>>> submit_raw(raw=
↳ "0xf86c02840ee6b280825208943e0a9b2ee8f8341a1aead3e7531d75f1e395f24b8901236efcbcb340000801ba03084
↳ ", network="testnet")
"0x04b3bfb804f2b3329555c6f3a17a794b3f099b6435a9cf58c78609ed93853907"
```

8.7 Utils

Ethereum Utils.

`swap.providers.ethereum.utils.is_network(network: str) → bool`

Check Ethereum network.

Parameters **network** (*str*) – Ethereum network.

Returns *bool* – Ethereum valid/invalid network.

```
>>> from swap.providers.ethereum.utils import is_network
>>> is_network(network="kovan")
True
```

`swap.providers.ethereum.utils.is_address(address: str) → bool`

Check Ethereum address.

Parameters **address** (*str*) – Ethereum address.

Returns *bool* – Ethereum valid/invalid address.

```
>>> from swap.providers.ethereum.utils import is_address
>>> is_address(address="0x1ee11011ae12103a488a82dc33e03f337bc93ba7")
True
```

`swap.providers.ethereum.utils.is_checksum_address(address: str) → bool`

Check Ethereum checksum address.

Parameters **address** (*str*) – Ethereum address.

Returns *bool* – Ethereum valid/invalid checksum address.

```
>>> from swap.providers.ethereum.utils import is_checksum_address
>>> is_checksum_address(address="0x1ee11011ae12103a488a82dc33e03f337bc93ba7")
False
>>> is_checksum_address(address="0x1Ee11011ae12103a488A82DC33e03f337Bc93ba7")
True
```

`swap.providers.ethereum.utils.to_checksum_address(address: str) → ChecksumAddress`

Change Ethereum address to checksum address.

Parameters **address** (*ChecksumAddress*) – Ethereum address.

Returns *str* – Ethereum checksum address.

```
>>> from swap.providers.ethereum.utils import to_checksum_address
>>> is_checksum_address(address="0x1ee11011ae12103a488a82dc33e03f337bc93ba7")
"0x1Ee11011ae12103a488A82DC33e03f337Bc93ba7"
```

`swap.providers.ethereum.utils.is_transaction_raw(transaction_raw: str) → bool`

Check Ethereum transaction raw.

Parameters `transaction_raw (str)` – Ethereum transaction raw.

Returns `bool` – Ethereum valid/invalid transaction raw.

```
>>> from swap.providers.ethereum.utils import is_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBenAyNGRrZmZlbHlzOHpjd3l1"
↳ ""
>>> is_transaction_raw(transaction_raw=transaction_raw)
True
```

`swap.providers.ethereum.utils.decode_transaction_raw(transaction_raw: str) → dict`

Decode Ethereum transaction raw.

Parameters `transaction_raw (str)` – Ethereum transaction raw.

Returns `dict` – Decoded ethereum transaction raw.

```
>>> from swap.providers.ethereum.utils import decode_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBenAyNGRrZmZlbHlzOHpjd3l1"
↳ ""
>>> decode_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'address': '...', 'transaction': {...}, 'unsigned_data':
↳ [...], 'signatures': [...], 'network': '...'}
```

`swap.providers.ethereum.utils.submit_transaction_raw(transaction_raw: str, provider: str = 'http', token: Optional[str] = None) → dict`

Submit Ethereum transaction raw.

Parameters

- **transaction_raw (str)** – Ethereum transaction raw.
- **provider (str)** – Ethereum network provider, defaults to `http`.
- **token (str)** – Infura API endpoint token, defaults to `4414fea5f7454211956b1627621450b4`.

Returns `dict` – Ethereum submitted transaction id, fee, type and date.

```
>>> from swap.providers.ethereum.utils import submit_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBenAyNGRrZmZlbHlzOHpjd3l1"
↳ ""
>>> submit_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'transaction_id': '...', 'network': '...', 'date': '...'
↳ '}'
```

`swap.providers.ethereum.utils.amount_unit_converter(amount: Union[int, float], unit_from: str = 'Wei2Ether') → Union[int, float]`

Ethereum amount unit converter.

Parameters

- **amount** (*int*, *float*) – Ethereum amount.
- **unit_from** (*str*) – Ethereum unit, default to Wei2Ether

Returns int, float – Ethereum amount.

```
>>> from swap.providers.ethereum.utils import amount_unit_converter
>>> amount_unit_converter(amount=100_000_000, unit_from="Wei2Ether")
0.1
```


VAPOR

Vapor is a layer 2 scalability solution that uses cross-chain technology to interact with the Bytom mainchain. Compared to Ethereum, the Vapor sidechain boasts faster transactions, lower costs, and less risk of congestion.

For more <https://bytom.io/en/sidechain/>

9.1 Wallet

The implementation of Hierarchical Deterministic (HD) wallets generator for Vapor blockchain.

class `swap.providers.vapor.wallet.Wallet`(*network: str = 'mainnet'*)

Vapor Wallet class.

Parameters `network` (*str*) – Vapor network, defaults to mainnet.

Returns `Wallet` – Vapor wallet instance.

Note: Vapor has only two networks, mainnet, solonet and testnet.

from_entropy(*entropy: str, language: str = 'english', passphrase: Optional[str] = None*) →

swap.providers.vapor.wallet.Wallet

Initiate Vapor wallet from entropy.

Parameters

- **entropy** (*str*) – Vapor entropy hex string.
- **language** (*str*) – Vapor wallet language, default to english.
- **passphrase** (*str*) – Vapor wallet passphrase, default to None.

Returns `Wallet` – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_mnemonic(*mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None*) →

swap.providers.vapor.wallet.Wallet

Initialize Vapor wallet from mnemonic.

Parameters

- **mnemonic** (*str*) – Vapor mnemonic words.

- **language** (*str*) – Vapor wallet language, default to english.
- **passphrase** (*str*) – Vapor wallet passphrase, default to None.

Returns Wallet – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park,
↳ clown build renew illness fault")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_seed(*seed: str*) → *swap.providers.vapor.wallet.Wallet*

Initialize Vapor wallet from seed.

Parameters **seed** (*str*) – Vapor Seed hex string.

Returns Wallet – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_seed(seed=
↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ ")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_xprivate_key(*xprivate_key: str*) → *swap.providers.vapor.wallet.Wallet*

Initiate Vapor wallet from xprivate key.

Parameters **xprivate_key** (*str*) – Vapor XPrivate key.

Returns Wallet – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_private_key(*private_key: str*) → *swap.providers.vapor.wallet.Wallet*

Initialize Vapor wallet from private key.

Parameters **private_key** (*str*) – Vapor Private key.

Returns Wallet – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_private_key(private_key=
↳ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512
↳ ")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_path(*path: str*) → *swap.providers.vapor.wallet.Wallet*

Drive Vapor wallet from path.

Parameters **path** (*str*) – Vapor derivation path.

Returns Wallet – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_indexes(*indexes: List[str]*) → *swap.providers.vapor.wallet.Wallet*

Drive Vapor wallet from indexes.

Parameters *indexes* (*list*) – Vapor derivation indexes.

Returns *Wallet* – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> wallet.from_indexes(indexes=["2c000000", "99000000", "01000000", "00000000",
↳ "01000000"])
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

from_index(*index: int, hardened: bool = False*) → *swap.providers.vapor.wallet.Wallet*

Drive Vapor wallet from index.

Parameters

- **index** (*int*) – Vapor wallet index.
- **hardened** (*bool*) – Use hardened, default to False.

Returns *Wallet* – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_index(index=44)
>>> wallet.from_index(index=153)
>>> wallet.from_index(index=1)
>>> wallet.from_index(index=0)
>>> wallet.from_index(index=1)
<swap.providers.vapor.wallet.Wallet object at 0x040DA268>
```

clean_derivation() → *swap.providers.vapor.wallet.Wallet*

Clean derivation Vapor wallet.

Returns *Wallet* – Vapor wallet instance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.indexes()
["2c000000", "99000000", "01000000", "00000000", "01000000"]
>>> wallet.path()
"m/44/153/1/0/1"
>>> wallet.clean_derivation()
```

(continues on next page)

(continued from previous page)

```
>>> wallet.indexes()
[]
>>> wallet.path()
None
```

strength() → Optional[int]
Get Vapor wallet strength.

Returns int – Vapor wallet strength.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.strength()
128
```

entropy() → Optional[str]
Get Vapor wallet entropy.

Returns str – Vapor wallet entropy.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.entropy()
"ed0802d701a033776811601dd6c5c4a9"
```

mnemonic() → Optional[str]
Get Vapor wallet mnemonic.

Returns str – Vapor wallet mnemonic.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.mnemonic()
"unfair divorce remind addict add roof park clown build renew illness fault"
```

passphrase() → Optional[str]
Get Vapor wallet passphrase.

Returns str – Vapor wallet passphrase.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9", passphrase=
↳ "meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]
Get Vapor wallet language.

Returns str – Vapor wallet language.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.language()
"english"
```

seed() → Optional[str]
Get Vapor wallet seed.

Returns str – Vapor wallet seed.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.seed()

↪ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↪ "
```

path() → Optional[str]
Get Vapor wallet derivation path.

Returns str – Vapor derivation path.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet = Wallet(network="mainnet", change=True, address=3)
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.path()
"m/44/153/1/0/1"
```

indexes() → list
Get Vapor wallet derivation indexes.

Returns list – Vapor derivation indexes.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.indexes()
['2c000000', '99000000', '01000000', '00000000', '01000000']
```

xprivate_key() → Optional[str]
Get Vapor wallet xprivate key.

Returns str – Vapor xprivate key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.xprivate_key()

↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↪ "
```

xpublic_key() → Optional[str]
Get Vapor wallet xpublic key.

Returns str – Vapor xpublic key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.xpublic_key()

↪ "f80a401807fde1ee5727ae032ee144e4b757e69431e68e6cd732eda3c8cd3936daedfdd0fd8f8df14e2084c7e8d..."
↪ "
```

expand_xprivate_key() → Optional[str]
Get Vapor wallet expand xprivate key.

Returns str – Vapor expand xprivate key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.expand_xprivate_key()

↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e465c68d75d8a29eb3fffd7e8213808..."
↪ "
```

child_xprivate_key() → Optional[str]
Get Vapor child wallet xprivate key.

Returns str – Vapor child xprivate key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.child_xprivate_key()

↪ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512..."
↪ "
```

child_xpublic_key() → Optional[str]
Get Vapor child wallet xpublic key.

Returns str – Vapor child xpublic key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.child_xpublic_key()

↪ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212db35f71c405fd5948ecffa2c512..."
↪ "
```

guid() → Optional[str]
Get Vapor wallet Blockcenter GUID.

Returns str – Vapor Blockcenter GUID.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.guid()
"9ed61a9b-e7b6-4cb7-94fb-932b738e4f66"
```

private_key() → str

Get Vapor wallet private key.

Returns str – Vapor private key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.private_key()
↪ "b0f9552e4fedac7f2e750ae984e36a97cf2b24609f7ec43f35606ed65eec6e46db35f71c405fd5948ecffa2c512"
↪ "
```

public_key() → str

Get Vapor wallet public key.

Returns str – Vapor public key.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.public_key()
"fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212"
```

program() → str

Get Vapor wallet control program.

Returns str – Vapor control program.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.program()
"0014b1592acbb917f13937166c2a9b6ce973296ebb60"
```

address(network: Optional[str] = None) → str

Get Vapor wallet address.

Parameters **network** (str) – Vapor network, defaults to mainnet.

Returns str – Vapor wallet address.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_indexes(indexes=["2c000000", "99000000", "01000000", "00000000",
↪ "01000000"])
```

(continues on next page)

(continued from previous page)

```
>>> wallet.address(network="mainnet")
"bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx"
```

balance(*asset: Union[str, swap.providers.vapor.assets.AssetNamespace]* = *''*, *unit: str* = 'NEU') → Union[int, float]
Get Vapor wallet balance.

Parameters

- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM asset.
- **unit** (*str*) – Vapor unit, default to NEU.

Returns int, float – Vapor wallet balance.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.balance(unit="BTM")
2.0
```

utxos(*asset: Union[str, swap.providers.vapor.assets.AssetNamespace]* = *''*, *limit: int* = 15) → list
Get Vapor wallet unspent transaction output (UTXO's).

Parameters

- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM asset.
- **limit** (*int*) – Limit of UTXO's, default is 15.

Returns list – Vapor unspent transaction outputs.

```
>>> from swap.providers.vapor.wallet import Wallet
>>> wallet: Wallet = Wallet(network="mainnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44/153/1/0/1")
>>> wallet.utxos()
[{'hash': '9843c9b9130bd87a9683f2c4e66456326beeeb2522c3352326de870c5c1329e',
  ↪ 'asset': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
  ↪ 'amount': 2000000000}]
```

9.2 Hash Time Lock Contract (HTLC)

Vapor Hash Time Lock Contract (HTLC).

class swap.providers.vapor.htlc.**HTLC**(*network: str* = 'mainnet', *contract_address: Optional[str]* = None)
Vapor Hash Time Lock Contract (HTLC).

Parameters **network** (*str*) – Vapor network, defaults to mainnet.

Returns HTLC – Vapor HTLC instance.

Note: Vapor has only three networks, mainnet, solonet and testnet.

build_htlc(*secret_hash: str, recipient_public_key: str, sender_public_key: str, endblock: int, use_script: bool = False*) → *swap.providers.vapor.htlc.HTLC*

Build Vapor Hash Time Lock Contract (HTLC).

Parameters

- **secret_hash** (*str*) – secret sha-256 hash.
- **recipient_public_key** (*str*) – Vapor recipient public key.
- **sender_public_key** (*str*) – Vapor sender public key.
- **endblock** (*int*) – Vapor expiration block height.
- **use_script** (*bool*) – Initialize HTLC by using script, default to False.

Returns HTLC – Vapor Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.rpc import get_current_block_height
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=get_current_block_height(plus=1000), use_script=False)
<swap.providers.vapor.htlc.HTLC object at 0x0409DAF0>
```

from_bytecode(*bytecode: str*) → *swap.providers.vapor.htlc.HTLC*

Initialize Vapor Hash Time Lock Contract (HTLC) from bytecode.

Parameters **bytecode** (*str*) – Vapor bytecode.

Returns HTLC – Vapor Hash Time Lock Contract (HTLC) instance.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ "
>>> htlc.from_bytecode(bytecode=bytecode)
<swap.providers.vapor.htlc.HTLC object at 0x0409DAF0>
```

bytecode() → *str*

Get Vapor Hash Time Lock Contract (HTLC) bytecode.

Returns *str* – Vapor HTLC bytecode.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
```

(continues on next page)

(continued from previous page)

```
>>> htlc.bytecode()
↪ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↪ "
```

opcode() → Optional[str]

Get Vapor Hash Time Lock Contract (HTLC) OP_Code.

Returns str – Vapor HTLC opcode.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=120723497)
>>> htlc.opcode()
"0x29183207 0xfe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212
↪ 0x3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e
↪ 0x3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb DEPTH
↪ 0x547a6416000000557aa888537a7cae7cac631f000000537acd9f6972ae7cac FALSE
↪ CHECKPREDICATE"
```

hash() → str

Get Vapor Hash Time Lock Contract (HTLC) hash.

Returns str – Vapor HTLC hash.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=120723497)
>>> htlc.hash()
"34a3db50301b941b8ed43dcfdbd3381df1b739fa64ab77e4264f703a45e0be31"
```

contract_address() → str

Get Vapor Hash Time Lock Contract (HTLC) address.

Returns str – Vapor HTLC address.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↪ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↪ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↪ endblock=120723497)
>>> htlc.contract_address()
"vp1qxj3ak5psrw2phrk58h8ah5ecrhcmww06vj4h0epxfacr530qhccs4pczgc"
```

balance(asset: Union[str, swap.providers.vapor.assets.AssetNamespace] =
 'ff', unit: str = 'NEU') → Union[int, float]
 Get Vapor HTLC balance.

Parameters

- **asset** (str, vapor.assets.AssetNamespace, vapor.assets.AssetNamespace)
 – Vapor asset id, defaults to BTM.
- **unit** (str) – Vapor unit, default to NEU.

Returns int, float – Vapor HTLC balance.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> htlc.balance(asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
0.1
```

utxos(asset: Union[str, swap.providers.vapor.assets.AssetNamespace] =
 'ff', limit: int = 15) → list
 Get Vapor HTLC unspent transaction output (UTXO's).

Parameters

- **asset** (str, vapor.assets.AssetNamespace) – Vapor asset id, defaults to BTM.
- **limit** (int) – Limit of UTXO's, default is 15.

Returns list – Vapor unspent transaction outputs.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_public_key=
↳ "3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e", sender_
↳ public_key="fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> htlc.utxos(asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
[{'hash': '144dd8355cae0d9aea6ca3fb1ff685fb7b455b1f9cb0c5992c9035844c664ad1',
↳ 'asset': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'amount': 100000000}]
```

9.3 Transaction

Bitcoin transaction in blockchain network.

class swap.providers.vapor.transaction.Transaction(network: str = 'mainnet')
Vapor Transaction.

Parameters network (str) – Vapor network, defaults to mainnet.

Returns Transaction – Vapor transaction instance.

Note: Vapor has only three networks, mainnet, solonet and mainnet.

fee(unit: str = 'NEU') → Union[int, float]

Get Vapor transaction fee.

Parameters unit (str) – Vapor unit, default to NEU.

Returns int, float – Vapor transaction fee.

```
>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.fee(unit="NEU")
509000
```

hash() → str

Get Vapor transaction hash.

returns str – Vapor transaction id/hash.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_transaction.hash()
"a09f3093aaff6c8c8f1a372eac68571ceea4928ccc8b9b54954863758447dec1"
```

json() → dict

Get Vapor transaction json format.

Returns dict – Vapor transaction json format.

```
>>> from swap.providers.vapor.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> refund_transaction.json()
{'tx_id': '6d964222bafb9d6968ee2eed988c837b1da56fcec6fd96329fff8c0d5518f92',
↳ 'version': 1, 'size': 181, 'time_range': 0, 'inputs': [{'type': 'spend',
↳ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'asset_definition': {}, 'amount': 100000000, 'control_program':
↳ '002034a3db50301b941b8ed43dcfdbd3381df1b739fa64ab77e4264f703a45e0be31',
↳ 'address': 'vp1qxj3ak5psrw2phrk58h8ah5ecrhcmww06vj4h0epxfacr530qhccs4pczgc',
↳ 'spent_output_id':
↳ '144dd8355cae0d9aea6ca3fb1ff685fb7b455b1f9cb0c5992c9035844c664ad1', 'input_id
↳ ': '576edbd5cf8682fb82eb8fb61ba3d6f25a9490777be607d2e75b2dbcbcbce89e',
↳ 'witness_arguments': None}], 'outputs': [{'type': 'control', 'id':
↳ 'b6a843f8257fc06ad922a69fa2cfa413277703ffb04512a35799d3c8a2c5d7a2', 'position
↳ ': 0, 'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
↳ definition': {}, 'amount': 9491000, 'control_program':
↳ '0014b1592acbb917f13937166c2a9b6ce973296ebb60', 'address':
↳ 'vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs'}], 'fee': 509000}
```

raw() → str

Get Vapor transaction raw.

Returns str – Vapor transaction raw.

```
>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.raw()
↳ "07010001016b0169df82cf7c7927786a6956937744ee82354c481b0f211ac52a5c1d744c4e3e7866ffffffffffff"
↳ "
```

type() → str

Get Vapor signature transaction type.

Returns str – Vapor signature transaction type.

```
>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
```

(continues on next page)

(continued from previous page)

```
>>> withdraw_transaction.type()
"vapor_withdraw_unsigned"
```

unsigned_datas(*detail: bool = False*) → List[dict]

Get Vapor transaction unsigned datas(messages) with instruction.

Parameters **detail** (*bool*) – Vapor unsigned datas to see detail, defaults to False.

Returns list – Vapor transaction unsigned datas.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_transaction.unsigned_datas()
[{'datas': ['d7107257ef5fbfb04fc4747d6887f230a30676ecd6703a58015878b54f1f7b4f'],
↳ 'public_key':
↳ 'fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212', 'network
↳ ': 'mainnet', 'path': 'm/44/153/1/0/1'}]
```

signatures() → List[List[str]]

Get Vapor transaction signatures(signed datas).

Returns list – Vapor transaction signatures.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> from swap.providers.vapor.solver import FundSolver
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ", path="m/44/153/1/0/1")
>>> fund_transaction.sign(solver=fund_solver)
```

(continues on next page)

(continued from previous page)

```
>>> fund_transaction.signatures()
[[
  ↳ '0d2e4e42fcee863e74195dceab1dfccf368055b171196faa90c53eaa2cea649bb43cc132354edad970b356aae5d
  ↳ ']]
```

9.3.1 NormalTransaction

class `swap.providers.vapor.transaction.NormalTransaction`(*network: str = 'mainnet'*)

Vapor Normal transaction.

Parameters **network** (*str*) – Vapor network, defaults to mainnet.

Returns NormalTransaction – Vapor normal transaction instance.

Warning: Do not forget to build transaction after initialize normal transaction.

build_transaction(*address: str, recipients: dict, asset: Union[str, swap.providers.vapor.assets.AssetNamespace] = 'ff', unit: str = 'NEU'*) → *swap.providers.vapor.transaction.NormalTransaction*

Build Vapor normal transaction.

Parameters

- **address** (*str*) – Vapor sender wallet address.
- **recipients** (*dict*) – Recipients Vapor address and amount.
- **asset** (*str, vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM.
- **unit** (*str*) – Vapor unit, default to NEU.

Returns NormalTransaction – Vapor normal transaction instance.

```
>>> from swap.providers.vapor.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="mainnet")
>>> normal_transaction.build_transaction(address=
  ↳ "vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag", recipients={
  ↳ "vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37": 10000000},
  ↳ asset="ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.vapor.transaction.NormalTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.vapor.solver.NormalSolver*) →

swap.providers.vapor.transaction.NormalTransaction

Sign Vapor normal transaction.

Parameters **solver** (*vapor.solver.NormalSolver*) – Vapor normal solver.

Returns NormalTransaction – Vapor normal transaction instance.

```
>>> from swap.providers.vapor.transaction import NormalTransaction
>>> from swap.providers.vapor.solver import NormalSolver
>>> from swap.providers.vapor.wallet import Wallet, DEFAULT_PATH
>>> sender_wallet: Wallet = Wallet(network="mainnet").from_entropy(entropy=
  ↳ "72fee73846f2d1a5807dc8c953bf79f1").from_path(path=DEFAULT_PATH)
```

(continues on next page)

(continued from previous page)

```

>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_wallet.
↳ xprivate_key())
>>> normal_transaction: NormalTransaction = NormalTransaction(network="mainnet")
>>> normal_transaction.build_transaction(address=sender_wallet.address(),
↳ recipients={"vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37":
↳ 100000000}, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> normal_transaction.sign(solver=normal_solver)
<swap.providers.vapor.transaction.NormalTransaction object at 0x0409DAF0>

```

transaction_raw() → str

Get Vapor normal transaction raw.

Returns str – Vapor normal transaction raw.

```

>>> from swap.providers.vapor.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="mainnet")
>>> normal_transaction.build_transaction(address=
↳ "vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag", recipients={
↳ "vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37": 100000000},
↳ asset="ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> normal_transaction.transaction_raw()

↳ "eyJmZWUiOiA2NzgsICJyYXciOiAiMDIwMDAwMDAwMTJjMzkyMjE3NDgzOTA2ZjkwMmU3M2M0YmMxMzI4NjRkZTU4MTU"
↳ ""

```

9.3.2 FundTransaction

class swap.providers.vapor.transaction.**FundTransaction**(network: str = 'mainnet')

Vapor Fund transaction.

Parameters **network** (str) – Vapor network, defaults to mainnet.

Returns FundTransaction – Vapor fund transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(address: str, htlc: swap.providers.vapor.htlc.HTLC, amount: Union[int, float], asset: Union[str, swap.providers.vapor.assets.AssetNamespace] = 'AA', unit: str = 'NEU') → swap.providers.vapor.transaction.FundTransaction

Build Vapor fund transaction.

Parameters

- **address** (str) – Vapor sender wallet address.
- **htlc** (str) – Vapor Hash Time Lock Contract (HTLC) instance.
- **amount** (int, float) – Vapor amount to fund.
- **asset** (str, vapor.assets.AssetNamespace) – Vapor asset id, defaults to BTM.
- **unit** (str) – Vapor unit, default to NEU.

Returns FundTransaction – Vapor fund transaction instance.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
<swap.providers.vapor.transaction.FundTransaction object at 0x0409DAF0>
```

sign(solver: swap.providers.vapor.solver.FundSolver) → swap.providers.vapor.transaction.FundTransaction
Sign Vapor fund transaction.

Parameters solver (vapor.solver.FundSolver) – Vapor fund solver.

Returns FundTransaction – Vapor fund transaction instance.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> from swap.providers.vapor.solver import FundSolver
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d",
↳ path="m/44/153/1/0/1")
>>> fund_transaction.sign(solver=fund_solver)
<swap.providers.vapor.transaction.FundTransaction object at 0x0409DAF0>
```

transaction_raw() → str

Get Vapor fund transaction raw.

Returns str – Vapor fund transaction raw.

```
>>> from swap.providers.vapor.htlc import HTLC
>>> from swap.providers.vapor.transaction import FundTransaction
>>> htlc: HTLC = HTLC(network="mainnet")
>>> htlc.build_htlc(secret_hash=
↳ "3a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb", recipient_
↳ public_key="3e0a377ae4afa031d4551599d9bb7d5b27f4736d77f78cac4d476f0ffba5ae3e",
↳ sender_public_key=
↳ "fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212",
↳ endblock=120723497)
```

(continued from previous page)

```

>>> fund_transaction: FundTransaction = FundTransaction(network="mainnet")
>>> fund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs", htlc=htlc, amount=0.1, asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", unit="BTM
↳ ")
>>> fund_transaction.transaction_raw()

↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djYrYXdtcXdwbmB"
↳ "

```

9.3.3 WithdrawTransaction

class `swap.providers.vapor.transaction.WithdrawTransaction`(*network: str = 'mainnet'*)
 Vapor Withdraw transaction.

Parameters **network** (*str*) – Vapor network, defaults to mainnet.

Returns `WithdrawTransaction` – Vapor withdraw transaction instance.

Warning: Do not forget to build transaction after initialize withdraw transaction.

build_transaction(*address: str, transaction_hash: str, asset: Union[str, swap.providers.vapor.assets.AssetNamespace]* = *'ff'*) → *swap.providers.vapor.transaction.WithdrawTransaction*

Build Vapor withdraw transaction.

Parameters

- **address** (*str*) – Vapor recipient wallet address.
- **transaction_hash** (*str*) – Vapor funded transaction hash/id.
- **asset** (*str, vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM.

Returns `WithdrawTransaction` – Vapor withdraw transaction instance.

```

>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464fff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.vapor.transaction.WithdrawTransaction object at 0x0409DAF0>

```

sign(*solver: swap.providers.vapor.solver.WithdrawSolver*) → *swap.providers.vapor.transaction.WithdrawTransaction*
 Sign Vapor withdraw transaction.

Parameters **solver** (*vapor.solver.WithdrawSolver*) – Vapor withdraw solver.

Returns `WithdrawTransaction` – Vapor withdraw transaction instance.

```

>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> from swap.providers.vapor.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ "", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_transaction.sign(solver=withdraw_solver)
<swap.providers.vapor.transaction.WithdrawTransaction object at 0x0409DAF0>

```

transaction_raw() → str

Get Vapor withdraw transaction raw.

Returns str – Vapor withdraw transaction raw.

```

>>> from swap.providers.vapor.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "mainnet")
>>> withdraw_transaction.build_transaction(address=
↳ "vp1q3plwvmvy4qhjmp5zffzmk50aagpujt6flnf63h", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> withdraw_transaction.transaction_raw()

↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGtd3cwNnZqNGG"
↳ ""

```

9.3.4 RefundTransaction

class swap.providers.vapor.transaction.**RefundTransaction**(network: str = 'mainnet')

Vapor Refund transaction.

Parameters **network** (str) – Vapor network, defaults to mainnet.

Returns RefundTransaction – Vapor refund transaction instance.

Warning: Do not forget to build transaction after initialize refund transaction.

build_transaction(address: str, transaction_hash: str, asset: Union[str, swap.providers.vapor.assets.AssetNamespace] = 'ffffffffffffffffffffffffffffffff') → swap.providers.vapor.transaction.RefundTransaction

Build Vapor refund transaction.

Parameters

- **address** (*str*) – Vapor sender wallet address.
- **transaction_hash** (*str*) – Vapor funded transaction hash/id
- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM.

Returns *RefundTransaction* – Vapor refund transaction instance.

```
>>> from swap.providers.vapor.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdclds4fkm8fwv5kawmqwvnpvs", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
<swap.providers.vapor.transaction.RefundTransaction object at 0x0409DAF0>
```

sign(*solver*: *swap.providers.vapor.solver.RefundSolver*) →
swap.providers.vapor.transaction.RefundTransaction
 Sign Vapor refund transaction.

Parameters *solver* (*vapor.solver.RefundSolver*) – Vapor refund solver.

Returns *RefundTransaction* – Vapor refund transaction instance.

```
>>> from swap.providers.vapor.transaction import RefundTransaction
>>> from swap.providers.vapor.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdclds4fkm8fwv5kawmqwvnpvs", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "58775359b7b3588dcdb1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ "", bytecode=bytecode)
>>> refund_transaction.sign(solver=refund_solver)
<swap.providers.vapor.transaction.RefundTransaction object at 0x0409DAF0>
```

transaction_raw() → *str*
 Get Vapor refund transaction raw.

Returns *str* – Vapor refund transaction raw.

```
>>> from swap.providers.vapor.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="mainnet")
>>> refund_transaction.build_transaction(address=
↳ "vp1qk9vj4jaezlcnjdclds4fkm8fwv5kawmqwvnpvs", transaction_hash=
↳ "37b36d7be5dfda0cc5dc3c918705464ff901dc5eadb6f4f049db03a679e02bfe", asset=
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")
>>> refund_transaction.transaction_raw()
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGNtd3cwNnZqNGG"
↳ ""
```

9.4 Solver

Vapor solver.

9.4.1 NormalSolver

```
class swap.providers.vapor.solver.NormalSolver(xprivate_key: str, account: int = 1, change: bool =
False, address: int = 1, path: Optional[str] = None,
indexes: Optional[List[str]] = None)
```

Vapor Normal solver.

Parameters

- **xprivate_key** (*str*) – Vapor sender xprivate key.
- **account** (*int*) – Vapor derivation account, defaults to 1.
- **change** (*bool*) – Vapor derivation change, defaults to False.
- **address** (*int*) – Vapor derivation address, defaults to 1.
- **path** (*str*) – Vapor derivation path, defaults to None.
- **indexes** (*list*) – Vapor derivation indexes, defaults to None.

Returns NormalSolver – Vapor normal solver instance.

```
>>> from swap.providers.vapor.solver import NormalSolver
>>> sender_xprivate_key: str =
↳ "205b15f70e253399da90b127b074ea02904594be9d54678207872ec1ba31ee51ef4490504bd2b6f99711367189245883
↳ "
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_xprivate_key)
<swap.providers.vapor.solver.NormalSolver object at 0x03FCCA60>
```

9.4.2 FundSolver

```
class swap.providers.vapor.solver.FundSolver(xprivate_key: str, account: int = 1, change: bool = False,
address: int = 1, path: Optional[str] = None, indexes:
Optional[List[str]] = None)
```

Vapor Fund solver.

Parameters

- **xprivate_key** (*str*) – Vapor sender xprivate key.
- **account** (*int*) – Vapor derivation account, defaults to 1.
- **change** (*bool*) – Vapor derivation change, defaults to False.
- **address** (*int*) – Vapor derivation address, defaults to 1.
- **path** (*str*) – Vapor derivation path, defaults to None.
- **indexes** (*list*) – Vapor derivation indexes, defaults to None.

Returns FundSolver – Vapor fund solver instance.

```
>>> from swap.providers.vapor.solver import FundSolver
>>> sender_xprivate_key: str =
↪ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8df4701
↪ "
>>> fund_solver: FundSolver = FundSolver(xprivate_key=sender_xprivate_key)
<swap.providers.vapor.solver.FundSolver object at 0x03FCCA60>
```

9.4.3 WithdrawSolver

```
class swap.providers.vapor.solver.WithdrawSolver(xprivate_key: str, secret_key: str, bytecode: str,
                                                  account: int = 1, change: bool = False, address: int
                                                  = 1, path: Optional[str] = None, indexes:
                                                  Optional[List[str]] = None)
```

Vapor Withdraw solver.

Parameters

- **xprivate_key** (*str*) – Vapor sender xprivate key.
- **secret_key** (*str*) – Secret password/passphrase.
- **bytecode** (*str*) – Vapor witness HTLC bytecode.
- **account** (*int*) – Vapor derivation account, defaults to 1.
- **change** (*bool*) – Vapor derivation change, defaults to False.
- **address** (*int*) – Vapor derivation address, defaults to 1.
- **path** (*str*) – Vapor derivation path, defaults to None.
- **indexes** (*list*) – Vapor derivation indexes, defaults to None.

Returns WithdrawSolver – Vapor withdraw solver instance.

```
>>> from swap.providers.vapor.solver import WithdrawSolver
>>> recipient_xprivate_key: str =
↪ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9f6501
↪ "
>>> bytecode: str =
↪ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa031d4
↪ "
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=recipient_
↪ xprivate_key, secret_key="Hello Meheret!", bytecode=bytecode)
<swap.providers.vapor.solver.WithdrawSolver object at 0x03FCCA60>
```

9.4.4 RefundSolver

```
class swap.providers.vapor.solver.RefundSolver(xprivate_key: str, bytecode: str, account: int = 1,
                                                change: bool = False, address: int = 1, path:
                                                Optional[str] = None, indexes: Optional[List[str]] =
                                                None)
```

Vapor Refund solver.

Parameters

- **xprivate_key** (*str*) – Vapor sender xprivate key.

- **bytecode** (*str*) – Vapor witness HTLC bytecode.
- **account** (*int*) – Vapor derivation account, defaults to 1.
- **change** (*bool*) – Vapor derivation change, defaults to False.
- **address** (*int*) – Vapor derivation address, defaults to 1.
- **path** (*str*) – Vapor derivation path, defaults to None.
- **indexes** (*list*) – Vapor derivation indexes, defaults to None.

Returns RefundSolver – Vapor refund solver instance.

```
>>> from swap.providers.vapor.solver import RefundSolver
>>> sender_xprivate_key: str =
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8df4701
↳ "
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa031d
↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=sender_xprivate_key,
↳ bytecode=bytecode)
<swap.providers.vapor.solver.RefundSolver object at 0x03FCCA60>
```

9.5 Signature

Vapor signature.

class swap.providers.vapor.signature.**Signature**(*network: str = 'mainnet'*)
Vapor Signature.

Parameters **network** (*str*) – Vapor network, defaults to mainnet.

Returns Signature – Vapor signature instance.

Note: Vapor has only three networks, mainnet, solonet and testnet.

fee(*unit: str = 'NEU'*) → Union[int, float]
Get Vapor transaction fee.

Parameters **unit** (*str*) – Vapor unit, default to NEU.

Returns int, float – Vapor transaction fee.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGRrZHM0ZmttOGZ3djVrYXdtcXdwbnB
↳ "
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
```

(continues on next page)

(continued from previous page)

```
>>> signature.fee(unit="NEU")
449000
```

hash() → str

Get Vapor signature transaction hash.

Returns str – Vapor signature transaction hash or transaction id.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogImJtMXF1bDYybnEybDhnbXZ2OWs5dmU0ZTA3bWxtdHhud2d4cHpsZzgz"
↳ ""
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edb9df74545f843c9537259e1d7a656983aef283d0ccebeccc2d33577a9"
↳ ", secret_key="Hello Meheret!", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
>>> signature.hash()
"904aeda199f05cbb7671e0d9ec95b3091f3c131cef8d634ae17216b9c2fea48c"
```

json() → dict

Get Vapor signature transaction json format.

Returns dict – Vapor signature transaction json format.

```
>>> from swap.providers.vapor.solver import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdtcXdwbnB"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dcdc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.json()
{'tx_id': 'a09f3093aaff6c8c8f1a372eac68571ceea4928ccc8b9b54954863758447dec1',
↳ 'version': 1, 'size': 279, 'time_range': 0, 'inputs': [{'type': 'spend',
↳ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'asset_definition': {}, 'amount': 88653000, 'control_program':
↳ '0014b1592acbb917f13937166c2a9b6ce973296ebb60', 'address':
↳ 'vp1qk9vj4jaezlcnjdckds4fkm8fwv5kawmqwvnpvs', 'spent_output_id':
↳ 'baa1fa7702447b83ceea10d075534638b4acd93074bb420d3a5399e35c35c8e9', 'input_id
↳ ': '294506b8df5389141854f6826b625cd7eac43f30fccf6118ae163e34b6b7fc1b',
↳ 'witness_arguments': [
↳ 'fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212']]],
↳ 'outputs': [{'type': 'control', 'id':
↳ '3e7369a5063743ca88961fe5745860c42e3b949c6baa99df08696063e8066996', 'position
↳ ': 0, 'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
↳ definition': {}, 'amount': 10000000, 'control_program':
↳ '002034a3db50301b941b8ed43dcfdbd3381df1b739fa64ab77e4264f703a45e0be31',
↳ 'address': 'vp1qxj3ak5psrw2phrk58h8ah5ecrhcmww06vj4h0epxfacr530qhccs4pczgc'},
↳ {'type': 'control', 'id':
```

(continued from previous page)

raw() → str

Get Vapor signature transaction raw.

Returns str – Vapor signature transaction raw.

```

>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGRrZHM0ZmttOGZ3djVrYXdtcXdwbnB"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.raw()

↳ "07010001015f015ddf82cf7c7927786a6956937744ee82354c481b0f211ac52a5c1d744c4e3e7866ffffffffffff"
↳ ""

```

type() → str

Get Vapor signature transaction type.

Returns str – Vapor signature transaction type.

```

>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGNtd3cwNnZqNGV"
↳ ""
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_refund_transaction_raw,
↳ solver=refund_solver)
>>> signature.type()
"vapor_refund_signed"

```

sign(transaction_raw: str, solver: Union[swap.providers.vapor.solver.NormalSolver, swap.providers.vapor.solver.FundSolver, swap.providers.vapor.solver.WithdrawSolver, swap.providers.vapor.solver.RefundSolver]) → Union[swap.providers.vapor.signature.NormalSignature, swap.providers.vapor.signature.FundSignature, swap.providers.vapor.signature.WithdrawSignature, swap.providers.vapor.signature.RefundSignature]

Sign unsigned transaction raw.

Parameters

- **transaction_raw** (str) – Vapor unsigned transaction raw.

- **solver** (vapor.solver.NormalSolver, vapor.solver.FundSolver, vapor.solver.WithdrawSolver, vapor.solver.RefundSolver) – Vapor solver

Returns NormalSignature, FundSignature, WithdrawSignature, RefundSignature – Vapor signature instance.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdtcXdwbnB"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
<swap.providers.vapor.signature.FundSignature object at 0x0409DAF0>
```

unsigned_datas() → List[dict]

Get Vapor transaction unsigned datas with instruction.

Returns list – Vapor transaction unsigned datas.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGNtd3cwNnZqNGGv"
↳ ""
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ ", secret_key='Hello Meheret!", bytecode=bytecode)
>>> signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
>>> signature.unsigned_datas()
[{'datas': ['3a123fd809d3ad845a92ad3e5a1f0cc103de511adf95cf30240d9164d6ff1964'],
↳ 'network': 'mainnet', 'path': None}]
```

signatures() → List[List[str]]

Get Vapor transaction signatures(signed datas).

Returns list – Vapor transaction signatures.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdtcXdwbnB"
↳ ""
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
```

(continues on next page)

(continued from previous page)

```
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.signatures()
[[
↳ '0d2e4e42fcee863e74195dceab1dfccf368055b171196faa90c53eaa2cea649bb43cc132354edad970b356aae5d
↳ ']]
```

transaction_raw() → str

Get Vapor signed transaction raw.

Returns str – Vapor signed transaction raw.

```
>>> from swap.providers.vapor.signature import Signature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcXdwbnB
↳ "
>>> signature: Signature = Signature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d
↳ ")
>>> signature.sign(transaction_raw=unsigned_fund_transaction_raw, solver=fund_
↳ solver)
>>> signature.transaction_raw()

↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGnrZHM0ZmttOGZ3djVrYXdtcXdwbnB
↳ "
```

9.5.1 NormalSignature

class swap.providers.vapor.signature.**NormalSignature**(network: str = 'mainnet')

Vapor Normal signature.

Parameters **network** (str) – Vapor network, defaults to mainnet.**Returns** NormalSignature – Vapor normal signature instance.**sign**(transaction_raw: str, solver: swap.providers.vapor.solver.NormalSolver) →

swap.providers.vapor.signature.NormalSignature

Sign unsigned normal transaction raw.

Parameters

- **transaction_raw** (str) – Vapor unsigned normal transaction raw.
- **solver** (vapor.solver.NormalSolver) – Vapor normal solver.

Returns NormalSignature – Vapor normal signature instance.

```
>>> from swap.providers.vapor.signature import NormalSignature
>>> from swap.providers.vapor.solver import NormalSolver
>>> unsigned_normal_transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZlJlc3MiOiAiYm0xcTluZlhlseDAyc3lmd2Q3bnBlaGZ4ejRsZGRoenFzdmUyZnU
↳ "
>>> sender_xprivate_key: str =
↳ "205b15f70e253399da90b127b074ea02904594be9d54678207872ec1ba31ee510f4498504bd2b6f997113671892
↳ "
↳ "
↳ "
```

(continues on next page)

(continued from previous page)

```
>>> normal_solver: NormalSolver = NormalSolver(sender_xprivate_key)
>>> normal_signature: NormalSignature = NormalSignature("mainnet")
>>> normal_signature.sign(unsigned_normal_transaction_raw, normal_solver)
<swap.providers.vapor.signature.NormalSignature object at 0x0409DAF0>
```

9.5.2 FundSignature

class swap.providers.vapor.signature.**FundSignature**(network: str = 'mainnet')

Vapor Fund signature.

Parameters **network** (str) – Vapor network, defaults to mainnet.

Returns FundSignature – Vapor fund signature instance.

sign(transaction_raw: str, solver: swap.providers.vapor.solver.FundSolver) →

swap.providers.vapor.signature.FundSignature

Sign unsigned fund transaction raw.

Parameters

- **transaction_raw** (str) – Vapor unsigned fund transaction raw.
- **solver** (vapor.solver.FundSolver) – Vapor fund solver.

Returns FundSignature – Vapor fund signature instance.

```
>>> from swap.providers.vapor.signature import FundSignature
>>> from swap.providers.vapor.solver import FundSolver
>>> unsigned_fund_transaction_raw: str =
↳ "eyJmZWUiOiA0NDkwMDAsICJhZGRyZXNzIjogInZwMXFrOXZqNGphZXpsY25qZGNrZHM0ZmttOGZ3djVrYXdtcXdwbnB"
↳ ""
>>> fund_signature: FundSignature = FundSignature(network="mainnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ ")
>>> fund_signature.sign(transaction_raw=unsigned_fund_transaction_raw,
↳ solver=fund_solver)
<swap.providers.vapor.signature.FundSignature object at 0x0409DAF0>
```

9.5.3 WithdrawSignature

class swap.providers.vapor.signature.**WithdrawSignature**(network: str = 'mainnet')

Vapor Withdraw signature.

Parameters **network** (str) – Vapor network, defaults to mainnet.

Returns WithdrawSignature – Vapor withdraw signature instance.

sign(transaction_raw: str, solver: swap.providers.vapor.solver.WithdrawSolver) →

swap.providers.vapor.signature.WithdrawSignature

Sign unsigned withdraw transaction raw.

Parameters

- **transaction_raw** (str) – Vapor unsigned withdraw transaction raw.

- **solver** (`vapor.solver.WithdrawSolver`) – Vapor withdraw solver.

Returns `WithdrawSignature` – Vapor withdraw signature instance.

```
>>> from swap.providers.vapor.signature import WithdrawSignature
>>> from swap.providers.vapor.solver import WithdrawSolver
>>> unsigned_withdraw_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGNtd3cwNnZqNG9"
↳ ""
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> withdraw_signature: WithdrawSignature = WithdrawSignature(network="mainnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "58dd4094155bbebf2868189231c47e4e0edbd9f74545f843c9537259e1d7a656983aef283d0ccebecc2d33577a9"
↳ "", secret_key="Hello Meheret!", bytecode=bytecode)
>>> withdraw_signature.sign(transaction_raw=unsigned_withdraw_transaction_raw,
↳ solver=withdraw_solver)
<swap.providers.vapor.signature.WithdrawSignature object at 0x0409DAF0>
```

9.5.4 RefundSignature

class `swap.providers.vapor.signature.RefundSignature` (`network: str = 'mainnet'`)

Vapor Refund signature.

Parameters **network** (`str`) – Vapor network, defaults to `mainnet`.

Returns `RefundSignature` – Vapor withdraw signature instance.

sign (`transaction_raw: str, solver: swap.providers.vapor.solver.RefundSolver`) →

`swap.providers.vapor.signature.RefundSignature`

Sign unsigned refund transaction raw.

Parameters

- **transaction_raw** (`str`) – Vapor unsigned refund transaction raw.
- **solver** (`vapor.solver.RefundSolver`) – Vapor refund solver.

Returns `RefundSignature` – Vapor refund signature instance.

```
>>> from swap.providers.vapor.signature import RefundSignature
>>> from swap.providers.vapor.solver import RefundSolver
>>> unsigned_refund_transaction_raw: str =
↳ "eyJmZWUiOiA1MDkwMDAsICJhZGRyZXNzIjogInZwMXF4ajNhazVwc3J3MnBocms1OGg4YWg1ZWNyaGNtd3cwNnZqNG9"
↳ ""
>>> bytecode: str =
↳ "042918320720fe6b3fd4458291b19605d92837ae1060cc0237e68022b2eb9faf01a118226212203e0a377ae4afa"
↳ ""
>>> refund_signature: RefundSignature = RefundSignature(network="mainnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "58775359b7b3588dc1bcf373489fa1272cacc03909f78469657b0208e66e46daedfdd0fd8f8df14e2084c7e8d"
↳ "", bytecode=bytecode)
>>> refund_signature.sign(transaction_raw=unsigned_refund_transaction_raw,
↳ solver=refund_solver)
<swap.providers.vapor.signature.RefundSignature object at 0x0409DAF0>
```

9.6 Remote Procedure Call (RPC)

Vapor remote procedure call.

```
swap.providers.vapor.rpc.get_balance(address: str, asset: Union[str,
swap.providers.vapor.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffff', network: str =
'mainnet', headers: dict = {'accept': 'application/json',
'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap
User-Agent 0.5.0'}, timeout: int = 60) → int
```

Get Vapor balance.

Parameters

- **address** (*str*) – Vapor address.
- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset, default to BTM.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *int* – Vapor asset balance (NEU amount).

```
>>> from swap.providers.vapor.rpc import get_balance
>>> from swap.providers.vapor.assets import BTM as ASSET
>>> get_balance(address="vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag", asset=ASSET,
↪network="mainnet")
970000000
```

```
swap.providers.vapor.rpc.get_utxos(program: str, asset: Union[str,
swap.providers.vapor.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffff', network: str = 'mainnet',
limit: int = 15, by: str = 'amount', order: str = 'desc', headers: dict =
{'accept': 'application/json', 'content-type': 'application/json;
charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int =
60) → list
```

Get Vapor unspent transaction outputs (UTXO's).

Parameters

- **program** (*str*) – Vapor control program.
- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset id, defaults to BTM.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **limit** (*int*) – Vapor utxo's limit, defaults to 15.
- **by** (*str*) – Sort by, defaults to amount.
- **order** (*str*) – Sort order, defaults to desc.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *list* – Vapor unspent transaction outputs (UTXO's).

```
>>> from swap.providers.vapor.rpc import get_utxos
>>> from swap.providers.vapor.assets import BTM as ASSET
>>> get_utxos(program="00142cda4f99ea8112e6fa61cdd26157ed6dc408332a", asset=ASSET,
↳ network="mainnet")
[{'hash': 'e152f88d33c6659ad823d15c5c65b2ed946d207c42430022cba9bb9b9d70a7a4', 'asset':
↳ ': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount':
↳ 587639800}, {'hash':
↳ '88289fa4c7633574931be7ce4102aeb24def0de20e38e7d69a5ddd6efc116b95', 'asset':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount':
↳ 8160000}, {'hash':
↳ 'f71c68f921b434cc2bcd469d26e7927aa6db7500e4cdeef814884f11c10f5de2', 'asset':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount':
↳ 100000}, {'hash': 'e46cfecc1f1a26413172ce81c78affb19408e613915642fa5fb04d3b0a4ffa65
↳ ', 'asset': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'amount': 100}]
```

```
swap.providers.vapor.rpc.estimate_transaction_fee(address: str, amount: int, asset: Union[str,
swap.providers.vapor.assets.AssetNamespace] =
'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
confirmations: int = 1, network: str = 'mainnet',
headers: dict = {'accept': 'application/json',
'content-type': 'application/json; charset=utf-8',
'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int
= 60) → int
```

Estimate Vapor transaction fee.

Parameters

- **address** (*str*) – Vapor address.
- **amount** (*int*) – Vapor amount (NEU amount).
- **asset** (*str*, *vapor.assets.AssetNamespace*) – Vapor asset id, default to BTM.
- **confirmations** (*int*) – Vapor confirmations, default to 1.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – request timeout, default to 60.

Returns *str* – Estimated transaction fee (NEU amount).

```
>>> from swap.providers.vapor.rpc import estimate_transaction_fee
>>> from swap.providers.vapor.assets import BTM as ASSET
>>> estimate_transaction_fee(address="vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag",
↳ asset=ASSET, amount=100_000, confirmations=100, network="mainnet")
449000
```

```
swap.providers.vapor.rpc.account_create(xpublic_key: str, label: str = '1st address', account_index: int =
1, network: str = 'mainnet', headers: dict = {'accept':
'application/json', 'content-type': 'application/json;
charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout:
int = 60) → dict
```

Create account in blockcenter.

Parameters

- **xpublic_key** (*str*) – Bytom xpublic key.
- **label** (*str*) – Bytom limit, defaults to 1st address.
- **account_index** (*str*) – Account index, defaults to 1.
- **network** (*str*) – Bytom network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – request timeout, default to 60.

Returns dict – Bytom blockcenter guid, address and label.

```
>>> from swap.providers.bytom.rpc import account_create
>>> account_create(xpublic_key=
↳ "f80a401807fde1ee5727ae032ee144e4b757e69431e68e6cd732eda3c8cd3936daedfdd0fd8f8df14e2084c7e8df470",
↳ ", network="mainnet")
{"guid": "9ed61a9b-e7b6-4cb7-94fb-932b738e4f66", "address":
↳ "bm1qk9vj4jaezlcnjdckds4fkm8fwv5kawmq9qrufx", "label": "1st address"}
```

`swap.providers.vapor.rpc.build_transaction`(*address: str, transaction: dict, network: str = 'mainnet', headers: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60*) → dict

Build Vapor transaction.

Parameters

- **address** (*str*) – Vapor address.
- **transaction** (*dict*) – Vapor transaction (inputs, outputs, fee, confirmations & forbid_chain_tx).
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Vapor built transaction.

```
>>> from swap.providers.vapor.rpc import build_transaction
>>> build_transaction(address="vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag",
↳ transaction={"fee": "0.1", "confirmations": 1, "inputs": [{"type": "spend_wallet",
↳ "amount": "0.1", "asset":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"}], "outputs": [
↳ {"type": "control_address", "amount": "0.1", "asset":
↳ "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", "address":
↳ "vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37"}]}, network=
↳ "mainnet")
{'tx': {'hash': 'f6b35e2f37862bc9a2cfbc9f21440102599fc5860ed73ba5c3f44e17408e2c8c',
↳ 'status': True, 'size': 279, 'submission_timestamp': 0, 'memo': '', 'inputs': [{
↳ 'script': '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
↳ 'vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag', 'asset': {'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'decimals': 0,
↳ 'unit': 'BTM'}, 'amount': '0.9', 'type': 'spend'}], 'outputs': [{'utxo_id':
↳ '793540933493c531efdc0dfd89d95041badc4e1efaf938d9916cdc7834984c74', 'script':
↳ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'address
↳ ': 'vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37', 'asset': {
↳ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
↳ 'decimals': 0, 'unit': 'BTM'}, 'amount': '0.1', 'type': 'control'}], 'balances': [
↳ '62c391358a7bccac6a3a1b9efd5339eb7207660372290ceb8718af2284467ba0', 'script':
↳ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
↳ 'vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag', 'asset': {'asset_id':
↳ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'decimals': 0,
↳ 'unit': 'BTM'}, 'amount': '0.7', 'type': 'control'}], 'fee': '0.1', 'balances': [
```

(continues on next page)

Get Vapor transaction detail.

- **transaction_hash** (*str*) – Vapor transaction hash/id.
- **network** (*str*) – Vapor network, defaults to `mainnet`.
- **headers** (*dict*) – Request headers, default to `common` headers.
- **timeout** (*int*) – Request timeout, default to `60`.

```
>>> from swap.providers.vapor.rpc import get_transaction
>>> get_transaction(transaction_hash=
→ "4e91bca76db112d3a356c17366df93e364a4922993414225f65390220730d0c1", network=
→ "mainnet")
{'tx_id': '961d984b04214dc202fb40f4c48466d10a2813a138a31e1d2877ad3b6af0ef4c',
→ 'timestamp': 1606993457000, 'block_hash':
→ '440e791390f61c615b974c9292acd1d43bad67368076ef6d86a77cab22f1c2119', 'block_height
→ ': 85098064, 'trx_amount': 0, 'trx_fee': 100000000, 'status_fail': False, 'is_vote
→ ': False, 'is_cross_chain': False, 'coinbase': 0, 'size': 646, 'chain_status':
→ 'mainnet', 'index_id': 18811685, 'mux_id':
→ '97fdbe17d62ae8f8f2024ebc6a231183e8ce7c4e8fde5645b9a3c973f8d0d3ad', 'inputs': [{
→ 'type': 'spend', 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 1
→ 0000, 'control_program':
→ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'address
→ ': 'vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37', 'spent_
→ output_id': 'c30e26caef4ad3436542700c5b32a91cdf0622c60a6c8a6e11cb1c0b250bc65f',
→ 'input_id': 'c470139ab9f9e81829e51096c57365392195ea2e90d7fb19e9eb2b309df22425',
→ 'witness_arguments': [
→ 'db718488496e0823b1cfd9ce64f226ffc4e9debd30eac0b751aa6bd28f694908ae0c0f5d39dd6ed697cae9b0857832f
→ ', '01',
→ '02e8032091ff7f525ff40874c4f47f0cab42e46e3bf53adad59adef9558ad1b6448f22e2203e0a377ae4afa031d4551
→ '], 'decode_program': ['DUP ', 'SHA3 ', 'DATA_32_
→ 4f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'EQUALVERIFY ',
→ 'DATA_8 ffffffffffffffffff', 'SWAP ', 'FALSE ', 'CHECKPREDICATE '], 'decimals': 8,
→ 'unit': 'BTM'}, {'type': 'spend', 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'amount': 1
→ 69900000, 'control_program': '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a',
→ 'address': 'vp1q9ndylx02syfwd7npehfxz4lddhzqsve2za23ag', 'spent_output_id':
→ '1a7f2357f2ec272ea2d96413aee511d2077447731a799110cef97de177739181', 'input_id':
→ '4f50c438b5006eafc547cc48128cb94d2e39430ef30f117aa85e6f30ac92ce09', 'witness_
→ arguments': [
→ 'e31abbf90f8b20cb41f4daedc2f558dedcbc258fcfb9a36aef8c0b4b80f448a78d1d835adb02cc918374c71df8c02c
→ ', '91ff7f525ff40874c4f47f0cab42e46e3bf53adad59adef9558ad1b6448f22e2'], 'decode_
→ program': ['DUP ', 'HASH160 ', 'DATA_20 2cda4f99ea8112e6fa61cdd26157ed6dc408332a',
→ 'EQUALVERIFY ', 'TXSIGHASH ', 'SWAP ', 'CHECKSIG '], 'decimals': 8, 'unit': 'BTM
→ '}], 'outputs': [{'type': 'control', 'id':
→ '20c00b6f9f4fc4f22ccee6c5f8b471a72b1f514f821b1c9c3d1f3243ff011cf1', 'position': 0,
```

(continued from previous page)

```
swap.providers.vapor.rpc.get_current_block_height(plus: int = 0, network: str = 'mainnet', headers:
dict = {'accept': 'application/json', 'content-type':
'application/json; charset=utf-8', 'user-agent':
'Swap User-Agent 0.5.0'}, timeout: int = 60) → int
```

Get Vapor transaction detail.

Parameters

- **plus** (*int*) – Add block number on current block height, default to 0.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *int* – Vapor current block height.

```
>>> from swap.providers.vapor.rpc import get_current_block_height
>>> get_current_block_height(plus=0)
678722
```

```
swap.providers.vapor.rpc.find_p2wsh_utxo(transaction: dict) → Optional[dict]
```

Find Vapor pay to witness script hash UTXO info's.

Parameters *transaction* (*dict*) – Vapor transaction detail.

Returns *dict* – Pay to Witness Script Hash (P2WSH) UTXO info's.

```
>>> from swap.providers.vapor.rpc import find_p2wsh_utxo, get_transaction
>>> find_p2wsh_utxo(transaction=get_transaction(
→ "28168825b2eaded02973313b1c4152a6362157590ec8cd3f530306259eb390ce", "mainnet"))
{'type': 'control', 'id':
→ 'e99f811f25837d0472321e4e237631f40912bf4ca40766a46c8064ccff77d03a', 'position': 0,
→ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
→ 'amount': 10499000, 'control_program':
→ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'address
→ ': 'vp1qf78sazxs539nmzztq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37', 'decimals': 8,
→ 'decode_program': ['DUP ', 'SHA3 ', 'DATA_32_
→ 4f8f0e88d0a44b3d884b07b6dd4536518ffcb596a91ca0e6b2f37e96463bbfc', 'EQUALVERIFY ',
→ 'DATA_8 ffffffffffffffffff', 'SWAP ', 'FALSE ', 'CHECKPREDICATE '], 'unit': 'BTM'}
```

```
swap.providers.vapor.rpc.decode_raw(raw: str, network: str = 'mainnet', headers: dict = {'accept':
'application/json', 'content-type': 'application/json; charset=utf-8',
'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → dict
```

Decode original Vapor raw.

Parameters

- **raw** (*str*) – Vapor transaction raw.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *dict* – Vapor decoded transaction raw.

```
>>> from swap.providers.vapor.rpc import decode_raw
>>> decode_raw(raw=
→ "07010001015f015d0c8382b6aadd32748d0a9490259bf9ba5b55f6ac283535f8752cf5d51621801cfffffffffffffffff
→ ", network="testnet")
{'tx_id': 'f6b35e2f37862bc9a2cfbc9f21440102599fc5860ed73ba5c3f44e17408e2c8c',
→ 'version': 1, 'size': 279, 'time_range': 0, 'inputs': [{'type': 'spend', 'asset_id
→ ': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
→ definition': {}, 'amount': 90000000, 'control_program':
→ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
→ 'vp1q9ndylx02syfwd7npehfxz41ddhzqsve2za23ag', 'spent_output_id':
→ 'f337ffe5333849636e7f6ca01b8a3aa0ef8cc50fadf875730cd40786bb504f80', 'input_id':
→ '437cebc2dbdfff6f5c821fbf6895455192685411bca64f796ff389554e0c23f44', 'witness_
→ arguments': ['91ff7f525ff40874c4f47f0cab42e46e3bf53adad59adef9558ad1b6448f22e2']}
→ ], 'outputs': [{'type': 'control', 'id':
→ '793540933493c531efdc0dfd89d95041badc4e1efaf938d9916cdc7834984c74', 'position': 0,
→ 'asset_id': 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
→ 'asset_definition': {}, 'amount': 10000000, 'control_program':
→ '00204f8f0e88d0a44b3d884b07b6dd4536518ffcbb596a91ca0e6b2f37e96463bbfc', 'address
→ ': 'vp1qf78sazxs539nmzqtq7md63fk2x8lew6ed2gu5rnt9um7jerrh07qcyvk37'}, {'type':
→ 'control', 'id': '62c391358a7bccac6a3a1b9efd5339eb7207660372290ceb8718af2284467ba0
→ ', 'position': 1, 'asset_id':
→ 'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff', 'asset_
→ definition': {}, 'amount': 70000000, 'control_program':
→ '00142cda4f99ea8112e6fa61cdd26157ed6dc408332a', 'address':
→ 'vp1q9ndylx02syfwd7npehfxz41ddhzqsve2za23ag'}], 'fee': 10000000}
```

```
swap.providers.vapor.rpc.submit_raw(address: str, raw: str, signatures: list, network: str = 'mainnet',
→ headers: dict = {'accept': 'application/json', 'content-type':
→ 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent
→ 0.5.0'}, timeout: int = 60) → str
```

Submit original Vapor raw into blockchain.

Parameters

- **address** (*str*) – Vapor address.
- **raw** (*str*) – Vapor transaction raw.
- **signatures** (*list*) – Vapor signed message datas.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns *str* – Vapor submitted transaction id/hash.

```
>>> from swap.providers.vapor.rpc import submit_raw
>>> submit_raw(address="vp1q9ndylx02syfwd7npehfxz41ddhzqsve2za23ag", raw=
→ "07010001015f015d0c8382b6aadd32748d0a9490259bf9ba5b55f6ac283535f8752cf5d51621801cfffffffffffffffff
→ ", signatures=[
→ "31818788bd6cfd255643242212efc1239db8f9dcd91b0e07ef1ddd38d8edf98c420da5578ec195ff7a5ddd72605a1973
→ "]], network="mainnet")
"2993414225f65390220730d0c1a356c14e91bca76db112d37366df93e364a492"
```

9.7 Utils

Vapor Utils.

`swap.providers.vapor.utils.get_address_type(address: str) → Optional[str]`

Get Vapor address type.

Parameters `address (str)` – Vapor address.

Returns `str` – Vapor address type (P2WPKH, P2WSH).

```
>>> from swap.providers.vapor.utils import get_address_type
>>> get_address_type(address="vp1qk9vj4jaez1cnjdckds4fkm8fwv5kawmqwvnpvs")
"p2wpkh"
```

`swap.providers.vapor.utils.is_network(network: str) → bool`

Check Vapor network.

Parameters `network (str)` – Vapor network.

Returns `bool` – Vapor valid/invalid network.

```
>>> from swap.providers.vapor.utils import is_network
>>> is_network(network="solonet")
True
```

`swap.providers.vapor.utils.is_address(address: str, network: Optional[str] = None, address_type: Optional[str] = None) → bool`

Check Vapor address.

Parameters

- **address** (`str`) – Vapor address.
- **network** (`str`) – Vapor network, defaults to `None`.
- **address_type** (`str`) – Vapor address type, defaults to `None`.

Returns `bool` – Vapor valid/invalid address.

```
>>> from swap.providers.vapor.utils import is_address
>>> is_address(address="vp1qk9vj4jaez1cnjdckds4fkm8fwv5kawmqwvnpvs", network=
↳ "mainnet")
True
```

`swap.providers.vapor.utils.is_transaction_raw(transaction_raw: str) → bool`

Check Vapor transaction raw.

Parameters `transaction_raw (str)` – Vapor transaction raw.

Returns `bool` – Vapor valid/invalid transaction raw.

```
>>> from swap.providers.vapor.utils import is_transaction_raw
>>> transaction_raw =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBlbnAyNGRrZmZlbHlzM0hpjd3l1"
↳ ""
>>> is_transaction_raw(transaction_raw=transaction_raw)
True
```

`swap.providers.vapor.utils.amount_unit_converter`(*amount*: Union[int, float], *unit_from*: str = 'NEU2BTM') → Union[int, float]

Vapor amount unit converter

Parameters

- **amount** (*int*, *float*) – Vapor any amount.
- **unit_from** (*str*) – Vapor unit convert from symbol, default to NEU2BTM.

Returns int, float – BTM asset amount.

```
>>> from swap.providers.vapor.utils import amount_unit_converter
>>> amount_unit_converter(amount=10_000_000, unit_from="NEU2BTM")
0.1
```

`swap.providers.vapor.utils.estimate_endblock`(*endtime*: int, *network*: str = 'mainnet', *headers*: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, *timeout*: int = 60) → int

Estimate Vapor expiration block height.

Parameters

- **endtime** (*int*) – Expiration block timestamp.
- **network** (*str*) – Vapor network, defaults to mainnet.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns str – Estimated Vapor endblock.

```
>>> from swap.providers.vapor.utils import estimate_endblock
>>> from swap.utils import get_current_timestamp
>>> estimate_endblock(endtime=get_current_timestamp(plus=3600))
680854
```

`swap.providers.vapor.utils.decode_transaction_raw`(*transaction_raw*: str, *headers*: dict = {'accept': 'application/json', 'content-type': 'application/json; charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, *timeout*: int = 60) → dict

Decode Vapor transaction raw.

Parameters

- **transaction_raw** (*str*) – Vapor transaction raw.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Decoded Vapor transaction raw.

```
>>> from swap.providers.vapor.utils import decode_transaction_raw
>>> transaction_raw =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTbwenAyNGRrZmZlbHlzOHpjd3l1"
↳ ""
>>> decode_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'address': '...', 'transaction': {...}, 'unsigned_datas
↳ ': [...], 'signatures': [...], 'network': '...'}
```

```
swap.providers.vapor.utils.submit_transaction_raw(transaction_raw: str, headers: dict = {'accept':  
                                              'application/json', 'content-type': 'application/json';  
                                              charset=utf-8', 'user-agent': 'Swap User-Agent  
0.5.0'}, timeout: int = 60) → dict
```

Submit Vapor transaction raw.

Parameters

- **transaction_raw** (*str*) – Vapor transaction raw.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 60.

Returns dict – Vapor submitted transaction id, fee, type and date.

```
>>> from swap.providers.vapor.utils import submit_transaction_raw  
>>> transaction_raw =  
↪ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBwenAyNGRrZmZlbHlzOHpjd311"  
↪ "  
>>> submit_transaction_raw(transaction_raw=transaction_raw)  
{'fee': ..., 'type': '...', 'transaction_hash': '...', 'network': '...', 'date': '..  
↪ .'}
```

eXchange inFinite (XinFin), is a Delegated Proof of Stake Consensus network (XDPoS), enabling Hybrid Relay Bridges, Instant Block Finality and Interoperability with ISO20022 messaging standards, making XinFin's Hybrid Architecture Developer friendly.

For more <https://xinfin.org>

10.1 Wallet

The implementation of Hierarchical Deterministic (HD) wallets generator for XinFin blockchain.

class `swap.providers.xinfin.wallet.Wallet`(*network: str = 'mainnet', provider: str = 'http'*)
XinFin Wallet class.

Parameters

- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns `Wallet` – XinFin wallet instance.

Note: XinFin has only two networks, `mainnet`, `apothem` and `testnet`.

from_entropy(*entropy: str, language: str = 'english', passphrase: Optional[str] = None*) →
swap.providers.xinfin.wallet.Wallet

Master from Entropy.

Parameters

- **entropy** (*str*) – XinFin wallet entropy.
- **language** (*str*) – XinFin wallet language, default to `english`.
- **passphrase** (*str*) – XinFin wallet passphrase, default to `None`.

Returns `Wallet` – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_mnemonic(*mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None*) → *swap.providers.xinfin.wallet.Wallet*

Master from Mnemonic.

Parameters

- **mnemonic** (*str*) – XinFin wallet mnemonic.
- **language** (*str*) – XinFin wallet language, default to english.
- **passphrase** (*str*) – XinFin wallet passphrase, default to None.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_mnemonic(mnemonic="unfair divorce remind addict add roof park,
↳ clown build renew illness fault")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_seed(*seed: str*) → *swap.providers.xinfin.wallet.Wallet*

Master from Seed.

Parameters **seed** (*str*) – XinFin wallet seed.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_seed(seed=
↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ ")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_xprivate_key(*xprivate_key: str, strict: bool = True*) → *swap.providers.xinfin.wallet.Wallet*

Master from Root XPrivate Key.

Parameters

- **xprivate_key** (*str*) – XinFin root xprivate key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xprivate_key(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWXUbHly3kQf
↳ ")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_xpublic_key(*xpublic_key: str, strict: bool = True*) → *swap.providers.xinfin.wallet.Wallet*

Master from Root XPublic Key.

Parameters

- **xpublic_key** (*str*) – XinFin root xpublic key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_xpublic_key(xpublic_key=
↳ "xpub661MyMwAqRbcG28HjdHc6zbHxBFzBJBC4ecFvVKBXxiucEBe5wirgQ9hzY2WQMjnurVjJbTjMWRskHi7jnSRkJ
↳ ")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_wif(wif: str) → *swap.providers.xinfin.wallet.Wallet*

Master from Wallet Important Format (WIF).

Parameters **wif** (str) – XinFin wallet important format.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_wif(wif="L1rYHjuxQtgTeU4qMUP6qnGqW9nstFt5drQktRuFGFSuGcCpZoJq")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_private_key(private_key) → *swap.providers.xinfin.wallet.Wallet*

Master from Private Key.

Parameters **private_key** (str) – XinFin private key.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_private_key(private_key=
↳ "8a4bc8131e99a5d1064cdbca6949aa2ec16152967b19f2cee3096daefd5ca857")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_path(path: str) → *swap.providers.xinfin.wallet.Wallet*

Drive XinFin wallet from path.

Parameters **path** (str) – XinFin derivation path.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

from_index(index: int, hardened: bool = False) → *swap.providers.xinfin.wallet.Wallet*

Drive XinFin wallet from index.

Parameters

- **index** (int) – XinFin derivation index.
- **hardened** (bool) – Use hardened index, default to False.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_index(44, harden=True)
>>> wallet.from_index(550, harden=True)
>>> wallet.from_index(0, harden=True)
>>> wallet.from_index(0)
>>> wallet.from_index(0)
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
```

clean_derivation() → *swap.providers.xinfin.wallet.Wallet*

Clean derivation XinFin wallet.

Returns Wallet – XinFin wallet instance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path("m/44'/550'/0'/0/0")
>>> wallet.path()
"m/44'/550'/0'/0/0"
>>> wallet.clean_derivation()
<swap.providers.xinfin.wallet.Wallet object at 0x040DA268>
>>> wallet.path()
None
```

strength() → Optional[int]

Get XinFin wallet strength.

Returns int – XinFin wallet strength.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.strength()
128
```

entropy() → Optional[str]

Get XinFin wallet entropy.

Returns str – XinFin wallet entropy.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.entropy()
"ed0802d701a033776811601dd6c5c4a9"
```

mnemonic() → Optional[str]

Get XinFin wallet mnemonic.

Returns str – XinFin wallet mnemonic.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.mnemonic()
"unfair divorce remind addict add roof park clown build renew illness fault"
```

passphrase() → Optional[str]
Get XinFin wallet passphrase.

Returns str – XinFin wallet passphrase.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9", passphrase=
↳ "meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]
Get XinFin wallet language.

Returns str – XinFin wallet language.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.language()
"english"
```

seed() → Optional[str]
Get XinFin wallet seed.

Returns str – XinFin wallet seed.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.seed()

↳ "1cfd5df8a523d53a36cee369a93fac4e9efab5e4e138d479da2fb6df730697574409d572fe8325ec22e8ed25dea
↳ "
```

root_xprivate_key(encoded: bool = True) → Optional[str]
Get XinFin wallet root xprivate key.

Parameters **encoded** (bool) – Encoded root xprivate key, default to True.

Returns str – XinFin wallet root xprivate key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xprivate_key()

↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWXUbHLy3kQf
↳ "
```

root_xpublic_key(encoded: bool = True) → Optional[str]
Get XinFin wallet root xpublic key.

Parameters `encoded` (*bool*) – Encoded root xprivate key, default to `True`.

Returns `str` – XinFin wallet root xpublic key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.root_xpublic_key()

↪ "xpub661MyMwAqRbcG28HjdHc6zbHxBfzBJBC4ecFvVKBXQxiucEBe5wirgQ9hzY2WQMjnurVjJbTjMWRskHi7jnSRkJ
↪ "
```

xprivate_key(*encoded=True*) → `Optional[str]`

Get XinFin wallet xprivate key.

Parameters `encoded` (*bool*) – Encoded xprivate key, default to `True`.

Returns `str` – XinFin wallet xprivate key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.xprivate_key()

↪ "xprvA3QFrUVTkKpfRhqjgPq897uDFAYtt9VhMdDuZVbPboVf9uPMcMmr7W8sTsrd8nFCsVGSBCpGC3jreRpu8Zs1xsG
↪ "
```

xpublic_key(*encoded: bool = True*) → `Optional[str]`

Get XinFin wallet xpublic key.

Parameters `encoded` (*bool*) – Encoded xprivate key, default to `True`.

Returns `str` – XinFin wallet xpublic key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.xpublic_key()

↪ "xpub6GPcFz2MahNxeBvCnRN8WFqwoCPPHCdYir9WMt11A92e2hiW9u66fJTMKAB81ns7kpAT3vsKi4QHWVSnt7V6crG
↪ "
```

uncompressed() → `str`

Get XinFin wallet uncompressed public key.

Returns `str` – XinFin wallet uncompressed public key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.uncompressed()

↪ "33fbc2f498d145a1827ee894a2ed5f14928523712047ad9fffc59cdda7d314e6707f731cc5b9a5018878fd503
↪ "
```

compressed() → str

Get XinFin wallet compressed public key.

Returns str – XinFin wallet compressed public key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.compressed()
"0333fbc2f498d145a1827ee894a2ed5f14928523712047ad9fffc59cdda7d314e6"
```

chain_code() → str

Get XinFin wallet chain code.

Returns str – XinFin wallet chain code.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.chain_code()
"ba8572f00241c17616903b07fed8ddcc1442677fa54ccd38e85049eee2310246"
```

private_key() → str

Get XinFin wallet private key.

Returns str – XinFin wallet private key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.private_key()
"8a4bc8131e99a5d1064cdbca6949aa2ec16152967b19f2cee3096daefd5ca857"
```

public_key() → str

Get XinFin wallet public key.

Returns str – XinFin wallet public key.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.public_key()
"0333fbc2f498d145a1827ee894a2ed5f14928523712047ad9fffc59cdda7d314e6"
```

path() → Optional[str]

Get XinFin wallet path.

Returns str – XinFin wallet path.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
```

(continues on next page)

(continued from previous page)

```
>>> wallet.path()
"m/44'/550'/0'/0/0"
```

address() → str

Get XinFin wallet address.

Returns str – XinFin wallet address.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.address()
"xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232"
```

wif() → str

Get XinFin wallet important format (WIF).

Returns str – XinFin wallet important format.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.wif()
"L1rYHjuxQtgTeU4qMUP6qnGqW9nstFt5drQktRuFGFSuGcCpZoJq"
```

hash(*private_key: Optional[str] = None*) → str

Get XinFin wallet public key/address hash.

Returns str – XinFin wallet public key/address hash.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.hash()
"dc8f505fccd7cb6f6ba93fd3795174f97efb43ae"
```

balance(*unit: str = 'Wei'*) → Union[Wei, int, float]

Get XinFin wallet balance.

Parameters **unit** (str) – XinFin unit, default to Wei.**Returns** Wei, int, float – XinFin wallet balance.

```
>>> from swap.providers.xinfin.wallet import Wallet
>>> wallet: Wallet = Wallet(network="testnet")
>>> wallet.from_entropy(entropy="ed0802d701a033776811601dd6c5c4a9")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.balance(unit="XDC")
96.96263982
```

xrc20_balance(*token_address: str*) → Tuple[int, str, str, int, str]

Get XinFin HTLC XRC20 balance.

Parameters **token_address** (str) – XinFin XRC20 token address.

Returns tuple – XinFin HTLC XRC20 balance and decimal.

[illegible]

10.2 Hash Time Lock Contract (HTLC)

XinFin Hash Time Lock Contract (HTLC).

```
class swap.providers.xinfin.htlc.HTLC(contract_address: Optional[str] = None, network: str = 'mainnet',
                                       xrc20: bool = False, provider: str = 'http', use_script: bool =
                                       False)
```

XinFin Hash Time Lock Contract (HTLC).

Parameters

- **contract_address** (*str*) – XinFin HTLC contract address, defaults to None.
- **network** (*str*) – XinFin network, defaults to mainnet.
- **xrc20** (*bool*) – HTLC XRC20 token, default to False.
- **provider** (*str*) – XinFin network provider, defaults to http.
- **use_script** (*bool*) – Initialize HTLC by using script, default to False.

Returns HTLC – XinFin HTLC instance.

Note: XinFin has only two networks, mainnet, apothem and testnet.

build_transaction(*address: str*) → *swap.providers.xinfin.htlc.HTLC*
Build XinFin HTLC transaction.

Parameters **address** (*str*) – XinFin address.

Returns HTLC – XinFin HTLC instance.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
↳ ")
<swap.providers.xinfin.htlc.HTLC object at 0x0409DAF0>
```

sign_transaction(private_key: str) → *swap.providers.xinfin.htlc.HTLC*
Sign XinFin HTLC transaction.

Parameters `private_key` (*str*) – XinFin private key.

Returns HTLC – XinFin HTLC instance.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
(continues on next page)
```

(continues on next page)

(continued from previous page)

```
>>> htlc.sign_transaction(private_key=
↳ "8a4bc8131e99a5d1064cdbca6949aa2ec16152967b19f2cee3096daefd5ca857")
<swap.providers.xinfin.htlc.HTLC object at 0x0409DAF0>
```

fee(unit: str = 'Wei') → Union[Wei, int, float]

Get XinFin HTLC transaction fee.

Parameters **unit** (str) – XinFin unit, default to Wei.

Returns Wei, int, float – XinFin transaction fee.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
↳ ")
>>> htlc.fee(unit="Wei")
1532786
```

hash() → Optional[str]

Get XinFin HTLC transaction hash.

Returns str – XinFin transaction hash.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
↳ ")
>>> htlc.sign_transaction(private_key=
↳ "8a4bc8131e99a5d1064cdbca6949aa2ec16152967b19f2cee3096daefd5ca857")
>>> htlc.hash()
"0x2f5a724c9eda4f5ae8fde2d02417f17d9b9c8f5319bb8b79bbb9e5728f3896cc"
```

json() → dict

Get XinFin HTLC transaction json.

Returns dict – XinFin transaction json.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
↳ ")
>>> htlc.json()
{'chainId': 1337, 'from': '0x2224caA2235DF8Da3D2016d2AB1137D2d548A232', 'value': 0, 'nonce': 0, 'gas': 1532786, 'gasPrice': 200000000000, 'data': '0x608060405234801561001057600080fd5b50611ae9806100206000396000f3fe608060405260004361061003f5', 'to': b''}
```

raw() → Optional[str]

Get XinFin HTLC transaction raw.

Returns str – XinFin transaction raw.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(network="testnet")
>>> htlc.build_transaction(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232
↳ ")
```

(continues on next page)

(continued from previous page)

```
>>> htlc.sign_transaction(private_key=
↳ "8a4bc8131e99a5d1064cdbca6949aa2ec16152967b19f2cee3096daefd5ca857")
>>> htlc.raw()
↳ "0xf91b5e808504a817c800831763728080b91b09608060405234801561001057600080fd5b50611ae9806100206"
↳ "
```

contract_address(*prefix: str = 'xdc'*) → Union[str, ChecksumAddress]

Get XinFin HTLC contract address.

Parameters **prefix** (*str*) – XinFin address prefix, default to xdc.

Returns *str*, ChecksumAddress – XinFin HTLC contract address.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.contract_address()
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7"
```

build_htlc(*secret_hash: str, recipient_address: str, sender_address: str, endtime: int, token_address: Optional[str] = None*) → *swap.providers.xinfin.htlc.HTLC*

Build XinFin Hash Time Lock Contract (HTLC).

Parameters

- **secret_hash** (*str*) – Secret sha-256 hash.
- **recipient_address** (*str*) – XinFin recipient address.
- **sender_address** (*str*) – XinFin sender address.
- **endtime** (*int*) – Expiration block time (Seconds).
- **token_address** (*bool*) – XinFin XRC20 token address, default to None.

Returns HTLC – XinFin HTLC instance.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
<swap.providers.xinfin.htlc.HTLC object at 0x0409DAF0>
```

abi() → list

Get XinFin HTLC ABI.

Returns list – XinFin HTLC ABI.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address= (continues on next page)
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
```

(continued from previous page)

```

>>> htlc.abi()
[{'anonymous': False, 'inputs': [{'indexed': True, 'internalType': 'bytes32',
↳ 'name': 'locked_contract_id', 'type': 'bytes32'}, {'indexed': False,
↳ 'internalType': 'bytes32', 'name': 'secret_hash', 'type': 'bytes32'}, {
↳ 'indexed': True, 'internalType': 'address', 'name': 'recipient', 'type':
↳ 'address'}, {'indexed': True, 'internalType': 'address', 'name': 'sender',
↳ 'type': 'address'}, {'indexed': False, 'internalType': 'uint256', 'name':
↳ 'endtime', 'type': 'uint256'}, {'indexed': False, 'internalType': 'uint256',
↳ 'name': 'amount', 'type': 'uint256'}], 'name': 'log_fund', 'type': 'event'}, {
↳ 'anonymous': False, 'inputs': [{'indexed': True, 'internalType': 'bytes32',
↳ 'name': 'locked_contract_id', 'type': 'bytes32'}], 'name': 'log_refund', 'type
↳ ': 'event'}, {'anonymous': False, 'inputs': [{'indexed': True, 'internalType
↳ ': 'bytes32', 'name': 'locked_contract_id', 'type': 'bytes32'}], 'name': 'log_
↳ withdraw', 'type': 'event'}, {'inputs': [{'internalType': 'bytes32', 'name':
↳ '_secret_hash', 'type': 'bytes32'}, {'internalType': 'address payable', 'name
↳ ': '_recipient', 'type': 'address'}, {'internalType': 'address payable', 'name
↳ ': '_sender', 'type': 'address'}, {'internalType': 'uint256', 'name': '_
↳ endtime', 'type': 'uint256'}], 'name': 'fund', 'outputs': [{'internalType':
↳ 'bytes32', 'name': 'locked_contract_id', 'type': 'bytes32'}], 'stateMutability
↳ ': 'payable', 'type': 'function'}, {'inputs': [{'internalType': 'bytes32',
↳ 'name': '_locked_contract_id', 'type': 'bytes32'}], 'name': 'get_locked_
↳ contract', 'outputs': [{'internalType': 'bytes32', 'name': 'id', 'type':
↳ 'bytes32'}, {'internalType': 'bytes32', 'name': 'secret_hash', 'type':
↳ 'bytes32'}, {'internalType': 'address', 'name': 'recipient', 'type': 'address
↳ '}, {'internalType': 'address', 'name': 'sender', 'type': 'address'}, {
↳ 'internalType': 'uint256', 'name': 'endtime', 'type': 'uint256'}, {
↳ 'internalType': 'uint256', 'name': 'amount', 'type': 'uint256'}, {
↳ 'internalType': 'bool', 'name': 'withdrawn', 'type': 'bool'}, {'internalType
↳ ': 'bool', 'name': 'refunded', 'type': 'bool'}, {'internalType': 'string',
↳ 'name': 'preimage', 'type': 'string'}], 'stateMutability': 'view', 'type':
↳ 'function'}, {'inputs': [{'internalType': 'bytes32', 'name': '_locked_
↳ contract_id', 'type': 'bytes32'}], 'name': 'refund', 'outputs': [{
↳ 'internalType': 'bool', 'name': '', 'type': 'bool'}], 'stateMutability':
↳ 'nonpayable', 'type': 'function'}, {'inputs': [{'internalType': 'bytes32',
↳ 'name': '_locked_contract_id', 'type': 'bytes32'}, {'internalType': 'string',
↳ 'name': '_preimage', 'type': 'string'}], 'name': 'withdraw', 'outputs': [{
↳ 'internalType': 'bool', 'name': '', 'type': 'bool'}], 'stateMutability':
↳ 'nonpayable', 'type': 'function'}]]

```

bytecode() → str

Get XinFin HTLC bytecode.

Returns str – XinFin HTLC bytecode.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> htlc.bytecode()

```

(continues on next page)

(continued from previous page)

```

↪ "608060405234801561001057600080fd5b50611ae9806100206000396000f3fe60806040526004361061003f576
↪ "

```

bytecode_runtime() → str

Get XinFin HTLC bytecode runtime.

Returns str – XinFin HTLC bytecode runtime.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↪ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↪ "xdcf8D43806260Cfc6cC79fB408BA1897054667F81C", sender_address=
↪ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↪ timestamp(plus=3600))
>>> htlc.bytecode_runtime()

↪ "60806040526004361061003f5760003560e01c806306a53665146100445780637249fbb614610081578063cfd4b
↪ "

```

opcode() → str

Get XinFin HTLC opcode.

Returns str – XinFin HTLC opcode.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↪ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↪ "xdcf8D43806260Cfc6cC79fB408BA1897054667F81C", sender_address=
↪ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↪ timestamp(plus=3600))
>>> htlc.bytecode_runtime()
"PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0
↪ DUP1 REVERT JUMPDEST POP PUSH2 0x1AE9 DUP1 PUSH2 0x20 PUSH1 0x0 CODECOPY
↪ PUSH1 0x0 RETURN INVALID PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE
↪ LT PUSH2 0x3F JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0xE0 SHR DUP1 PUSH4
↪ 0x6A53665 EQ PUSH2 0x44 JUMPI DUP1 PUSH4 0x7249FBB6 EQ PUSH2 0x81 JUMPI DUP1
↪ PUSH4 0xCFD4B66E EQ PUSH2 0xBE JUMPI DUP1 PUSH4 0xF4FD3062 EQ PUSH2 0x103
↪ JUMPI JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO PUSH2
↪ 0x50 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0x6B PUSH1 0x4 DUP1
↪ CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2 0x66 SWAP2 SWAP1 PUSH2 0xF29 JUMP
↪ JUMPDEST PUSH2 0x133 JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x78 SWAP2 SWAP1
↪ PUSH2 0x12E1 JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN
↪ JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0x8D JUMPI PUSH1 0x0 DUP1 REVERT
↪ JUMPDEST POP PUSH2 0xA8 PUSH1 0x4 DUP1 CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2
↪ 0xA3 SWAP2 SWAP1 PUSH2 0xE74 JUMP JUMPDEST PUSH2 0x449 JUMP JUMPDEST PUSH1
↪ 0x40 MLOAD PUSH2 0xB5 SWAP2 SWAP1 PUSH2 0x12E1 JUMP JUMPDEST PUSH1 0x40 MLOAD
↪ DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0xCA JUMPI
↪ PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0xE5 PUSH1 0x4 DUP1 CALLDATASIZE SUB
↪ DUP2 ADD SWAP1 PUSH2 0xE0 SWAP2 SWAP1 PUSH2 0xE74 JUMP JUMPDEST PUSH2 0x73D
↪ JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0xFA SWAP10 SWAP9 SWAP8 SWAP7 SWAP6
↪ SWAP5 SWAP4 SWAP3 SWAP2 SWAP1 PUSH2 0x1317 JUMP JUMPDEST PUSH1 0x40 MLOAD
↪ DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH2 0x11D PUSH1 0x4 DUP1 CALLDATASIZE
↪ SUB DUP2 ADD SWAP1 PUSH2 0x118 SWAP2 SWAP1 PUSH2 0xEC6 JUMP JUMPDEST PUSH2
↪ 0x8E4 JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x12A SWAP2 SWAP1 PUSH2 0x12FC
↪ JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH1 0x0
↪ DUP2 PUSH2 0x12E DUP2 PUSH2 0xCA7 JUMP JUMPDEST PUSH2 0x12E JUMPI PUSH1 0x40

```

(continues on next page)

10.2. Hash Time Lock Contract (HTLC)

193

```

↪ SUB DUP2 ADD SWAP1 PUSH2 0x118 SWAP2 SWAP1 PUSH2 0xEC6 JUMP JUMPDEST PUSH2
↪ 0x8E4 JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x12A SWAP2 SWAP1 PUSH2 0x12FC
↪ JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH1 0x0
↪ DUP2 PUSH2 0x12E DUP2 PUSH2 0xCA7 JUMP JUMPDEST PUSH2 0x12E JUMPI PUSH1 0x40

```

(continued from previous page)

```
balance(unit: str = 'Wei') → Union[Wei, int, float]
```

Get XinFin HTLC balance.

Parameters **unit** (*str*) – XinFin unit, default to XDC.

Returns int, float – XinFin HTLC balance.

```
>>> from swap.providers.bitcoin.htlc import HTLC
>>> from swap.utils import sha256
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.balance(unit="XDC")
1.56
```

$$\mathbf{xrc20_balance}(token_address: str) \rightarrow \text{Tuple}[\text{int}, \text{str}, \text{str}, \text{int}, \text{str}]$$

Get XinFin HTLC XRC20 balance.

Parameters `token_address` (*str*) – XinFin XRC20 token address.

Returns tuple – XinFin HTLC XRC20 balance and decimal.

[illegible]

10.3 Transaction

XinFin transaction in blockchain network.

[illegible]

XinFin Transaction.

Parameters

- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **xrc20** (*bool*) – Transaction XRC20 token, default to `False`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns Transaction – XinFin transaction instance.

Note: XinFin has only three networks, mainnet, apothem and testnet.

```
fee(unit: str = 'Wei') → Union[Wei, int, float]
```

Get XinFin transaction fee.

Parameters **unit** (*str*) – XinFin unit, default to Wei.

Returns Wei, int, float – XinFin transaction fee.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.providers.xinfin.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", htlc=htlc, amount=3, unit="XDC
↳ ")
>>> fund_transaction.fee(unit="Wei")
138436

```

hash() → Optional[str]

Get XinFin transaction hash.

Returns str – XinFin transaction hash.

```

>>> from swap.providers.xinfin.transaction import WithdrawTransaction
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "testnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeeef9", secret_
↳ key="Hello Meheret!", address="xdcf8D43806260CFc6cC79fB408BA1897054667F81C",
↳ contract_address="xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMcNvSDyHT6MnmJJGKHMrcUqaYpGojrug1Z
↳ ", path="m/44'/550'/0'/0/0")
>>> withdraw_transaction.sign(solver=withdraw_solver)
>>> withdraw_transaction.hash()
"0xe8e8738c791385738661573ad4de63dd81b77d240b6138ca476ea8cdcb29a21"

```

json() → dict

Get XinFin transaction fee.

Returns Wei, int, float – XinFin transaction fee.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.providers.xinfin.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", htlc=htlc, amount=3, unit="XDC
↳ ")

```

(continues on next page)

(continued from previous page)

```
>>> fund_transaction.json()
{'chainId': 1337, 'from': '0x2224caA2235DF8Da3D2016d2AB1137D2d548A232', 'value': 3000000000000000000, 'nonce': 2, 'gas': 138436, 'gasPrice': 20000000000, 'to': '0xdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7', 'data': '0xf4fd30623a26da82ead15a80533a02696656b14b5dbfd84eb14790f2e1be5e9e45820eeb00000000000000000000'}
>>>
```

raw() → Optional[str]
Get XinFin transaction hash.

Returns str – XinFin transaction hash.

```
>>> from swap.providers.xinfin.transaction import RefundTransaction
>>> from swap.providers.xinfin.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="testnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeeef9", address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWUubHLY3kQf
↳ ", path="m/44'/550'/0'/0/0")
>>> refund_transaction.sign(solver=refund_solver)
>>> refund_transaction.hash()

↳ "0xf88a028504a817c80082e76094de06b10c67765c8c0b9f64e0ef423b45eb86b8e780a47249fbb61909575c436a
↳ "
```

type() → str
Get XinFin transaction hash.

Returns str – XinFin transaction hash.

```
>>> from swap.providers.xinfin.transaction import WithdrawTransaction
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "testnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeef9", secret_
↳ key="Hello Meheret!", address="xdcf8D43806260Cfc6cC79fB408BA1897054667F81C",
↳ contract_address="xcdcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
>>> withdraw_transaction.type()
"xinfin_withdraw_unsigned"
```

signature() → dict
Get XinFin transaction hash.

Returns str – XinFin transaction hash.

```
>>> from swap.providers.xinfin.transaction import RefundTransaction
>>> from swap.providers.xinfin.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="testnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaef9", address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", contract_address=
↳ "xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
```

(continues on next page)

(continues on next page)

(continued from previous page)

```

>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHly3kQf",
↳ path="m/44'/550'/0'/0/0")
>>> refund_transaction.sign(solver=refund_solver)
>>> refund_transaction.signature()
{'hash': '0x90449ab8e3736feae4980554bb129b408f88d0003e569022cf8e00817cc2a7d9',
↳ 'rawTransaction':
↳ '0xf88a028504a817c80082e76094de06b10c67765c8c0b9f64e0ef423b45eb86b8e780a47249fbb61909575c436',
↳ 'r':
↳ '42895942847608608192932856733711858695420995837709512084644654454168196927042',
↳ 's':
↳ '23849944468865388317715121201379699989753020274517556595896033163737398323890',
↳ 'v': 2709}

```

transaction_raw() → str

Get XinFin fund transaction raw.

Returns str – XinFin fund transaction raw.

```

>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.providers.xinfin.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xgcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260Cfc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", htlc=htlc, amount=3, unit="XDC
↳ ")
>>> fund_transaction.transaction_raw()

↳ "eyJmZWUiOiAxMzg0MzYsICJ0eXB1IjogInhpbmZpb19mdW5kX3Vuc2lnbmVkiiwgInRyYW5zYWN0aW9uIjoeyJjaGF"
↳ ""

```

10.3.1 NormalTransaction

class swap.providers.xinfin.transaction.**NormalTransaction**(network: str = 'mainnet', xrc20: bool = False, provider: str = 'http')

XinFin Normal transaction.

Parameters

- **network** (str) – XinFin network, defaults to mainnet.
- **xrc20** (bool) – Normal transaction XRC20 token, default to False.
- **provider** (str) – XinFin network provider, defaults to http.

Returns NormalTransaction – XinFin normal transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(*address: str, recipient: dict, token_address: Optional[str] = None, unit: str = 'Wei'*) → *swap.providers.xinfin.transaction.NormalTransaction*

Build XinFin normal transaction.

Parameters

- **address** (*str*) – XinFin sender address.
- **recipient** (*dict*) – Recipients XinFin address and amount.
- **token_address** (*bool*) – XinFin XRC20 token address, default to None.
- **unit** (*str*) – XinFin unit, default to Wei.

Returns *NormalTransaction* – XinFin normal transaction instance.

```
>>> from swap.providers.xinfin.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", recipient={
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C": 100_000_000})
<swap.providers.xinfin.transaction.FundTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.xinfin.solver.NormalSolver*) →

swap.providers.xinfin.transaction.NormalTransaction

Sign XinFin normal transaction.

Parameters *solver* (*xinfin.solver.NormalSolver*) – XinFin normal solver.

Returns *NormalTransaction* – XinFin normal transaction instance.

```
>>> from swap.providers.xinfin.transaction import NormalTransaction
>>> from swap.providers.xinfin.solver import NormalSolver
>>> normal_transaction: NormalTransaction = NormalTransaction(network="testnet")
>>> normal_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", recipient={
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C": 100_000_000})
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHLY3kQf
↳ ", address=0)
>>> normal_transaction.sign(solver=normal_solver)
<swap.providers.xinfin.transaction.FundTransaction object at 0x0409DAF0>
```

10.3.2 FundTransaction

class *swap.providers.xinfin.transaction.FundTransaction*(*network: str = 'mainnet', xrc20: bool = False, provider: str = 'http'*)

XinFin Fund transaction.

Parameters

- **network** (*str*) – XinFin network, defaults to mainnet.
- **xrc20** (*bool*) – Fund transaction XRC20 token, default to False.
- **provider** (*str*) – XinFin network provider, defaults to http.

Returns FundTransaction – XinFin fund transaction instance.

Warning: Do not forget to build transaction after initialize fund transaction.

build_transaction(*address: str, htlc: swap.providers.xinfin.htlc.HTLC, amount: Union[Wei, int, float], unit: str = 'Wei'*) → *swap.providers.xinfin.transaction.FundTransaction*

Build XinFin fund transaction.

Parameters

- **htlc** (*xinfin.htlc.HTLC*) – XinFin HTLC instance.
- **address** (*str*) – XinFin sender address.
- **amount** (*Wei, int, float*) – XinFin amount or XRC20 amount.
- **unit** (*str*) – XinFin unit, default to Wei.

Returns FundTransaction – XinFin fund transaction instance.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.providers.xinfin.transaction import FundTransaction
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdbcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", htlc=htlc, amount=3, unit="XDC
↳ ")
<swap.providers.xinfin.transaction.FundTransaction object at 0x0409DAF0>
```

sign(*solver: swap.providers.xinfin.solver.FundSolver*) → *swap.providers.xinfin.transaction.FundTransaction*

Sign XinFin fund transaction.

Parameters **solver** (*xinfin.solver.FundSolver*) – XinFin fund solver.

Returns FundTransaction – XinFin fund transaction instance.

```
>>> from swap.providers.xinfin.htlc import HTLC
>>> from swap.providers.xinfin.transaction import FundTransaction
>>> from swap.providers.xinfin.solver import FundSolver
>>> from swap.utils import sha256, get_current_timestamp
>>> htlc: HTLC = HTLC(contract_address=
↳ "xdbcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7", network="testnet")
>>> htlc.build_htlc(secret_hash=sha256("Hello Meheret!"), recipient_address=
↳ "xdcf8D43806260CFc6cC79fB408BA1897054667F81C", sender_address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", endtime=get_current_
↳ timestamp(plus=3600))
>>> fund_transaction: FundTransaction = FundTransaction(network="testnet")
>>> fund_transaction.build_transaction(address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", htlc=htlc, amount=3, unit="XDC
↳ ")
```

(continues on next page)

(continued from previous page)

```
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwMv1fQEDioiGZXWXUbHly3kQf
↳ ", path="m/44'/550'/0'/0/0")
>>> fund_transaction.sign(solver=fund_solver)
<swap.providers.xinfin.transaction.FundTransaction object at 0x0409DAF0>
```

10.3.3 WithdrawTransaction

```
class swap.providers.xinfin.transaction.WithdrawTransaction(network: str = 'mainnet', xrc20: bool
= False, provider: str = 'http')
```

XinFin Withdraw transaction.

Parameters

- **network** (*str*) – XinFin network, defaults to mainnet.
- **xrc20** (*bool*) – Withdraw transaction XRC20 token, default to False.
- **provider** (*str*) – XinFin network provider, defaults to http.

Returns WithdrawTransaction – XinFin withdraw transaction instance.

Warning: Do not forget to build transaction after initialize withdraw transaction.

```
build_transaction(transaction_hash: str, address: str, secret_key: str, contract_address: Optional[str] =
None) → swap.providers.xinfin.transaction.WithdrawTransaction
```

Build XinFin withdraw transaction.

Parameters

- **transaction_hash** (*str*) – XinFin HTLC funded transaction hash.
- **address** (*str*) – XinFin recipient address.
- **secret_key** (*str*) – Secret password/passphrase.
- **contract_address** (*str*) – XinFin HTLC contract address, defaults to None.

Returns WithdrawTransaction – XinFin withdraw transaction instance.

```
>>> from swap.providers.xinfin.transaction import WithdrawTransaction
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "testnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeef9", secret_
↳ key="Hello Meheret!", address="xdcf8D43806260CFc6cC79fB408BA1897054667F81C",
↳ contract_address="xdcdE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
<swap.providers.xinfin.transaction.WithdrawTransaction object at 0x0409DAF0>
```

```
sign(solver: swap.providers.xinfin.solver.WithdrawSolver) →
swap.providers.xinfin.transaction.WithdrawTransaction
```

Sign XinFin withdraw transaction.

Parameters **solver** (*xinfin.solver.WithdrawSolver*) – XinFin withdraw solver.

Returns WithdrawTransaction – XinFin withdraw transaction instance.

```
>>> from swap.providers.xinfin.transaction import WithdrawTransaction
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> withdraw_transaction: WithdrawTransaction = WithdrawTransaction(network=
↳ "testnet")
>>> withdraw_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeef9", secret_
↳ key="Hello Meheret!", address="xdcf8D43806260CFc6cC79fB408BA1897054667F81C",
↳ contract_address="xcdcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMCnVSDyHT6MnmJJGKHMrcUqaYpGojrug1ZL
↳ ", path="m/44'/550'/0'/0/0")
>>> withdraw_transaction.sign(solver=withdraw_solver)
<swap.providers.xinfin.transaction.WithdrawTransaction object at 0x0409DAF0>
```

10.3.4 RefundTransaction

class swap.providers.xinfin.transaction.**RefundTransaction**(network: str = 'mainnet', xrc20: bool = False, provider: str = 'http')

XinFin Refund transaction.

Parameters

- **network** (str) – XinFin network, defaults to mainnet.
- **xrc20** (bool) – Refund transaction XRC20 token, default to False.
- **provider** (str) – XinFin network provider, defaults to http.

Returns RefundTransaction – XinFin refund transaction instance.

Warning: Do not forget to build transaction after initialize refund transaction.

build_transaction(transaction_hash: str, address: str, contract_address: Optional[str] = None) → swap.providers.xinfin.transaction.RefundTransaction

Build XinFin refund transaction.

Parameters

- **transaction_hash** (str) – XinFin HTLC funded transaction hash.
- **address** (str) – XinFin sender address.
- **contract_address** (str) – XinFin HTLC contract address, defaults to None.

Returns RefundTransaction – XinFin refund transaction instance.

```
>>> from swap.providers.xinfin.transaction import RefundTransaction
>>> refund_transaction: RefundTransaction = RefundTransaction(network="testnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaeef9", address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", contract_address=
↳ "xcdcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
<swap.providers.xinfin.transaction.RefundTransaction object at 0x0409DAF0>
```

sign(*solver*: swap.providers.xinfin.solver.RefundSolver) →
 swap.providers.xinfin.transaction.RefundTransaction
 Sign XinFin refund transaction.

Parameters **solver** (xinfin.solver.RefundSolver) – XinFin refund solver.

Returns RefundTransaction – XinFin refund transaction instance.

```
>>> from swap.providers.xinfin.transaction import RefundTransaction
>>> from swap.providers.xinfin.solver import RefundSolver
>>> refund_transaction: RefundTransaction = RefundTransaction(network="testnet")
>>> refund_transaction.build_transaction(transaction_hash=
↳ "0x0d4c93546aa3e5e476455931a63f1a97a2624e3b516e3fd8e3a582cb20aaef9", address=
↳ "xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232", contract_address=
↳ "xdbcE06b10c67765c8C0b9F64E0eF423b45Eb86b8e7")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWXUbHly3kQf
↳ ", path="m/44'/550'/0'/0/0")
>>> refund_transaction.sign(solver=refund_solver)
<swap.providers.xinfin.transaction.RefundTransaction object at 0x0409DAF0>
```

10.4 Solver

XinFin solver.

10.4.1 NormalSolver

class swap.providers.xinfin.solver.**NormalSolver**(*xprivate_key*: str, *account*: int = 0, *change*: bool = False, *address*: int = 0, *path*: Optional[str] = None, *strict*: bool = True)

XinFin Normal solver.

Parameters

- **xprivate_key** (str) – XinFin sender xprivate key.
- **account** (int) – XinFin derivation account, defaults to 0.
- **change** (bool) – XinFin derivation change, defaults to False.
- **address** (int) – XinFin derivation address, defaults to 0.
- **path** (str) – XinFin derivation path, defaults to None.
- **strict** (bool) – Strict for must be root xprivate key, default to True.

Returns NormalSolver – XinFin normal solver instance.

```
>>> from swap.providers.xinfin.solver import NormalSolver
>>> sender_root_xprivate_key: str =
↳ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfm4GqNRE9gWQHky25BpvBO
↳ "
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=sender_root_xprivate_
↳ key)
<swap.providers.xinfin.solver.FundSolver object at 0x03FCCA60>
```

10.4.2 FundSolver

```
class swap.providers.xinfin.solver.FundSolver(xprivate_key: str, account: int = 0, change: bool =
False, address: int = 0, path: Optional[str] = None,
strict: bool = True)
```

XinFin Fund solver.

Parameters

- **xprivate_key** (*str*) – XinFin sender xprivate key.
- **account** (*int*) – XinFin derivation account, defaults to 0.
- **change** (*bool*) – XinFin derivation change, defaults to False.
- **address** (*int*) – XinFin derivation address, defaults to 0.
- **path** (*str*) – XinFin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns FundSolver – XinFin fund solver instance.

```
>>> from swap.providers.xinfin.solver import FundSolver
>>> sender_root_xprivate_key: str =
↳ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB"
↳ ""
>>> fund_solver: FundSolver = FundSolver(xprivate_key=sender_root_xprivate_key,
↳ path="m/44'/550'/0'/0/0")
<swap.providers.xinfin.solver.FundSolver object at 0x03FCCA60>
```

10.4.3 WithdrawSolver

```
class swap.providers.xinfin.solver.WithdrawSolver(xprivate_key: str, account: int = 0, change: bool =
False, address: int = 0, path: Optional[str] =
None, strict: bool = True)
```

XinFin Withdraw solver.

Parameters

- **xprivate_key** (*str*) – XinFin sender xprivate key.
- **account** (*int*) – XinFin derivation account, defaults to 0.
- **change** (*bool*) – XinFin derivation change, defaults to False.
- **address** (*int*) – XinFin derivation address, defaults to 0.
- **path** (*str*) – XinFin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns WithdrawSolver – XinFin withdraw solver instance.

```
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> recipient_root_xprivate_key: str =
↳ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvB"
↳ ""
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=recipient_root_
↳ xprivate_key)
<swap.providers.xinfin.solver.WithdrawSolver object at 0x03FCCA60>
```

10.4.4 RefundSolver

```
class swap.providers.xinfin.solver.RefundSolver(xprivate_key: str, account: int = 0, change: bool = False, address: int = 0, path: Optional[str] = None, strict: bool = True)
```

XinFin Refund solver.

Parameters

- **xprivate_key** (*str*) – XinFin sender xprivate key.
- **account** (*int*) – XinFin derivation account, defaults to 0.
- **change** (*bool*) – XinFin derivation change, defaults to False.
- **address** (*int*) – XinFin derivation address, defaults to 0.
- **path** (*str*) – XinFin derivation path, defaults to None.
- **strict** (*bool*) – Strict for must be root xprivate key, default to True.

Returns RefundSolver – XinFin refund solver instance.

```
>>> from swap.providers.xinfin.solver import RefundSolver
>>> sender_root_xprivate_key: str =
↳ "xprv9s21ZrQH143K3XihXQBN8Uar2WBtrjSzK2oRDEGQ25pA2kKAADoQXaiiVXht163ZTrdtTXfM4GqNRE9gWQHky25BpvBC
↳ "
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=sender_root_xprivate_
↳ key)
<swap.providers.xinfin.solver.RefundSolver object at 0x03FCCA60>
```

10.5 Signature

XinFin signature.

```
class swap.providers.xinfin.signature.Signature(network: str = 'mainnet', xrc20: bool = False, provider: str = 'http')
```

XinFin Signature.

Parameters

- **network** (*str*) – XinFin network, defaults to mainnet.
- **xrc20** (*bool*) – Signature XRC20 token, default to False.
- **provider** (*str*) – XinFin network provider, defaults to http.

Returns Signature – XinFin signature instance.

Note: XinFin has only two networks, mainnet, apothem and testnet.

fee(*unit: str = 'Wei'*) → Union[Wei, int, float]

Get XinFin signature fee.

Parameters **unit** (*str*) – XinFin unit, default to Wie.

Returns Wei, int, float – XinFin signature fee.

```
>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> signature: Signature = Signature(network="testnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMcNvSDyHT6MnmJJGKHMrCUqaYpGojrug1ZJ
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3dpdGhkcmF3X3Vuc2lnbmVkJiIiwInRyYW5zYW90aW9uIjogeyJ
↳ ", solver=withdraw_solver)
>>> signature.fee(unit="Wei")
82689
```

hash() → Optional[str]
Get XinFin signature has.

Returns str – XinFin signature hash.

```
>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import RefundSolver
>>> signature: Signature = Signature(network="testnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmQTLhRgf86uZyCjK2ou36YdUjt5frjwihGwMv1fQEDioiGZXWXUbHLy3kQf
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAiA1OTIzMiwgInR5cGUiOiAiAieGluZmluX3JlZnVuZGF9bnNpZ25lZCIsICJ0cmFuc2FjdGlvbIi6IHsiY2hl
↳ ", solver=refund_solver)
>>> signature.hash()
"0x90449ab8e3736feae4980554bb129b408f88d0003e569022cf8e00817cc2a7d9"
```

json() → dict
Get XinFin signature json.

Returns dict – XinFin signature json.

[illegible]

raw() → Optional[str]
Get XinFin signature raw.

Returns str – XinFin signature raw.

```

>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> signature: Signature = Signature(network="testnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMCnVSDyHT6MnmJJGKHMrcUqaYpGojrug1Zf
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3dpdGhkcmF3X3Vuc2lnbmVkIiwgInRyYW5zYWN0aW9uIjogeyJ
↳ ", solver=withdraw_solver)
>>> signature.raw()

↳ "0xf8ec808504a817c8008301430194de06b10c67765c8c0b9f64e0ef423b45eb86b8e780b88406a536651909575
↳ "

```

type() → str

Get XinFin signature type.

Returns str – XinFin signature type.

```

>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import FundSolver
>>> signature: Signature = Signature(network="testnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHly3kQf
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0eXBliJogInhpbmZpb19mdW5kX3Vuc2lnbmVkIiwgInRyYW5zYWN0aW9uIjogeyJjaGF
↳ ", solver=fund_solver)
>>> signature.type()
"xinfin_fund_signed"

```

sign(*transaction_raw*: str, *solver*: Union[swap.providers.xinfin.solver.NormalSolver, swap.providers.xinfin.solver.FundSolver, swap.providers.xinfin.solver.WithdrawSolver, swap.providers.xinfin.solver.RefundSolver]) → Union[swap.providers.xinfin.signature.NormalSignature, swap.providers.xinfin.signature.FundSignature, swap.providers.xinfin.signature.WithdrawSignature, swap.providers.xinfin.signature.RefundSignature]
Sign XinFin unsigned transaction raw.

Parameters

- **transaction_raw** (str) – XinFin unsigned transaction raw.
- **solver** (xinfin.solver.NormalSolver, xinfin.solver.FundSolver, xinfin.solver.WithdrawSolver, xinfin.solver.RefundSolver) – XinFin solver.

Returns NormalSignature, FundSignature, WithdrawSignature, RefundSignature – XinFin signature instance.

```

>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import FundSolver
>>> signature: Signature = Signature(network="testnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHly3kQf
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0eXBliJogInhpbmZpb19mdW5kX3Vuc2lnbmVkIiwgInRyYW5zYWN0aW9uIjogeyJjaGF
↳ ", solver=fund_solver)

```

(continues on next page)

(continued from previous page)

```
<swap.providers.xinfin.signature.FundSignature object at 0x0409DAF0>
```

signature() → dict

Get XinFin signature.

Returns dict – XinFin signature.

```
>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> signature: Signature = Signature(network="testnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMCnVSDyHT6MnmJJGKHMrcUqaYpGojrug1Z",
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3dpdGhkcmF3X3Vuc2lnbmVkiIiwgInRyYW5zYWNoaW9uIjoeyJjaGF",
↳ ", solver=withdraw_solver)
>>> signature.signature()
{'hash': '0xe8e8738c791385738661573ad4de63dd81b77d240b6138ca476ea8cdcb29a21',
↳ 'rawTransaction':
↳ '0xf8ec808504a817c8008301430194de06b10c67765c8c0b9f64e0ef423b45eb86b8e780b88406a536651909575',
↳ ', 'r':
↳ '115683075740172584287236173170973052486872064110718784013746063807450268107094',
↳ 's':
↳ '10987326587522303302152973055763806493281157878637620947188858604750528344964',
↳ 'v': 2709}
```

transaction_raw() → str

Get XinFin signed transaction raw.

Returns str – XinFin signed transaction raw.

```
>>> from swap.providers.xinfin.signature import Signature
>>> from swap.providers.xinfin.solver import FundSolver
>>> signature: Signature = Signature(network="testnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWXUbHLY3kQf",
↳ ", account=0, change=False, address=0)
>>> signature.sign(transaction_raw=
↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3dpdGhkcmF3X3Vuc2lnbmVkiIiwgInRyYW5zYWNoaW9uIjoeyJjaGF",
↳ ", solver=fund_solver)
>>> signature.transaction_raw()

↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3dpdGhkcmF3X3Vuc2lnbmVkiIiwgInRyYW5zYWNoaW9uIjoeyJjaGF",
↳ "
```

10.5.1 NormalSignature

```
class swap.providers.xinfin.signature.NormalSignature(network: str = 'mainnet', xrc20: bool = False,
                                                    provider: str = 'http')
```

XinFin Normal signature.

Parameters

- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **xrc20** (*bool*) – Normal signature XRC20 token, default to `False`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns `NormalSignature` – XinFin normal signature instance.

Note: XinFin has only five networks, `mainnet`, `ropsten`, `kovan`, `rinkeby` and `testnet`.

```
sign(transaction_raw: str, solver: swap.providers.xinfin.solver.NormalSolver) →
    swap.providers.xinfin.signature.NormalSignature
Sign XinFin unsigned normal transaction raw.
```

Parameters

- **transaction_raw** (*str*) – XinFin unsigned normal transaction raw.
- **solver** (`xinfin.solver.NormalSolver`) – XinFin solver.

Returns `NormalSignature` – XinFin normal signature instance.

```
>>> from swap.providers.xinfin.signature import NormalSignature
>>> from swap.providers.xinfin.solver import NormalSolver
>>> normal_signature: NormalSignature = NormalSignature(network="mainnet")
>>> normal_solver: NormalSolver = NormalSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUubHLY3kQf
↳ ")
>>> normal_signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0cmFuc2FjdGlvbiI6IHsiY2hhaW5JZCI6IDEzMzcsICJmcm9tIjogIjB4NjllMDRmZTE
↳ ", solver=normal_solver)
<swap.providers.xinfin.signature.NormalSignature object at 0x0409DAF0>
```

10.5.2 FundSignature

```
class swap.providers.xinfin.signature.FundSignature(network: str = 'mainnet', xrc20: bool = False,
                                                    provider: str = 'http')
```

XinFin Fund signature.

Parameters

- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **xrc20** (*bool*) – Fund signature XRC20 token, default to `False`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns `FundSignature` – XinFin fund signature instance.

Note: XinFin has only two networks, mainnet, apothem and testnet.

sign(*transaction_raw*: str, *solver*: swap.providers.xinfin.solver.FundSolver) →
 swap.providers.xinfin.signature.FundSignature
 Sign XinFin unsigned fund transaction raw.

Parameters

- **transaction_raw** (str) – XinFin unsigned fund transaction raw.
- **solver** (xinfin.solver.FundSolver) – XinFin solver.

Returns FundSignature – XinFin fund signature instance.

```
>>> from swap.providers.xinfin.signature import FundSignature
>>> from swap.providers.xinfin.solver import FundSolver
>>> fund_signature: FundSignature = FundSignature(network="testnet")
>>> fund_solver: FundSolver = FundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCJk2ou36YdUJt5frjwihGwmV1fQEDioiGZXWUubHly3kQf
↳ ", account=0, change=False, address=0)
>>> fund_signature.sign(transaction_raw=
↳ "eyJmZWUiOiAxMzg0NDgsICJ0eXBBIjogInhpbmZpb19mdW5kX3Vuc2lnbmVkiiwgInRyYW5zYWN0aW9uIjogetJjaGF
↳ ", solver=fund_solver)
<swap.providers.xinfin.signature.FundSignature object at 0x0409DAF0>
```

10.5.3 WithdrawSignature

class swap.providers.xinfin.signature.**WithdrawSignature**(*network*: str = 'mainnet', *xrc20*: bool =
 False, *provider*: str = 'http')

XinFin Withdraw signature.

Parameters

- **network** (str) – XinFin network, defaults to mainnet.
- **xrc20** (bool) – Fund signature XRC20 token, default to False.
- **provider** (str) – XinFin network provider, defaults to http.

Returns WithdrawSignature – XinFin withdraw signature instance.

Note: XinFin has only two networks, mainnet, apothem and testnet.

sign(*transaction_raw*: str, *solver*: swap.providers.xinfin.solver.WithdrawSolver) →
 swap.providers.xinfin.signature.WithdrawSignature
 Sign XinFin unsigned withdraw transaction raw.

Parameters

- **transaction_raw** (str) – XinFin unsigned withdraw transaction raw.
- **solver** (xinfin.solver.WithdrawSolver) – XinFin withdraw solver.

Returns WithdrawSignature – XinFin withdraw signature instance.

```

>>> from swap.providers.xinfin.signature import WithdrawSignature
>>> from swap.providers.xinfin.solver import WithdrawSolver
>>> withdraw_signature: WithdrawSignature = WithdrawSignature(network="testnet")
>>> withdraw_solver: WithdrawSolver = WithdrawSolver(xprivate_key=
↳ "xprv9s21ZrQH143K4Kpce43z5guPyxLrFoc2i8aQAq835Zzp4Rt7i6nZaMCnVSDyHT6MnmJJGKHMrcUqaYpGojrug1ZJ
↳ ", account=0, change=False, address=0)
>>> withdraw_signature.sign(transaction_raw=
↳ "eyJmZWUiOiA4MjY4OSwgInR5cGUiOiAieGluZmluX3ZpdGhkcmluX3Z3Vuc2lnbmVklIiwgInRyYW5zYW9uIjogeyJ
↳ ", solver=withdraw_solver)
<swap.providers.xinfin.signature.WithdrawSignature object at 0x0409DAF0>

```

10.5.4 RefundSignature

class swap.providers.xinfin.signature.**RefundSignature**(network: str = 'mainnet', xrc20: bool = False, provider: str = 'http')

XinFin Refund signature.

Parameters

- **network** (str) – XinFin network, defaults to mainnet.
- **xrc20** (bool) – Fund signature XRC20 token, default to False.
- **provider** (str) – XinFin network provider, defaults to http.

Returns RefundSignature – XinFin refund signature instance.

Note: XinFin has only two networks, mainnet, apothem and testnet.

sign(transaction_raw: str, solver: swap.providers.xinfin.solver.RefundSolver) → swap.providers.xinfin.signature.RefundSignature
Sign XinFin unsigned refund transaction raw.

Parameters

- **transaction_raw** (str) – XinFin unsigned refund transaction raw.
- **solver** (xinfin.solver.RefundSolver) – XinFin refund solver.

Returns RefundSignature – XinFin refund signature instance.

```

>>> from swap.providers.xinfin.signature import RefundSignature
>>> from swap.providers.xinfin.solver import RefundSolver
>>> refund_signature: RefundSignature = RefundSignature(network="testnet")
>>> refund_solver: RefundSolver = RefundSolver(xprivate_key=
↳ "xprv9s21ZrQH143K3Y3pdbkbjreZQ9RVmqTLhRgf86uZyCjk2ou36YdUJt5frjwihGWmV1fQEDioiGZXWUUbHly3kQf
↳ ", account=0, change=False, address=0)
>>> refund_signature.sign(transaction_raw=
↳ "eyJmZWUiOiA1OTIzMiwgInR5cGUiOiAieGluZmluX3JlZnVuZD91bnNpZ25lZCIsICJ0cmFuc2FjdGlvdjI6IHsiY2h
↳ ", solver=refund_solver)
<swap.providers.xinfin.signature.RefundSignature object at 0x0409DAF0>

```

10.6 Remote Procedure Call (RPC)

XinFin remote procedure call.

`swap.providers.xinfin.rpc.get_web3(network: str = 'mainnet', provider: str = 'http') → web3.main.Web3`
Get XinFin Web3 instance.

Parameters

- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns Web3 – XinFin Web3 instance.

```
>>> from swap.providers.xinfin.rpc import get_web3
>>> get_web3(network="testnet", provider="http")
<web3.main.Web3 object at 0x000001DDECCD0640>
```

```
swap.providers.xinfin.rpc.get_balance(address: str, network: str = 'mainnet', provider: str = 'http') → Wei
```

Get XinFin balance.

Parameters

- **address** (*str*) – XinFin address.
- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns Wei – XinFin balance (Wei).

```
>>> from swap.providers.xinfin.rpc import get_balance
>>> get_balance("xdc70c1eb09363603a3b6391deb2daa6d2561a62f52", "mainnet")
71560900
```

```
swap.providers.xinfin.rpc.get_xrc20_balance(address: str, token_address: str, network: str = 'mainnet',
                                             provider: str = 'http') → Tuple[int, str, str, int, str]
```

Get XinFin XRC20 token balance.

Parameters

- **address** (*str*) – XinFin address.
- **token_address** (*str*) – XinFin XRC20 token address.
- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns tuple – XinFin XRC20 token balance and decimals.

[illegible]

```
swap.providers.xinfin.rpc.get_xrc20_decimals(token_address: str, network: str = 'mainnet', provider: str = 'http') → int
```

Get XinFin XRC20 token decimals.

Parameters

- **token_address** (*str*) – XinFin XRC20 token address.
- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns `int` – XinFin XRC20 token decimals.

```
>>> from swap.providers.xinfin.rpc import get_xrc20_decimals
>>> get_xrc20_decimals(token_address="0xDA6844e863bdfEE6AaFf888D2D34Bf1B7c37861",
↳ network="testnet")
18
```

`swap.providers.xinfin.rpc.get_transaction(transaction_hash: str, network: str = 'mainnet', provider: str = 'http') → dict`

Get XinFin transaction detail.

Parameters

- **transaction_hash** (*str*) – XinFin transaction hash/id.
- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns `dict` – XinFin transaction detail.

```
>>> from swap.providers.xinfin.rpc import get_transaction
>>> get_transaction(transaction_hash=
↳ "0xa4d57071427e3310b3e2fb16e7712f8d8aaaafb31ce5fcd6534fc50848905948")
{'hash': '0xa4d57071427e3310b3e2fb16e7712f8d8aaaafb31ce5fcd6534fc50848905948',
↳ 'nonce': 0, 'blockHash':
↳ '0xb33a804ae10713bf549db8ec749f7d650347613ac784db1a8d17e0cb03741bf0', 'blockNumber
↳ ': 1, 'transactionIndex': 0, 'from': '0x96cA14396341480E3b6384D1d1397d1f7f5a0AB7',
↳ 'to': None, 'value': 0, 'gas': 367400, 'gasPrice': 250000000, 'input':
↳ '0x608060405234801561001057600080fd5b506040518060400160405280600581526020017f48656c6c6f0000000000
↳ ', 'v': 28, 'r':
↳ '0xa593dcfd7f7b17f8b22907e9c4b03721312a4d00dfd99f8f7267ccd5eb7d4613', 's':
↳ '0x70cd172ae92de7a046dfe28de1db8657f8c3b3ed00c060392fb1d5080646927b'}
```

`swap.providers.xinfin.rpc.get_transaction_receipt(transaction_hash: str, network: str = 'mainnet', provider: str = 'http', headers: dict = {'accept': 'application/json', 'content-type': 'application/json', 'charset=utf-8', 'user-agent': 'Swap User-Agent 0.5.0'}, timeout: int = 60) → Optional[dict]`

Get XinFin transaction receipt.

Parameters

- **transaction_hash** (*str*) – XinFin transaction hash/id.
- **network** (*str*) – XinFin network, defaults to `mainnet`.
- **provider** (*str*) – XinFin network provider, defaults to `http`.
- **headers** (*dict*) – Request headers, default to `common` headers.
- **timeout** (*int*) – request timeout, default to `60`.

Returns `dict` – XinFin transaction receipt.

```
swap.providers.xinfin.rpc.wait_for_transaction_receipt(transaction_hash: str, network: str =
'mainnet', timeout: int = 60, provider: str =
'http', headers: dict = {'accept':
'application/json', 'content-type':
'application/json; charset=utf-8',
'user-agent': 'Swap User-Agent 0.5.0'}) →
dict
```

Parameters

- **transaction_hash** (*str*) – XinFin transaction hash/id.
- **network** (*str*) – XinFin network, defaults to mainnet.
- **timeout** (*int*) – request timeout, default to 60.
- **provider** (*str*) – XinFin network provider, defaults to http.
- **headers** (*dict*) – Request headers. default to `common` headers.

[illegible]

Parameters **raw** (*str*) – XinFin transaction raw.

213

```
>>> from swap.providers.xinfin.rpc import decode_raw
>>> decode_raw(raw=
↳ "0xf86c02840ee6b280825208943e0a9b2ee8f8341a1ead3e7531d75f1e395f24b8901236efcbcb340000801ba03084
↳ ")
{'hash': '0x04b3bfb804f2b3329555c6f3a17a794b3f099b6435a9cf58c78609ed93853907', 'from':
↳ '0x3769F63e3b694cD2e973e28af59bdFd751303273', 'to':
↳ '0x3e0a9B2Ee8F8341A1aEaD3E7531d75f1e395F24b', 'nonce': 2, 'gas': 21000, 'gas_price':
↳ '250000000', 'value': 2100000000000000000000, 'data': '0x', 'chain_id': -4, 'r':
↳ '0x3084982e4a9dd897d3cc1b2c8cc2d1b106b9d302eb23f6fae7d0e57e53e043f8', 's':
↳ '0x116f13f9ab385f6b53e7821b3335ced924a1ceb88303347cd0af4aa75e6bfb73', 'v': 27}
```

`swap.providers.xinfin.rpc.submit_raw(raw: str, network: str = 'mainnet', provider: str = 'http') → str`
 Submit original XinFin raw into blockchain.

Parameters

- **raw** (*str*) – XinFin transaction raw.
- **network** (*str*) – XinFin network, defaults to mainnet.
- **provider** (*str*) – XinFin network provider, defaults to http.

Returns *str* – XinFin submitted transaction hash/id.

```
>>> from swap.providers.xinfin.rpc import submit_raw
>>> submit_raw(raw=
↳ "0xf86c02840ee6b280825208943e0a9b2ee8f8341a1ead3e7531d75f1e395f24b8901236efcbcb340000801ba03084
↳ ", network="testnet")
"0x04b3bfb804f2b3329555c6f3a17a794b3f099b6435a9cf58c78609ed93853907"
```

10.7 Utils

XinFin Utils.

`swap.providers.xinfin.utils.is_network(network: str) → bool`
 Check XinFin network.

Parameters **network** (*str*) – XinFin network.

Returns *bool* – XinFin valid/invalid network.

```
>>> from swap.providers.xinfin.utils import is_network
>>> is_network(network="apothem")
True
```

`swap.providers.xinfin.utils.is_address(address: str) → bool`
 Check XinFin address.

Parameters **address** (*str*) – XinFin address.

Returns *bool* – XinFin valid/invalid address.

```
>>> from swap.providers.xinfin.utils import is_address
>>> is_address(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232")
True
```

`swap.providers.xinfin.utils.is_checksum_address(address: str) → bool`
 Check XinFin checksum address.

Parameters `address (str)` – XinFin address.

Returns `bool` – XinFin valid/invalid checksum address.

```
>>> from swap.providers.xinfin.utils import is_checksum_address
>>> is_checksum_address(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232")
False
>>> is_checksum_address(address="xdc1Ee11011ae12103a488A82DC33e03f337Bc93ba7")
True
```

`swap.providers.xinfin.utils.to_checksum_address(address: str, prefix: str = 'xdc') → Union[str, ChecksumAddress]`
 Change XinFin address to checksum address.

Parameters

- **address (str)** – XinFin address.
- **prefix (str)** – XinFin address prefix, default to `xdc`.

Returns `str, ChecksumAddress` – XinFin checksum address.

```
>>> from swap.providers.xinfin.utils import to_checksum_address
>>> to_checksum_address(address="xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232")
"xdc2224caA2235DF8Da3D2016d2AB1137D2d548A232"
```

`swap.providers.xinfin.utils.is_transaction_raw(transaction_raw: str) → bool`
 Check XinFin transaction raw.

Parameters `transaction_raw (str)` – XinFin transaction raw.

Returns `bool` – XinFin valid/invalid transaction raw.

```
>>> from swap.providers.xinfin.utils import is_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBwenAyNGRrZmZlbHlzOHpjd311"
↳ ""
>>> is_transaction_raw(transaction_raw=transaction_raw)
True
```

`swap.providers.xinfin.utils.decode_transaction_raw(transaction_raw: str) → dict`
 Decode XinFin transaction raw.

Parameters `transaction_raw (str)` – XinFin transaction raw.

Returns `dict` – Decoded xinfin transaction raw.

```
>>> from swap.providers.xinfin.utils import decode_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTBwenAyNGRrZmZlbHlzOHpjd311"
↳ ""
>>> decode_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'address': '...', 'transaction': {...}, 'unsigned_datas
↳ ': [...], 'signatures': [...], 'network': '...'}
```

`swap.providers.xinfin.utils.submit_transaction_raw(transaction_raw: str, provider: str = 'http') → dict`

Submit XinFin transaction raw.

Parameters

- **transaction_raw** (*str*) – XinFin transaction raw.
- **provider** (*str*) – XinFin network provider, defaults to `http`.

Returns dict – XinFin submitted transaction id, fee, type and date.

```
>>> from swap.providers.xinfin.utils import submit_transaction_raw
>>> transaction_raw: str =
↳ "eyJmZWUiOiAxMDAwMDAwMCwgImFkZHJlc3MiOiAiYm0xcWU5MHFqdDl3NG04cnQzdG51dTbwenAyNGRrZmZlbHlzOHpjd311"
↳ ""
>>> submit_transaction_raw(transaction_raw=transaction_raw)
{'fee': ..., 'type': '...', 'transaction_id': '...', 'network': '...', 'date': '...'
↳ '}'
```

`swap.providers.xinfin.utils.amount_unit_converter(amount: Union[int, float], unit_from: str = 'Wei2XDC') → Union[int, float]`

XinFin amount unit converter.

Parameters

- **amount** (*int*, *float*) – XinFin amount.
- **unit_from** (*str*) – XinFin unit from, default to `Wei2XDC`

Returns int, float – XinFin amount.

```
>>> from swap.providers.xinfin.utils import amount_unit_converter
>>> amount_unit_converter(amount=100_000_000, unit_from="Wei2XDC")
0.1
```

PYTHON MODULE INDEX

S

- swap.providers.bitcoin.htlc, 41
- swap.providers.bitcoin.rpc, 61
- swap.providers.bitcoin.signature, 54
- swap.providers.bitcoin.solver, 52
- swap.providers.bitcoin.transaction, 44
- swap.providers.bitcoin.utils, 64
- swap.providers.bitcoin.wallet, 33
- swap.providers.bytom.htlc, 74
- swap.providers.bytom.rpc, 96
- swap.providers.bytom.signature, 89
- swap.providers.bytom.solver, 87
- swap.providers.bytom.transaction, 78
- swap.providers.bytom.utils, 102
- swap.providers.bytom.wallet, 67
- swap.providers.ethereum.htlc, 113
- swap.providers.ethereum.rpc, 135
- swap.providers.ethereum.signature, 128
- swap.providers.ethereum.solver, 126
- swap.providers.ethereum.transaction, 118
- swap.providers.ethereum.utils, 139
- swap.providers.ethereum.wallet, 105
- swap.providers.vapor.htlc, 150
- swap.providers.vapor.rpc, 172
- swap.providers.vapor.signature, 165
- swap.providers.vapor.solver, 163
- swap.providers.vapor.transaction, 154
- swap.providers.vapor.utils, 178
- swap.providers.vapor.wallet, 143
- swap.providers.xinfin.htlc, 189
- swap.providers.xinfin.rpc, 211
- swap.providers.xinfin.signature, 204
- swap.providers.xinfin.solver, 202
- swap.providers.xinfin.transaction, 194
- swap.providers.xinfin.utils, 214
- swap.providers.xinfin.wallet, 181
- swap.utils, 29

Symbols

- account <account>
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-ethereum-sign command line option, 20
 - swap-vapor-sign command line option, 23
 - swap-xinfin-sign command line option, 27
- address <address>
 - swap-bitcoin-fund command line option, 10
 - swap-bitcoin-refund command line option, 11
 - swap-bitcoin-sign command line option, 12
 - swap-bitcoin-withdraw command line option, 13
 - swap-bytom-fund command line option, 14
 - swap-bytom-refund command line option, 15
 - swap-bytom-sign command line option, 16
 - swap-bytom-withdraw command line option, 17
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-sign command line option, 20
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-fund command line option, 21
 - swap-vapor-refund command line option, 22
 - swap-vapor-sign command line option, 23
 - swap-vapor-withdraw command line option, 24
 - swap-xinfin-refund command line option, 26
 - swap-xinfin-sign command line option, 27
 - swap-xinfin-withdraw command line option, 27
- amount <amount>
 - swap-bitcoin-fund command line option, 10
 - swap-bytom-fund command line option, 14
 - swap-ethereum-fund command line option, 18
 - swap-vapor-fund command line option, 21
 - swap-xinfin-fund command line option, 25
- asset <asset>
 - swap-bytom-fund command line option, 14
 - swap-bytom-refund command line option, 15
 - swap-bytom-withdraw command line option, 17
 - swap-vapor-fund command line option, 21
 - swap-vapor-refund command line option, 22
 - swap-vapor-withdraw command line option, 24
- bytecode <bytecode>
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-vapor-sign command line option, 23
- change <change>
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-ethereum-sign command line option, 20
 - swap-vapor-sign command line option, 23
 - swap-xinfin-sign command line option, 27
- contract-address <contract_address>
 - swap-bitcoin-fund command line option, 10
 - swap-bytom-fund command line option, 14
 - swap-ethereum-fund command line option, 18
 - swap-ethereum-htlc command line option, 19
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-fund command line option, 21
 - swap-xinfin-fund command line option, 25
 - swap-xinfin-htlc command line option, 26
 - swap-xinfin-refund command line option, 26
 - swap-xinfin-withdraw command line option, 27
- endblock <endblock>
 - swap-bytom-htlc command line option, 15
 - swap-vapor-htlc command line option, 22

```
--endpoint <endpoint>
    swap-bitcoin-submit command line option,
    13
--endtime <endtime>
    swap-bitcoin-htlc command line option, 11
    swap-bitcoin-sign command line option, 12
    swap-ethereum-fund command line option,
    18
    swap-xinfin-fund command line option, 25
--erc20 <erc20>
    swap-ethereum-fund command line option,
    18
    swap-ethereum-htlc command line option,
    19
    swap-ethereum-refund command line
    option, 19
    swap-ethereum-withdraw command line
    option, 21
--indent <indent>
    swap-bitcoin-decode command line option,
    10
    swap-bitcoin-htlc command line option, 11
    swap-bytom-decode command line option, 14
    swap-bytom-htlc command line option, 15
    swap-ethereum-decode command line
    option, 17
    swap-ethereum-htlc command line option,
    19
    swap-vapor-decode command line option, 21
    swap-vapor-htlc command line option, 22
    swap-xinfin-decode command line option,
    24
    swap-xinfin-htlc command line option, 26
--indexes <indexes>
    swap-bytom-sign command line option, 16
    swap-vapor-sign command line option, 23
--network <network>
    swap-bitcoin-fund command line option, 10
    swap-bitcoin-htlc command line option, 11
    swap-bitcoin-refund command line option,
    11
    swap-bitcoin-withdraw command line
    option, 13
    swap-bytom-fund command line option, 14
    swap-bytom-htlc command line option, 15
    swap-bytom-refund command line option, 15
    swap-bytom-withdraw command line option,
    17
    swap-ethereum-fund command line option,
    18
    swap-ethereum-htlc command line option,
    19
    swap-ethereum-refund command line
    option, 19
    swap-ethereum-withdraw command line
    option, 20
    swap-vapor-fund command line option, 22
    swap-vapor-htlc command line option, 22
    swap-vapor-refund command line option, 23
    swap-vapor-withdraw command line option,
    24
    swap-xinfin-fund command line option, 25
    swap-xinfin-htlc command line option, 26
    swap-xinfin-refund command line option,
    26
    swap-xinfin-withdraw command line
    option, 27
--offline <offline>
    swap-bitcoin-decode command line option,
    10
--path <path>
    swap-bitcoin-sign command line option, 12
    swap-bytom-sign command line option, 16
    swap-ethereum-sign command line option,
    20
    swap-vapor-sign command line option, 23
    swap-xinfin-sign command line option, 27
--recipient-address <recipient_address>
    swap-bitcoin-htlc command line option, 11
    swap-ethereum-fund command line option,
    18
    swap-xinfin-fund command line option, 25
--recipient-public-key
    <recipient_public_key>
    swap-bytom-htlc command line option, 15
    swap-vapor-htlc command line option, 22
--secret-hash <secret_hash>
    swap-bitcoin-htlc command line option, 11
    swap-bytom-htlc command line option, 15
    swap-ethereum-fund command line option,
    18
    swap-vapor-htlc command line option, 22
    swap-xinfin-fund command line option, 25
--secret-key <secret_key>
    swap-bitcoin-sign command line option, 12
    swap-bytom-sign command line option, 16
    swap-ethereum-withdraw command line
    option, 20
    swap-vapor-sign command line option, 23
    swap-xinfin-withdraw command line
    option, 27
--sender-address <sender_address>
    swap-bitcoin-htlc command line option, 11
    swap-ethereum-fund command line option,
    18
    swap-xinfin-fund command line option, 25
--sender-public-key <sender_public_key>
    swap-bytom-htlc command line option, 15
```

```

    swap-vapor-htlc command line option, 22
--token-address <token_address>
    swap-ethereum-fund command line option,
    18
    swap-xinfin-fund command line option, 25
--transaction-hash <transaction_hash>
    swap-bitcoin-refund command line option,
    11
    swap-bitcoin-withdraw command line
    option, 13
    swap-bytom-refund command line option, 15
    swap-bytom-withdraw command line option,
    17
    swap-ethereum-refund command line
    option, 19
    swap-ethereum-withdraw command line
    option, 20
    swap-vapor-refund command line option, 22
    swap-vapor-withdraw command line option,
    24
    swap-xinfin-refund command line option,
    26
    swap-xinfin-withdraw command line
    option, 27
--transaction-raw <transaction_raw>
    swap-bitcoin-decode command line option,
    10
    swap-bitcoin-sign command line option, 12
    swap-bitcoin-submit command line option,
    13
    swap-bytom-decode command line option, 14
    swap-bytom-sign command line option, 16
    swap-bytom-submit command line option, 16
    swap-ethereum-decode command line
    option, 17
    swap-ethereum-sign command line option,
    20
    swap-ethereum-submit command line
    option, 20
    swap-vapor-decode command line option, 21
    swap-vapor-sign command line option, 23
    swap-vapor-submit command line option, 24
    swap-xinfin-decode command line option,
    24
    swap-xinfin-sign command line option, 27
    swap-xinfin-submit command line option,
    27
--unit <unit>
    swap-bitcoin-fund command line option, 10
    swap-bytom-fund command line option, 14
    swap-ethereum-fund command line option,
    18
    swap-vapor-fund command line option, 21
    swap-xinfin-fund command line option, 25
--version
    swap command line option, 9
--version <version>
    swap-bitcoin-fund command line option, 10
    swap-bitcoin-refund command line option,
    11
    swap-bitcoin-sign command line option, 12
    swap-bitcoin-withdraw command line
    option, 13
--xprivate-key <xprivate_key>
    swap-bitcoin-sign command line option, 12
    swap-bytom-sign command line option, 16
    swap-ethereum-sign command line option,
    20
    swap-vapor-sign command line option, 23
    swap-xinfin-sign command line option, 27
--xrc20 <xrc20>
    swap-xinfin-fund command line option, 25
    swap-xinfin-htlc command line option, 26
    swap-xinfin-refund command line option,
    26
    swap-xinfin-withdraw command line
    option, 28
-a
    swap-bitcoin-fund command line option, 10
    swap-bitcoin-refund command line option,
    11
    swap-bitcoin-withdraw command line
    option, 13
    swap-bytom-fund command line option, 14
    swap-bytom-refund command line option, 15
    swap-bytom-withdraw command line option,
    17
    swap-ethereum-refund command line
    option, 19
    swap-ethereum-withdraw command line
    option, 20
    swap-vapor-fund command line option, 21
    swap-vapor-refund command line option, 22
    swap-vapor-withdraw command line option,
    24
    swap-xinfin-refund command line option,
    26
    swap-xinfin-withdraw command line
    option, 27
-ac
    swap-bitcoin-sign command line option, 12
    swap-bytom-sign command line option, 16
    swap-ethereum-sign command line option,
    20
    swap-vapor-sign command line option, 23
    swap-xinfin-sign command line option, 27
-ad
    swap-bitcoin-sign command line option, 12

```

- swap-bytom-sign command line option, 16
- swap-ethereum-sign command line option, 20
- swap-vapor-sign command line option, 23
- swap-xinfin-sign command line option, 27
- am
 - swap-bitcoin-fund command line option, 10
 - swap-bytom-fund command line option, 14
 - swap-ethereum-fund command line option, 18
 - swap-vapor-fund command line option, 21
 - swap-xinfin-fund command line option, 25
- as
 - swap-bytom-fund command line option, 14
 - swap-bytom-refund command line option, 15
 - swap-bytom-withdraw command line option, 17
 - swap-vapor-fund command line option, 21
 - swap-vapor-refund command line option, 22
 - swap-vapor-withdraw command line option, 24
- b
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-vapor-sign command line option, 23
- ca
 - swap-bitcoin-fund command line option, 10
 - swap-bytom-fund command line option, 14
 - swap-ethereum-fund command line option, 18
 - swap-ethereum-htlc command line option, 19
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-fund command line option, 21
 - swap-xinfin-fund command line option, 25
 - swap-xinfin-htlc command line option, 26
 - swap-xinfin-refund command line option, 26
 - swap-xinfin-withdraw command line option, 27
- ch
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-ethereum-sign command line option, 20
 - swap-vapor-sign command line option, 23
 - swap-xinfin-sign command line option, 27
- e
 - swap-bitcoin-htlc command line option, 11
 - swap-bitcoin-sign command line option, 12
 - swap-bitcoin-submit command line option, 13
 - swap-bytom-htlc command line option, 15
 - swap-ethereum-fund command line option, 18
 - swap-vapor-htlc command line option, 22
 - swap-xinfin-fund command line option, 25
- e20
 - swap-ethereum-fund command line option, 18
 - swap-ethereum-htlc command line option, 19
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-withdraw command line option, 21
- i
 - swap-bitcoin-decode command line option, 10
 - swap-bitcoin-htlc command line option, 11
 - swap-bytom-decode command line option, 14
 - swap-bytom-htlc command line option, 15
 - swap-bytom-sign command line option, 16
 - swap-ethereum-decode command line option, 17
 - swap-ethereum-htlc command line option, 19
 - swap-vapor-decode command line option, 21
 - swap-vapor-htlc command line option, 22
 - swap-vapor-sign command line option, 23
 - swap-xinfin-decode command line option, 24
 - swap-xinfin-htlc command line option, 26
- n
 - swap-bitcoin-fund command line option, 10
 - swap-bitcoin-htlc command line option, 11
 - swap-bitcoin-refund command line option, 11
 - swap-bitcoin-withdraw command line option, 13
 - swap-bytom-fund command line option, 14
 - swap-bytom-htlc command line option, 15
 - swap-bytom-refund command line option, 15
 - swap-bytom-withdraw command line option, 17
 - swap-ethereum-fund command line option, 18
 - swap-ethereum-htlc command line option, 19
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-fund command line option, 22

- swap-vapor-htlc command line option, 22
- swap-vapor-refund command line option, 23
- swap-vapor-withdraw command line option, 24
- swap-xinfin-fund command line option, 25
- swap-xinfin-htlc command line option, 26
- swap-xinfin-refund command line option, 26
- swap-xinfin-withdraw command line option, 27
- o
 - swap-bitcoin-decode command line option, 10
- p
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-ethereum-sign command line option, 20
 - swap-vapor-sign command line option, 23
 - swap-xinfin-sign command line option, 27
- ra
 - swap-bitcoin-htlc command line option, 11
 - swap-ethereum-fund command line option, 18
 - swap-xinfin-fund command line option, 25
- rpk
 - swap-bytom-htlc command line option, 15
 - swap-vapor-htlc command line option, 22
- sa
 - swap-bitcoin-htlc command line option, 11
 - swap-ethereum-fund command line option, 18
 - swap-xinfin-fund command line option, 25
- sh
 - swap-bitcoin-htlc command line option, 11
 - swap-bytom-htlc command line option, 15
 - swap-ethereum-fund command line option, 18
 - swap-vapor-htlc command line option, 22
 - swap-xinfin-fund command line option, 25
- sk
 - swap-bitcoin-sign command line option, 12
 - swap-bytom-sign command line option, 16
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-sign command line option, 23
 - swap-xinfin-withdraw command line option, 27
- spk
 - swap-bytom-htlc command line option, 15
 - swap-vapor-htlc command line option, 22
- ta
 - swap-ethereum-fund command line option, 18
- th
 - swap-bitcoin-refund command line option, 11
 - swap-bitcoin-withdraw command line option, 13
 - swap-bytom-refund command line option, 15
 - swap-bytom-withdraw command line option, 17
 - swap-ethereum-refund command line option, 19
 - swap-ethereum-withdraw command line option, 20
 - swap-vapor-refund command line option, 22
 - swap-vapor-withdraw command line option, 24
 - swap-xinfin-refund command line option, 26
 - swap-xinfin-withdraw command line option, 27
- tr
 - swap-bitcoin-decode command line option, 10
 - swap-bitcoin-sign command line option, 12
 - swap-bitcoin-submit command line option, 13
 - swap-bytom-decode command line option, 14
 - swap-bytom-sign command line option, 16
 - swap-bytom-submit command line option, 16
 - swap-ethereum-decode command line option, 17
 - swap-ethereum-sign command line option, 20
 - swap-ethereum-submit command line option, 20
 - swap-vapor-decode command line option, 21
 - swap-vapor-sign command line option, 23
 - swap-vapor-submit command line option, 24
 - swap-xinfin-decode command line option, 24
 - swap-xinfin-sign command line option, 27
 - swap-xinfin-submit command line option, 27
- u
 - swap-bitcoin-fund command line option, 10
 - swap-bytom-fund command line option, 14
 - swap-ethereum-fund command line option, 18
 - swap-vapor-fund command line option, 21
 - swap-xinfin-fund command line option, 25
- v
 - swap command line option, 9
 - swap-bitcoin-fund command line option, 10
 - swap-bitcoin-refund command line option,

11
 swap-bitcoin-sign command line option, 12
 swap-bitcoin-withdraw command line
 option, 13
 -x20
 swap-xinfin-fund command line option, 25
 swap-xinfin-htlc command line option, 26
 swap-xinfin-refund command line option,
 26
 swap-xinfin-withdraw command line
 option, 28
 -xpk
 swap-bitcoin-sign command line option, 12
 swap-bytom-sign command line option, 16
 swap-ethereum-sign command line option,
 20
 swap-vapor-sign command line option, 23
 swap-xinfin-sign command line option, 27

A

abi() (*swap.providers.ethereum.htlc.HTLC method*),
 115
 abi() (*swap.providers.xinfin.htlc.HTLC method*), 191
 account_create() (in module
swap.providers.bytom.rpc), 97
 account_create() (in module
swap.providers.vapor.rpc), 173
 address() (*swap.providers.bitcoin.wallet.Wallet
 method*), 40
 address() (*swap.providers.bytom.wallet.Wallet
 method*), 73
 address() (*swap.providers.ethereum.wallet.Wallet
 method*), 112
 address() (*swap.providers.vapor.wallet.Wallet method*),
 149
 address() (*swap.providers.xinfin.wallet.Wallet method*),
 188
 amount_unit_converter() (in module
swap.providers.bitcoin.utils), 65
 amount_unit_converter() (in module
swap.providers.bytom.utils), 103
 amount_unit_converter() (in module
swap.providers.ethereum.utils), 140
 amount_unit_converter() (in module
swap.providers.vapor.utils), 178
 amount_unit_converter() (in module
swap.providers.xinfin.utils), 216

B

balance() (*swap.providers.bitcoin.htlc.HTLC method*),
 43
 balance() (*swap.providers.bitcoin.wallet.Wallet
 method*), 40

balance() (*swap.providers.bytom.htlc.HTLC method*),
 76
 balance() (*swap.providers.bytom.wallet.Wallet
 method*), 74
 balance() (*swap.providers.ethereum.htlc.HTLC
 method*), 118
 balance() (*swap.providers.ethereum.wallet.Wallet
 method*), 112
 balance() (*swap.providers.vapor.htlc.HTLC method*),
 152
 balance() (*swap.providers.vapor.wallet.Wallet method*),
 150
 balance() (*swap.providers.xinfin.htlc.HTLC method*),
 194
 balance() (*swap.providers.xinfin.wallet.Wallet method*),
 188
 build_htlc() (*swap.providers.bitcoin.htlc.HTLC
 method*), 41
 build_htlc() (*swap.providers.bytom.htlc.HTLC
 method*), 75
 build_htlc() (*swap.providers.ethereum.htlc.HTLC
 method*), 115
 build_htlc() (*swap.providers.vapor.htlc.HTLC
 method*), 151
 build_htlc() (*swap.providers.xinfin.htlc.HTLC
 method*), 191
 build_transaction() (in module
swap.providers.bytom.rpc), 98
 build_transaction() (in module
swap.providers.vapor.rpc), 174
 build_transaction()
 (*swap.providers.bitcoin.transaction.FundTransaction
 method*), 48
 build_transaction()
 (*swap.providers.bitcoin.transaction.NormalTransaction
 method*), 46
 build_transaction()
 (*swap.providers.bitcoin.transaction.RefundTransaction
 method*), 50
 build_transaction()
 (*swap.providers.bitcoin.transaction.WithdrawTransaction
 method*), 49
 build_transaction()
 (*swap.providers.bytom.transaction.FundTransaction
 method*), 82
 build_transaction()
 (*swap.providers.bytom.transaction.NormalTransaction
 method*), 81
 build_transaction()
 (*swap.providers.bytom.transaction.RefundTransaction
 method*), 85
 build_transaction()
 (*swap.providers.bytom.transaction.WithdrawTransaction
 method*), 84

build_transaction() (swap.providers.ethereum.htlc.HTLC method), 113	bytecode_runtime() (swap.providers.xinfin.htlc.HTLC method), 193
build_transaction() (swap.providers.ethereum.transaction.FundTransaction method), 123	C
build_transaction() (swap.providers.ethereum.transaction.NormalTransaction method), 121	chain_code() (swap.providers.bitcoin.wallet.Wallet method), 39
build_transaction() (swap.providers.ethereum.transaction.RefundTransaction method), 125	chain_code() (swap.providers.ethereum.wallet.Wallet method), 111
build_transaction() (swap.providers.ethereum.transaction.WithdrawTransaction method), 124	chain_code() (swap.providers.xinfin.wallet.Wallet method), 187
build_transaction() (swap.providers.vapor.transaction.FundTransaction method), 158	child_xprivate_key() (swap.providers.bytom.wallet.Wallet method), 72
build_transaction() (swap.providers.vapor.transaction.NormalTransaction method), 157	child_xprivate_key() (swap.providers.vapor.wallet.Wallet method), 148
build_transaction() (swap.providers.vapor.transaction.RefundTransaction method), 161	child_xpublic_key() (swap.providers.bytom.wallet.Wallet method), 72
build_transaction() (swap.providers.vapor.transaction.WithdrawTransaction method), 160	child_xpublic_key() (swap.providers.vapor.wallet.Wallet method), 148
build_transaction() (swap.providers.xinfin.htlc.HTLC method), 189	clean_derivation() (swap.providers.bitcoin.wallet.Wallet method), 36
build_transaction() (swap.providers.xinfin.transaction.FundTransaction method), 199	clean_derivation() (swap.providers.bytom.wallet.Wallet method), 69
build_transaction() (swap.providers.xinfin.transaction.NormalTransaction method), 198	clean_derivation() (swap.providers.ethereum.wallet.Wallet method), 108
build_transaction() (swap.providers.xinfin.transaction.RefundTransaction method), 201	clean_derivation() (swap.providers.vapor.wallet.Wallet method), 145
build_transaction() (swap.providers.xinfin.transaction.WithdrawTransaction method), 200	clean_derivation() (swap.providers.xinfin.wallet.Wallet method), 184
bytecode() (swap.providers.bitcoin.htlc.HTLC method), 42	clean_transaction_raw() (in module swap.utils), 31
bytecode() (swap.providers.bytom.htlc.HTLC method), 75	compressed() (swap.providers.bitcoin.wallet.Wallet method), 39
bytecode() (swap.providers.ethereum.htlc.HTLC method), 116	compressed() (swap.providers.ethereum.wallet.Wallet method), 111
bytecode() (swap.providers.vapor.htlc.HTLC method), 151	compressed() (swap.providers.xinfin.wallet.Wallet method), 186
bytecode() (swap.providers.xinfin.htlc.HTLC method), 192	contract_address() (swap.providers.bitcoin.htlc.HTLC method), 43
bytecode_runtime() (swap.providers.ethereum.htlc.HTLC method), 117	contract_address() (swap.providers.bytom.htlc.HTLC method), 76
	contract_address() (swap.providers.ethereum.htlc.HTLC method), 115
	contract_address() (swap.providers.vapor.htlc.HTLC method), 152
	contract_address() (swap.providers.xinfin.htlc.HTLC method), 191
	D
	decode_raw() (in module swap.providers.bitcoin.rpc), 62

<code>decode_raw()</code> (in module <code>swap.providers.bytom.rpc</code>), 100	<code>fee()</code> (<code>swap.providers.bitcoin.transaction.Transaction</code> method), 44
<code>decode_raw()</code> (in module <code>swap.providers.ethereum.rpc</code>), 138	<code>fee()</code> (<code>swap.providers.bytom.signature.Signature</code> method), 89
<code>decode_raw()</code> (in module <code>swap.providers.vapor.rpc</code>), 176	<code>fee()</code> (<code>swap.providers.bytom.transaction.Transaction</code> method), 78
<code>decode_raw()</code> (in module <code>swap.providers.xinfin.rpc</code>), 213	<code>fee()</code> (<code>swap.providers.ethereum.htlc.HTLC</code> method), 114
<code>decode_transaction_raw()</code> (in module <code>swap.providers.bitcoin.utils</code>), 65	<code>fee()</code> (<code>swap.providers.ethereum.signature.Signature</code> method), 129
<code>decode_transaction_raw()</code> (in module <code>swap.providers.bytom.utils</code>), 103	<code>fee()</code> (<code>swap.providers.ethereum.transaction.Transaction</code> method), 118
<code>decode_transaction_raw()</code> (in module <code>swap.providers.ethereum.utils</code>), 140	<code>fee()</code> (<code>swap.providers.vapor.signature.Signature</code> method), 165
<code>decode_transaction_raw()</code> (in module <code>swap.providers.vapor.utils</code>), 179	<code>fee()</code> (<code>swap.providers.vapor.transaction.Transaction</code> method), 154
<code>decode_transaction_raw()</code> (in module <code>swap.providers.xinfin.utils</code>), 215	<code>fee()</code> (<code>swap.providers.xinfin.htlc.HTLC</code> method), 190
<code>double_sha256()</code> (in module <code>swap.utils</code>), 31	<code>fee()</code> (<code>swap.providers.xinfin.signature.Signature</code> method), 204
E	<code>fee()</code> (<code>swap.providers.xinfin.transaction.Transaction</code> method), 194
<code>entropy()</code> (<code>swap.providers.bitcoin.wallet.Wallet</code> method), 36	<code>fee_calculator()</code> (in module <code>swap.providers.bitcoin.utils</code>), 64
<code>entropy()</code> (<code>swap.providers.bytom.wallet.Wallet</code> method), 70	<code>find_p2sh_utxo()</code> (in module <code>swap.providers.bitcoin.rpc</code>), 62
<code>entropy()</code> (<code>swap.providers.ethereum.wallet.Wallet</code> method), 108	<code>find_p2wsh_utxo()</code> (in module <code>swap.providers.bytom.rpc</code>), 100
<code>entropy()</code> (<code>swap.providers.vapor.wallet.Wallet</code> method), 146	<code>find_p2wsh_utxo()</code> (in module <code>swap.providers.vapor.rpc</code>), 176
<code>entropy()</code> (<code>swap.providers.xinfin.wallet.Wallet</code> method), 184	<code>from_bytecode()</code> (<code>swap.providers.bitcoin.htlc.HTLC</code> method), 42
<code>entropy_to_mnemonic()</code> (in module <code>swap.utils</code>), 31	<code>from_bytecode()</code> (<code>swap.providers.bytom.htlc.HTLC</code> method), 75
<code>erc20_balance()</code> (<code>swap.providers.ethereum.htlc.HTLC</code> method), 118	<code>from_bytecode()</code> (<code>swap.providers.vapor.htlc.HTLC</code> method), 151
<code>erc20_balance()</code> (<code>swap.providers.ethereum.wallet.Wallet</code> method), 112	<code>from_entropy()</code> (<code>swap.providers.bitcoin.wallet.Wallet</code> method), 33
<code>estimate_endblock()</code> (in module <code>swap.providers.bytom.utils</code>), 103	<code>from_entropy()</code> (<code>swap.providers.bytom.wallet.Wallet</code> method), 67
<code>estimate_endblock()</code> (in module <code>swap.providers.vapor.utils</code>), 179	<code>from_entropy()</code> (<code>swap.providers.ethereum.wallet.Wallet</code> method), 105
<code>estimate_transaction_fee()</code> (in module <code>swap.providers.bytom.rpc</code>), 97	<code>from_entropy()</code> (<code>swap.providers.vapor.wallet.Wallet</code> method), 143
<code>estimate_transaction_fee()</code> (in module <code>swap.providers.vapor.rpc</code>), 173	<code>from_entropy()</code> (<code>swap.providers.xinfin.wallet.Wallet</code> method), 181
<code>expand_xprivate_key()</code> (<code>swap.providers.bytom.wallet.Wallet</code> method), 72	<code>from_index()</code> (<code>swap.providers.bitcoin.wallet.Wallet</code> method), 35
<code>expand_xprivate_key()</code> (<code>swap.providers.vapor.wallet.Wallet</code> method), 148	<code>from_index()</code> (<code>swap.providers.bytom.wallet.Wallet</code> method), 69
F	<code>from_index()</code> (<code>swap.providers.ethereum.wallet.Wallet</code> method), 107
<code>fee()</code> (<code>swap.providers.bitcoin.signature.Signature</code> method), 54	<code>from_index()</code> (<code>swap.providers.vapor.wallet.Wallet</code> method), 145
	<code>from_index()</code> (<code>swap.providers.xinfin.wallet.Wallet</code> method), 194

method), 183
 from_indexes() (swap.providers.bytom.wallet.Wallet method), 69
 from_indexes() (swap.providers.vapor.wallet.Wallet method), 145
 from_mnemonic() (swap.providers.bitcoin.wallet.Wallet method), 33
 from_mnemonic() (swap.providers.bytom.wallet.Wallet method), 67
 from_mnemonic() (swap.providers.ethereum.wallet.Wallet method), 106
 from_mnemonic() (swap.providers.vapor.wallet.Wallet method), 143
 from_mnemonic() (swap.providers.xinfin.wallet.Wallet method), 181
 from_opcode() (swap.providers.bitcoin.htlc.HTLC method), 42
 from_path() (swap.providers.bitcoin.wallet.Wallet method), 35
 from_path() (swap.providers.bytom.wallet.Wallet method), 68
 from_path() (swap.providers.ethereum.wallet.Wallet method), 107
 from_path() (swap.providers.vapor.wallet.Wallet method), 144
 from_path() (swap.providers.xinfin.wallet.Wallet method), 183
 from_private_key() (swap.providers.bitcoin.wallet.Wallet method), 35
 from_private_key() (swap.providers.bytom.wallet.Wallet method), 68
 from_private_key() (swap.providers.ethereum.wallet.Wallet method), 107
 from_private_key() (swap.providers.vapor.wallet.Wallet method), 144
 from_private_key() (swap.providers.xinfin.wallet.Wallet method), 183
 from_seed() (swap.providers.bitcoin.wallet.Wallet method), 34
 from_seed() (swap.providers.bytom.wallet.Wallet method), 68
 from_seed() (swap.providers.ethereum.wallet.Wallet method), 106
 from_seed() (swap.providers.vapor.wallet.Wallet method), 144
 from_seed() (swap.providers.xinfin.wallet.Wallet method), 182
 from_wif() (swap.providers.bitcoin.wallet.Wallet method), 35
 from_wif() (swap.providers.ethereum.wallet.Wallet method), 107
 from_wif() (swap.providers.xinfin.wallet.Wallet method), 183
 from_xprivate_key() (swap.providers.bitcoin.wallet.Wallet method), 34
 from_xprivate_key() (swap.providers.bytom.wallet.Wallet method), 68
 from_xprivate_key() (swap.providers.ethereum.wallet.Wallet method), 106
 from_xprivate_key() (swap.providers.vapor.wallet.Wallet method), 144
 from_xprivate_key() (swap.providers.xinfin.wallet.Wallet method), 182
 from_xpublic_key() (swap.providers.bitcoin.wallet.Wallet method), 34
 from_xpublic_key() (swap.providers.ethereum.wallet.Wallet method), 106
 from_xpublic_key() (swap.providers.xinfin.wallet.Wallet method), 182
 FundSignature (class in swap.providers.bitcoin.signature), 58
 FundSignature (class in swap.providers.bytom.signature), 94
 FundSignature (class in swap.providers.ethereum.signature), 133
 FundSignature (class in swap.providers.vapor.signature), 170
 FundSignature (class in swap.providers.xinfin.signature), 208
 FundSolver (class in swap.providers.bitcoin.solver), 52
 FundSolver (class in swap.providers.bytom.solver), 87
 FundSolver (class in swap.providers.ethereum.solver), 127
 FundSolver (class in swap.providers.vapor.solver), 163
 FundSolver (class in swap.providers.xinfin.solver), 203
 FundTransaction (class in swap.providers.bitcoin.transaction), 47
 FundTransaction (class in swap.providers.bytom.transaction), 82
 FundTransaction (class in swap.providers.ethereum.transaction), 122
 FundTransaction (class in swap.providers.vapor.transaction), 158
 FundTransaction (class in swap.providers.xinfin.transaction), 198

G

generate_entropy() (in module swap.utils), 29
 generate_mnemonic() (in module swap.utils), 29
 generate_passphrase() (in module swap.utils), 29
 get_address_hash() (in module swap.providers.bitcoin.utils), 66

get_address_type()	(in module swap.providers.bitcoin.utils), 64	148
get_address_type()	(in module swap.providers.bytom.utils), 102	
get_address_type()	(in module swap.providers.vapor.utils), 178	
get_balance()	(in module swap.providers.bitcoin.rpc), 61	
get_balance()	(in module swap.providers.bytom.rpc), 96	
get_balance()	(in module swap.providers.ethereum.rpc), 135	
get_balance()	(in module swap.providers.vapor.rpc), 172	
get_balance()	(in module swap.providers.xinfin.rpc), 211	
get_current_block_height()	(in module swap.providers.bytom.rpc), 99	
get_current_block_height()	(in module swap.providers.vapor.rpc), 176	
get_current_timestamp()	(in module swap.utils), 29	
get_entropy_strength()	(in module swap.utils), 30	
get_erc20_balance()	(in module swap.providers.ethereum.rpc), 136	
get_erc20_decimals()	(in module swap.providers.ethereum.rpc), 136	
get_mnemonic_language()	(in module swap.utils), 30	
get_mnemonic_strength()	(in module swap.utils), 30	
get_transaction()	(in module swap.providers.bitcoin.rpc), 61	
get_transaction()	(in module swap.providers.bytom.rpc), 99	
get_transaction()	(in module swap.providers.ethereum.rpc), 136	
get_transaction()	(in module swap.providers.vapor.rpc), 175	
get_transaction()	(in module swap.providers.xinfin.rpc), 212	
get_transaction_receipt()	(in module swap.providers.ethereum.rpc), 137	
get_transaction_receipt()	(in module swap.providers.xinfin.rpc), 212	
get_utxos()	(in module swap.providers.bitcoin.rpc), 61	
get_utxos()	(in module swap.providers.bytom.rpc), 96	
get_utxos()	(in module swap.providers.vapor.rpc), 172	
get_web3()	(in module swap.providers.ethereum.rpc), 135	
get_web3()	(in module swap.providers.xinfin.rpc), 211	
get_xrc20_balance()	(in module swap.providers.xinfin.rpc), 211	
get_xrc20_decimals()	(in module swap.providers.xinfin.rpc), 211	
guid()	(swap.providers.bytom.wallet.Wallet method), 72	
guid()	(swap.providers.vapor.wallet.Wallet method),	

H

hash()	(swap.providers.bitcoin.htlc.HTLC method), 43
hash()	(swap.providers.bitcoin.signature.Signature method), 55
hash()	(swap.providers.bitcoin.transaction.Transaction method), 45
hash()	(swap.providers.bitcoin.wallet.Wallet method), 40
hash()	(swap.providers.bytom.htlc.HTLC method), 76
hash()	(swap.providers.bytom.signature.Signature method), 90
hash()	(swap.providers.bytom.transaction.Transaction method), 78
hash()	(swap.providers.ethereum.htlc.HTLC method), 114
hash()	(swap.providers.ethereum.signature.Signature method), 129
hash()	(swap.providers.ethereum.transaction.Transaction method), 119
hash()	(swap.providers.ethereum.wallet.Wallet method), 112
hash()	(swap.providers.vapor.htlc.HTLC method), 152
hash()	(swap.providers.vapor.signature.Signature method), 166
hash()	(swap.providers.vapor.transaction.Transaction method), 154
hash()	(swap.providers.xinfin.htlc.HTLC method), 190
hash()	(swap.providers.xinfin.signature.Signature method), 205
hash()	(swap.providers.xinfin.transaction.Transaction method), 195
hash()	(swap.providers.xinfin.wallet.Wallet method), 188
HTLC	(class in swap.providers.bitcoin.htlc), 41
HTLC	(class in swap.providers.bytom.htlc), 74
HTLC	(class in swap.providers.ethereum.htlc), 113
HTLC	(class in swap.providers.vapor.htlc), 150
HTLC	(class in swap.providers.xinfin.htlc), 189

I

indexes()	(swap.providers.bytom.wallet.Wallet method), 71
indexes()	(swap.providers.vapor.wallet.Wallet method), 147
is_address()	(in module swap.providers.bitcoin.utils), 64
is_address()	(in module swap.providers.bytom.utils), 102
is_address()	(in module swap.providers.ethereum.utils), 139
is_address()	(in module swap.providers.vapor.utils), 178

[is_address\(\)](#) (in module [swap.providers.xinfin.utils](#)), 214
[is_checksum_address\(\)](#) (in module [swap.providers.ethereum.utils](#)), 139
[is_checksum_address\(\)](#) (in module [swap.providers.xinfin.utils](#)), 214
[is_entropy\(\)](#) (in module [swap.utils](#)), 30
[is_mnemonic\(\)](#) (in module [swap.utils](#)), 30
[is_network\(\)](#) (in module [swap.providers.bitcoin.utils](#)), 64
[is_network\(\)](#) (in module [swap.providers.bytom.utils](#)), 102
[is_network\(\)](#) (in module [swap.providers.ethereum.utils](#)), 139
[is_network\(\)](#) (in module [swap.providers.vapor.utils](#)), 178
[is_network\(\)](#) (in module [swap.providers.xinfin.utils](#)), 214
[is_transaction_raw\(\)](#) (in module [swap.providers.bitcoin.utils](#)), 64
[is_transaction_raw\(\)](#) (in module [swap.providers.bytom.utils](#)), 102
[is_transaction_raw\(\)](#) (in module [swap.providers.ethereum.utils](#)), 140
[is_transaction_raw\(\)](#) (in module [swap.providers.vapor.utils](#)), 178
[is_transaction_raw\(\)](#) (in module [swap.providers.xinfin.utils](#)), 215

J

[json\(\)](#) ([swap.providers.bitcoin.signature.Signature](#) method), 55
[json\(\)](#) ([swap.providers.bitcoin.transaction.Transaction](#) method), 45
[json\(\)](#) ([swap.providers.bytom.signature.Signature](#) method), 90
[json\(\)](#) ([swap.providers.bytom.transaction.Transaction](#) method), 78
[json\(\)](#) ([swap.providers.ethereum.htlc.HTLC](#) method), 114
[json\(\)](#) ([swap.providers.ethereum.signature.Signature](#) method), 129
[json\(\)](#) ([swap.providers.ethereum.transaction.Transaction](#) method), 119
[json\(\)](#) ([swap.providers.vapor.signature.Signature](#) method), 166
[json\(\)](#) ([swap.providers.vapor.transaction.Transaction](#) method), 154
[json\(\)](#) ([swap.providers.xinfin.htlc.HTLC](#) method), 190
[json\(\)](#) ([swap.providers.xinfin.signature.Signature](#) method), 205
[json\(\)](#) ([swap.providers.xinfin.transaction.Transaction](#) method), 195

L

[language\(\)](#) ([swap.providers.bitcoin.wallet.Wallet](#) method), 37
[language\(\)](#) ([swap.providers.bytom.wallet.Wallet](#) method), 70
[language\(\)](#) ([swap.providers.ethereum.wallet.Wallet](#) method), 109
[language\(\)](#) ([swap.providers.vapor.wallet.Wallet](#) method), 146
[language\(\)](#) ([swap.providers.xinfin.wallet.Wallet](#) method), 185

M

[mnemonic\(\)](#) ([swap.providers.bitcoin.wallet.Wallet](#) method), 36
[mnemonic\(\)](#) ([swap.providers.bytom.wallet.Wallet](#) method), 70
[mnemonic\(\)](#) ([swap.providers.ethereum.wallet.Wallet](#) method), 108
[mnemonic\(\)](#) ([swap.providers.vapor.wallet.Wallet](#) method), 146
[mnemonic\(\)](#) ([swap.providers.xinfin.wallet.Wallet](#) method), 184
[mnemonic_to_entropy\(\)](#) (in module [swap.utils](#)), 31
[module](#)
[swap.providers.bitcoin.htlc](#), 41
[swap.providers.bitcoin.rpc](#), 61
[swap.providers.bitcoin.signature](#), 54
[swap.providers.bitcoin.solver](#), 52
[swap.providers.bitcoin.transaction](#), 44
[swap.providers.bitcoin.utils](#), 64
[swap.providers.bitcoin.wallet](#), 33
[swap.providers.bytom.htlc](#), 74
[swap.providers.bytom.rpc](#), 96
[swap.providers.bytom.signature](#), 89
[swap.providers.bytom.solver](#), 87
[swap.providers.bytom.transaction](#), 78
[swap.providers.bytom.utils](#), 102
[swap.providers.bytom.wallet](#), 67
[swap.providers.ethereum.htlc](#), 113
[swap.providers.ethereum.rpc](#), 135
[swap.providers.ethereum.signature](#), 128
[swap.providers.ethereum.solver](#), 126
[swap.providers.ethereum.transaction](#), 118
[swap.providers.ethereum.utils](#), 139
[swap.providers.ethereum.wallet](#), 105
[swap.providers.vapor.htlc](#), 150
[swap.providers.vapor.rpc](#), 172
[swap.providers.vapor.signature](#), 165
[swap.providers.vapor.solver](#), 163
[swap.providers.vapor.transaction](#), 154
[swap.providers.vapor.utils](#), 178
[swap.providers.vapor.wallet](#), 143
[swap.providers.xinfin.htlc](#), 189

swap.providers.xinfin.rpc, 211
swap.providers.xinfin.signature, 204
swap.providers.xinfin.solver, 202
swap.providers.xinfin.transaction, 194
swap.providers.xinfin.utils, 214
swap.providers.xinfin.wallet, 181
swap.utils, 29

N

NestedNamespace (class in swap.utils), 29
NormalSignature (class in swap.providers.bitcoin.signature), 58
NormalSignature (class in swap.providers.bytom.signature), 93
NormalSignature (class in swap.providers.ethereum.signature), 132
NormalSignature (class in swap.providers.vapor.signature), 169
NormalSignature (class in swap.providers.xinfin.signature), 208
NormalSolver (class in swap.providers.bitcoin.solver), 52
NormalSolver (class in swap.providers.bytom.solver), 87
NormalSolver (class in swap.providers.ethereum.solver), 126
NormalSolver (class in swap.providers.vapor.solver), 163
NormalSolver (class in swap.providers.xinfin.solver), 202
NormalTransaction (class in swap.providers.bitcoin.transaction), 46
NormalTransaction (class in swap.providers.bytom.transaction), 81
NormalTransaction (class in swap.providers.ethereum.transaction), 121
NormalTransaction (class in swap.providers.vapor.transaction), 157
NormalTransaction (class in swap.providers.xinfin.transaction), 197

O

opcode() (swap.providers.bitcoin.htlc.HTLC method), 43
opcode() (swap.providers.bytom.htlc.HTLC method), 76
opcode() (swap.providers.ethereum.htlc.HTLC method), 117
opcode() (swap.providers.vapor.htlc.HTLC method), 152
opcode() (swap.providers.xinfin.htlc.HTLC method), 193

P

passphrase() (swap.providers.bitcoin.wallet.Wallet

method), 37
passphrase() (swap.providers.bytom.wallet.Wallet method), 70
passphrase() (swap.providers.ethereum.wallet.Wallet method), 109
passphrase() (swap.providers.vapor.wallet.Wallet method), 146
passphrase() (swap.providers.xinfin.wallet.Wallet method), 185
path() (swap.providers.bitcoin.wallet.Wallet method), 39
path() (swap.providers.bytom.wallet.Wallet method), 71
path() (swap.providers.ethereum.wallet.Wallet method), 111
path() (swap.providers.vapor.wallet.Wallet method), 147
path() (swap.providers.xinfin.wallet.Wallet method), 187
private_key() (swap.providers.bitcoin.wallet.Wallet method), 39
private_key() (swap.providers.bytom.wallet.Wallet method), 73
private_key() (swap.providers.ethereum.wallet.Wallet method), 111
private_key() (swap.providers.vapor.wallet.Wallet method), 149
private_key() (swap.providers.xinfin.wallet.Wallet method), 187
program() (swap.providers.bytom.wallet.Wallet method), 73
program() (swap.providers.vapor.wallet.Wallet method), 149
public_key() (swap.providers.bitcoin.wallet.Wallet method), 39
public_key() (swap.providers.bytom.wallet.Wallet method), 73
public_key() (swap.providers.ethereum.wallet.Wallet method), 111
public_key() (swap.providers.vapor.wallet.Wallet method), 149
public_key() (swap.providers.xinfin.wallet.Wallet method), 187

R

raw() (swap.providers.bitcoin.signature.Signature method), 56
raw() (swap.providers.bitcoin.transaction.Transaction method), 45
raw() (swap.providers.bytom.signature.Signature method), 91
raw() (swap.providers.bytom.transaction.Transaction method), 79
raw() (swap.providers.ethereum.htlc.HTLC method), 114

`raw()` (*swap.providers.ethereum.signature.Signature method*), 130
`raw()` (*swap.providers.ethereum.transaction.Transaction method*), 120
`raw()` (*swap.providers.vapor.signature.Signature method*), 167
`raw()` (*swap.providers.vapor.transaction.Transaction method*), 155
`raw()` (*swap.providers.xinfin.htlc.HTLC method*), 190
`raw()` (*swap.providers.xinfin.signature.Signature method*), 205
`raw()` (*swap.providers.xinfin.transaction.Transaction method*), 196
`RefundSignature` (class in *swap.providers.bitcoin.signature*), 60
`RefundSignature` (class in *swap.providers.bytom.signature*), 95
`RefundSignature` (class in *swap.providers.ethereum.signature*), 134
`RefundSignature` (class in *swap.providers.vapor.signature*), 171
`RefundSignature` (class in *swap.providers.xinfin.signature*), 210
`RefundSolver` (class in *swap.providers.bitcoin.solver*), 54
`RefundSolver` (class in *swap.providers.bytom.solver*), 88
`RefundSolver` (class in *swap.providers.ethereum.solver*), 128
`RefundSolver` (class in *swap.providers.vapor.solver*), 164
`RefundSolver` (class in *swap.providers.xinfin.solver*), 204
`RefundTransaction` (class in *swap.providers.bitcoin.transaction*), 50
`RefundTransaction` (class in *swap.providers.bytom.transaction*), 85
`RefundTransaction` (class in *swap.providers.ethereum.transaction*), 125
`RefundTransaction` (class in *swap.providers.vapor.transaction*), 161
`RefundTransaction` (class in *swap.providers.xinfin.transaction*), 201
`root_xprivate_key()` (*swap.providers.bitcoin.wallet.Wallet method*), 37
`root_xprivate_key()` (*swap.providers.ethereum.wallet.Wallet method*), 109
`root_xprivate_key()` (*swap.providers.xinfin.wallet.Wallet method*), 185
`root_xpublic_key()` (*swap.providers.bitcoin.wallet.Wallet method*), 37
`root_xpublic_key()` (*swap.providers.ethereum.wallet.Wallet method*), 109
`root_xpublic_key()` (*swap.providers.xinfin.wallet.Wallet method*), 185

S

`seed()` (*swap.providers.bitcoin.wallet.Wallet method*), 37
`seed()` (*swap.providers.bytom.wallet.Wallet method*), 71
`seed()` (*swap.providers.ethereum.wallet.Wallet method*), 109
`seed()` (*swap.providers.vapor.wallet.Wallet method*), 147
`seed()` (*swap.providers.xinfin.wallet.Wallet method*), 185
`sha256()` (in module *swap.utils*), 31
`sign()` (*swap.providers.bitcoin.signature.FundSignature method*), 59
`sign()` (*swap.providers.bitcoin.signature.NormalSignature method*), 58
`sign()` (*swap.providers.bitcoin.signature.RefundSignature method*), 60
`sign()` (*swap.providers.bitcoin.signature.Signature method*), 57
`sign()` (*swap.providers.bitcoin.signature.WithdrawSignature method*), 59
`sign()` (*swap.providers.bitcoin.transaction.FundTransaction method*), 48
`sign()` (*swap.providers.bitcoin.transaction.NormalTransaction method*), 47
`sign()` (*swap.providers.bitcoin.transaction.RefundTransaction method*), 51
`sign()` (*swap.providers.bitcoin.transaction.WithdrawTransaction method*), 49
`sign()` (*swap.providers.bytom.signature.FundSignature method*), 94
`sign()` (*swap.providers.bytom.signature.NormalSignature method*), 93
`sign()` (*swap.providers.bytom.signature.RefundSignature method*), 95
`sign()` (*swap.providers.bytom.signature.Signature method*), 91
`sign()` (*swap.providers.bytom.signature.WithdrawSignature method*), 94
`sign()` (*swap.providers.bytom.transaction.FundTransaction method*), 83
`sign()` (*swap.providers.bytom.transaction.NormalTransaction method*), 81
`sign()` (*swap.providers.bytom.transaction.RefundTransaction method*), 86
`sign()` (*swap.providers.bytom.transaction.WithdrawTransaction method*), 84
`sign()` (*swap.providers.ethereum.signature.FundSignature method*), 133

<code>sign()</code> (<code>swap.providers.ethereum.signature.NormalSignature</code> method), 132	<code>sign_transaction()</code> (<code>swap.providers.xinfin.htlc.HTLC</code> method), 189
<code>sign()</code> (<code>swap.providers.ethereum.signature.RefundSignature</code> method), 134	<code>Signature</code> (class in <code>swap.providers.bitcoin.signature</code>), 54
<code>sign()</code> (<code>swap.providers.ethereum.signature.Signature</code> method), 130	<code>Signature</code> (class in <code>swap.providers.bytom.signature</code>), 89
<code>sign()</code> (<code>swap.providers.ethereum.signature.WithdrawSignature</code> method), 134	<code>Signature</code> (class in <code>swap.providers.ethereum.signature</code>), 128
<code>sign()</code> (<code>swap.providers.ethereum.transaction.FundTransaction</code> method), 123	<code>Signature</code> (class in <code>swap.providers.vapor.signature</code>), 165
<code>sign()</code> (<code>swap.providers.ethereum.transaction.NormalTransaction</code> method), 122	<code>Signature</code> (class in <code>swap.providers.xinfin.signature</code>), 204
<code>sign()</code> (<code>swap.providers.ethereum.transaction.RefundTransaction</code> method), 126	<code>signature()</code> (<code>swap.providers.ethereum.signature.Signature</code> method), 131
<code>sign()</code> (<code>swap.providers.ethereum.transaction.WithdrawTransaction</code> method), 124	<code>signature()</code> (<code>swap.providers.ethereum.transaction.Transaction</code> method), 120
<code>sign()</code> (<code>swap.providers.vapor.signature.FundSignature</code> method), 170	<code>signature()</code> (<code>swap.providers.xinfin.signature.Signature</code> method), 207
<code>sign()</code> (<code>swap.providers.vapor.signature.NormalSignature</code> method), 169	<code>signature()</code> (<code>swap.providers.xinfin.transaction.Transaction</code> method), 196
<code>sign()</code> (<code>swap.providers.vapor.signature.RefundSignature</code> method), 171	<code>signatures()</code> (<code>swap.providers.bytom.signature.Signature</code> method), 92
<code>sign()</code> (<code>swap.providers.vapor.signature.Signature</code> method), 167	<code>signatures()</code> (<code>swap.providers.bytom.transaction.Transaction</code> method), 80
<code>sign()</code> (<code>swap.providers.vapor.signature.WithdrawSignature</code> method), 170	<code>signatures()</code> (<code>swap.providers.vapor.signature.Signature</code> method), 168
<code>sign()</code> (<code>swap.providers.vapor.transaction.FundTransaction</code> method), 159	<code>signatures()</code> (<code>swap.providers.vapor.transaction.Transaction</code> method), 156
<code>sign()</code> (<code>swap.providers.vapor.transaction.NormalTransaction</code> method), 157	<code>strength()</code> (<code>swap.providers.bitcoin.wallet.Wallet</code> method), 36
<code>sign()</code> (<code>swap.providers.vapor.transaction.RefundTransaction</code> method), 162	<code>strength()</code> (<code>swap.providers.bytom.wallet.Wallet</code> method), 70
<code>sign()</code> (<code>swap.providers.vapor.transaction.WithdrawTransaction</code> method), 160	<code>strength()</code> (<code>swap.providers.ethereum.wallet.Wallet</code> method), 108
<code>sign()</code> (<code>swap.providers.xinfin.signature.FundSignature</code> method), 209	<code>strength()</code> (<code>swap.providers.vapor.wallet.Wallet</code> method), 146
<code>sign()</code> (<code>swap.providers.xinfin.signature.NormalSignature</code> method), 208	<code>strength()</code> (<code>swap.providers.xinfin.wallet.Wallet</code> method), 184
<code>sign()</code> (<code>swap.providers.xinfin.signature.RefundSignature</code> method), 210	<code>submit_raw()</code> (in module <code>swap.providers.bitcoin.rpc</code>), 63
<code>sign()</code> (<code>swap.providers.xinfin.signature.Signature</code> method), 206	<code>submit_raw()</code> (in module <code>swap.providers.bytom.rpc</code>), 101
<code>sign()</code> (<code>swap.providers.xinfin.signature.WithdrawSignature</code> method), 209	<code>submit_raw()</code> (in module <code>swap.providers.ethereum.rpc</code>), 138
<code>sign()</code> (<code>swap.providers.xinfin.transaction.FundTransaction</code> method), 199	<code>submit_raw()</code> (in module <code>swap.providers.vapor.rpc</code>), 177
<code>sign()</code> (<code>swap.providers.xinfin.transaction.NormalTransaction</code> method), 198	<code>submit_raw()</code> (in module <code>swap.providers.xinfin.rpc</code>), 214
<code>sign()</code> (<code>swap.providers.xinfin.transaction.RefundTransaction</code> method), 201	<code>submit_transaction_raw()</code> (in module <code>swap.providers.bitcoin.utils</code>), 66
<code>sign()</code> (<code>swap.providers.xinfin.transaction.WithdrawTransaction</code> method), 200	<code>submit_transaction_raw()</code> (in module <code>swap.providers.bytom.utils</code>), 104
<code>sign_transaction()</code> (<code>swap.providers.ethereum.htlc.HTLC</code> method), 113	<code>submit_transaction_raw()</code> (in module <code>swap.providers.ethereum.utils</code>), 140
	<code>submit_transaction_raw()</code> (in module)

```

    swap.providers.vapor.utils), 179
submit_transaction_raw()      (in
    swap.providers.xinfin.utils), 215
swap command line option
    --version, 9
    -v, 9
swap.providers.bitcoin.htlc
    module, 41
swap.providers.bitcoin.rpc
    module, 61
swap.providers.bitcoin.signature
    module, 54
swap.providers.bitcoin.solver
    module, 52
swap.providers.bitcoin.transaction
    module, 44
swap.providers.bitcoin.utils
    module, 64
swap.providers.bitcoin.wallet
    module, 33
swap.providers.bytom.htlc
    module, 74
swap.providers.bytom.rpc
    module, 96
swap.providers.bytom.signature
    module, 89
swap.providers.bytom.solver
    module, 87
swap.providers.bytom.transaction
    module, 78
swap.providers.bytom.utils
    module, 102
swap.providers.bytom.wallet
    module, 67
swap.providers.ethereum.htlc
    module, 113
swap.providers.ethereum.rpc
    module, 135
swap.providers.ethereum.signature
    module, 128
swap.providers.ethereum.solver
    module, 126
swap.providers.ethereum.transaction
    module, 118
swap.providers.ethereum.utils
    module, 139
swap.providers.ethereum.wallet
    module, 105
swap.providers.vapor.htlc
    module, 150
swap.providers.vapor.rpc
    module, 172
swap.providers.vapor.signature
    module, 165
swap.providers.vapor.solver
    module, 163
swap.providers.vapor.transaction
    module, 154
swap.providers.vapor.utils
    module, 178
swap.providers.vapor.wallet
    module, 143
swap.providers.xinfin.htlc
    module, 189
swap.providers.xinfin.rpc
    module, 211
swap.providers.xinfin.signature
    module, 204
swap.providers.xinfin.solver
    module, 202
swap.providers.xinfin.transaction
    module, 194
swap.providers.xinfin.utils
    module, 214
swap.providers.xinfin.wallet
    module, 181
swap.utils
    module, 29
swap-bitcoin-decode command line option
    --indent <indent>, 10
    --offline <offline>, 10
    --transaction-raw <transaction_raw>, 10
    -i, 10
    -o, 10
    -tr, 10
swap-bitcoin-fund command line option
    --address <address>, 10
    --amount <amount>, 10
    --contract-address <contract_address>, 10
    --network <network>, 10
    --unit <unit>, 10
    --version <version>, 10
    -a, 10
    -am, 10
    -ca, 10
    -n, 10
    -u, 10
    -v, 10
swap-bitcoin-htlc command line option
    --endtime <endtime>, 11
    --indent <indent>, 11
    --network <network>, 11
    --recipient-address <recipient_address>,
        11
    --secret-hash <secret_hash>, 11
    --sender-address <sender_address>, 11
    -e, 11
    -i, 11

```

```
-n, 11
-ra, 11
-sa, 11
-sh, 11
swap-bitcoin-refund command line option
--address <address>, 11
--network <network>, 11
--transaction-hash <transaction_hash>, 11
--version <version>, 11
-a, 11
-n, 11
-th, 11
-v, 11
swap-bitcoin-sign command line option
--account <account>, 12
--address <address>, 12
--bytecode <bytecode>, 12
--change <change>, 12
--endtime <endtime>, 12
--path <path>, 12
--secret-key <secret_key>, 12
--transaction-raw <transaction_raw>, 12
--version <version>, 12
--xprivate-key <xprivate_key>, 12
-ac, 12
-ad, 12
-b, 12
-ch, 12
-e, 12
-p, 12
-sk, 12
-tr, 12
-v, 12
-xpk, 12
swap-bitcoin-submit command line option
--endpoint <endpoint>, 13
--transaction-raw <transaction_raw>, 13
-e, 13
-tr, 13
swap-bitcoin-withdraw command line option
--address <address>, 13
--network <network>, 13
--transaction-hash <transaction_hash>, 13
--version <version>, 13
-a, 13
-n, 13
-th, 13
-v, 13
swap-bytom-decode command line option
--indent <indent>, 14
--transaction-raw <transaction_raw>, 14
-i, 14
-tr, 14
swap-bytom-fund command line option
--address <address>, 14
--amount <amount>, 14
--asset <asset>, 14
--contract-address <contract_address>, 14
--network <network>, 14
--unit <unit>, 14
-a, 14
-am, 14
-as, 14
-ca, 14
-n, 14
-u, 14
swap-bytom-htlc command line option
--endblock <endblock>, 15
--indent <indent>, 15
--network <network>, 15
--recipient-public-key
    <recipient_public_key>, 15
--secret-hash <secret_hash>, 15
--sender-public-key <sender_public_key>,
    15
-e, 15
-i, 15
-n, 15
-rpk, 15
-sh, 15
-spk, 15
swap-bytom-refund command line option
--address <address>, 15
--asset <asset>, 15
--network <network>, 15
--transaction-hash <transaction_hash>, 15
-a, 15
-as, 15
-n, 15
-th, 15
swap-bytom-sign command line option
--account <account>, 16
--address <address>, 16
--bytecode <bytecode>, 16
--change <change>, 16
--indexes <indexes>, 16
--path <path>, 16
--secret-key <secret_key>, 16
--transaction-raw <transaction_raw>, 16
--xprivate-key <xprivate_key>, 16
-ac, 16
-ad, 16
-b, 16
-ch, 16
-i, 16
-p, 16
-sk, 16
-tr, 16
```

```

    -xpk, 16
swap-bytom-submit command line option
    --transaction-raw <transaction_raw>, 16
    -tr, 16
swap-bytom-withdraw command line option
    --address <address>, 17
    --asset <asset>, 17
    --network <network>, 17
    --transaction-hash <transaction_hash>, 17
    -a, 17
    -as, 17
    -n, 17
    -th, 17
swap-ethereum-decode command line option
    --indent <indent>, 17
    --transaction-raw <transaction_raw>, 17
    -i, 17
    -tr, 17
swap-ethereum-fund command line option
    --amount <amount>, 18
    --contract-address <contract_address>, 18
    --endtime <endtime>, 18
    --erc20 <erc20>, 18
    --network <network>, 18
    --recipient-address <recipient_address>,
        18
    --secret-hash <secret_hash>, 18
    --sender-address <sender_address>, 18
    --token-address <token_address>, 18
    --unit <unit>, 18
    -am, 18
    -ca, 18
    -e, 18
    -e20, 18
    -n, 18
    -ra, 18
    -sa, 18
    -sh, 18
    -ta, 18
    -u, 18
swap-ethereum-htlc command line option
    --contract-address <contract_address>, 19
    --erc20 <erc20>, 19
    --indent <indent>, 19
    --network <network>, 19
    -ca, 19
    -e20, 19
    -i, 19
    -n, 19
swap-ethereum-refund command line option
    --address <address>, 19
    --contract-address <contract_address>, 19
    --erc20 <erc20>, 19
    --network <network>, 19
    --transaction-hash <transaction_hash>, 19
    -a, 19
    -ca, 19
    -e20, 19
    -n, 19
    -th, 19
swap-ethereum-sign command line option
    --account <account>, 20
    --address <address>, 20
    --change <change>, 20
    --path <path>, 20
    --transaction-raw <transaction_raw>, 20
    --xprivate-key <xprivate_key>, 20
    -ac, 20
    -ad, 20
    -ch, 20
    -p, 20
    -tr, 20
    -xpk, 20
swap-ethereum-submit command line option
    --transaction-raw <transaction_raw>, 20
    -tr, 20
swap-ethereum-withdraw command line option
    --address <address>, 20
    --contract-address <contract_address>, 20
    --erc20 <erc20>, 21
    --network <network>, 20
    --secret-key <secret_key>, 20
    --transaction-hash <transaction_hash>, 20
    -a, 20
    -ca, 20
    -e20, 21
    -n, 20
    -sk, 20
    -th, 20
swap-vapor-decode command line option
    --indent <indent>, 21
    --transaction-raw <transaction_raw>, 21
    -i, 21
    -tr, 21
swap-vapor-fund command line option
    --address <address>, 21
    --amount <amount>, 21
    --asset <asset>, 21
    --contract-address <contract_address>, 21
    --network <network>, 22
    --unit <unit>, 21
    -a, 21
    -am, 21
    -as, 21
    -ca, 21
    -n, 22
    -u, 21
swap-vapor-htlc command line option

```

```
--endblock <endblock>, 22
--indent <indent>, 22
--network <network>, 22
--recipient-public-key
    <recipient_public_key>, 22
--secret-hash <secret_hash>, 22
--sender-public-key <sender_public_key>,
    22
-e, 22
-i, 22
-n, 22
-rpk, 22
-sh, 22
-spk, 22
swap-vapor-refund command line option
--address <address>, 22
--asset <asset>, 22
--network <network>, 23
--transaction-hash <transaction_hash>, 22
-a, 22
-as, 22
-n, 23
-th, 22
swap-vapor-sign command line option
--account <account>, 23
--address <address>, 23
--bytecode <bytecode>, 23
--change <change>, 23
--indexes <indexes>, 23
--path <path>, 23
--secret-key <secret_key>, 23
--transaction-raw <transaction_raw>, 23
--xprivate-key <xprivate_key>, 23
-ac, 23
-ad, 23
-b, 23
-ch, 23
-i, 23
-p, 23
-sk, 23
-tr, 23
-xpk, 23
swap-vapor-submit command line option
--transaction-raw <transaction_raw>, 24
-tr, 24
swap-vapor-withdraw command line option
--address <address>, 24
--asset <asset>, 24
--network <network>, 24
--transaction-hash <transaction_hash>, 24
-a, 24
-as, 24
-n, 24
-th, 24
swap-xinfin-decode command line option
--indent <indent>, 24
--transaction-raw <transaction_raw>, 24
-i, 24
-tr, 24
swap-xinfin-fund command line option
--amount <amount>, 25
--contract-address <contract_address>, 25
--endtime <endtime>, 25
--network <network>, 25
--recipient-address <recipient_address>,
    25
--secret-hash <secret_hash>, 25
--sender-address <sender_address>, 25
--token-address <token_address>, 25
--unit <unit>, 25
--xrc20 <xrc20>, 25
-am, 25
-ca, 25
-e, 25
-n, 25
-ra, 25
-sa, 25
-sh, 25
-ta, 25
-u, 25
-x20, 25
swap-xinfin-htlc command line option
--contract-address <contract_address>, 26
--indent <indent>, 26
--network <network>, 26
--xrc20 <xrc20>, 26
-ca, 26
-i, 26
-n, 26
-x20, 26
swap-xinfin-refund command line option
--address <address>, 26
--contract-address <contract_address>, 26
--network <network>, 26
--transaction-hash <transaction_hash>, 26
--xrc20 <xrc20>, 26
-a, 26
-ca, 26
-n, 26
-th, 26
-x20, 26
swap-xinfin-sign command line option
--account <account>, 27
--address <address>, 27
--change <change>, 27
--path <path>, 27
--transaction-raw <transaction_raw>, 27
--xprivate-key <xprivate_key>, 27
```

-ac, 27
 -ad, 27
 -ch, 27
 -p, 27
 -tr, 27
 -xpk, 27
 swap-xinfin-submit command line option
 --transaction-raw <transaction_raw>, 27
 -tr, 27
 swap-xinfin-withdraw command line option
 --address <address>, 27
 --contract-address <contract_address>, 27
 --network <network>, 27
 --secret-key <secret_key>, 27
 --transaction-hash <transaction_hash>, 27
 --xrc20 <xrc20>, 28
 -a, 27
 -ca, 27
 -n, 27
 -sk, 27
 -th, 27
 -x20, 28

T

to_checksum_address() (in module *swap.providers.ethereum.utils*), 139
 to_checksum_address() (in module *swap.providers.xinfin.utils*), 215
 Transaction (class in *swap.providers.bitcoin.transaction*), 44
 Transaction (class in *swap.providers.bytom.transaction*), 78
 Transaction (class in *swap.providers.ethereum.transaction*), 118
 Transaction (class in *swap.providers.vapor.transaction*), 154
 Transaction (class in *swap.providers.xinfin.transaction*), 194
 transaction_raw() (*swap.providers.bitcoin.signature.Signature* method), 57
 transaction_raw() (*swap.providers.bitcoin.transaction.FundTransaction* method), 48
 transaction_raw() (*swap.providers.bitcoin.transaction.NormalTransaction* method), 47
 transaction_raw() (*swap.providers.bitcoin.transaction.RefundTransaction* method), 51
 transaction_raw() (*swap.providers.bitcoin.transaction.WithdrawTransaction* method), 50
 transaction_raw() (*swap.providers.bytom.signature.Signature* method), 93
 transaction_raw() (*swap.providers.bytom.transaction.FundTransaction* method), 83
 transaction_raw() (*swap.providers.bytom.transaction.NormalTransaction* method), 82
 transaction_raw() (*swap.providers.bytom.transaction.RefundTransaction* method), 86
 transaction_raw() (*swap.providers.bytom.transaction.WithdrawTransaction* method), 85
 transaction_raw() (*swap.providers.ethereum.signature.Signature* method), 131
 transaction_raw() (*swap.providers.ethereum.transaction.Transaction* method), 121
 transaction_raw() (*swap.providers.vapor.signature.Signature* method), 169
 transaction_raw() (*swap.providers.vapor.transaction.FundTransaction* method), 159
 transaction_raw() (*swap.providers.vapor.transaction.NormalTransaction* method), 158
 transaction_raw() (*swap.providers.vapor.transaction.RefundTransaction* method), 162
 transaction_raw() (*swap.providers.vapor.transaction.WithdrawTransaction* method), 161
 transaction_raw() (*swap.providers.xinfin.signature.Signature* method), 207
 transaction_raw() (*swap.providers.xinfin.transaction.Transaction* method), 197
 type() (*swap.providers.bitcoin.signature.Signature* method), 56
 type() (*swap.providers.bitcoin.transaction.Transaction* method), 46
 type() (*swap.providers.bytom.signature.Signature* method), 91
 type() (*swap.providers.bytom.transaction.Transaction* method), 79
 type() (*swap.providers.ethereum.signature.Signature* method), 130
 type() (*swap.providers.ethereum.transaction.Transaction* method), 120
 type() (*swap.providers.vapor.signature.Signature* method), 167
 type() (*swap.providers.vapor.transaction.Transaction* method), 155
 type() (*swap.providers.xinfin.signature.Signature* method), 206
 type() (*swap.providers.xinfin.transaction.Transaction* method), 196

U

uncompressed() (*swap.providers.bitcoin.wallet.Wallet* method), 38
 uncompressed() (*swap.providers.ethereum.wallet.Wallet* method), 110
 uncompressed() (*swap.providers.xinfin.wallet.Wallet* method), 186
 unsigned_data() (*swap.providers.bytom.signature.Signature* method), 92
 unsigned_data() (*swap.providers.bytom.transaction.Transaction* method), 80

unsigned_datas() (swap.providers.vapor.signature.Signature method), 168
 unsigned_datas() (swap.providers.vapor.transaction.Transaction method), 156
 utxos() (swap.providers.bitcoin.htlc.HTLC method), 44
 utxos() (swap.providers.bitcoin.wallet.Wallet method), 41
 utxos() (swap.providers.bytom.htlc.HTLC method), 77
 utxos() (swap.providers.bytom.wallet.Wallet method), 74
 utxos() (swap.providers.vapor.htlc.HTLC method), 153
 utxos() (swap.providers.vapor.wallet.Wallet method), 150

W

wait_for_transaction_receipt() (in module swap.providers.ethereum.rpc), 137
 wait_for_transaction_receipt() (in module swap.providers.xinfin.rpc), 213
 Wallet (class in swap.providers.bitcoin.wallet), 33
 Wallet (class in swap.providers.bytom.wallet), 67
 Wallet (class in swap.providers.ethereum.wallet), 105
 Wallet (class in swap.providers.vapor.wallet), 143
 Wallet (class in swap.providers.xinfin.wallet), 181
 wif() (swap.providers.bitcoin.wallet.Wallet method), 40
 wif() (swap.providers.ethereum.wallet.Wallet method), 112
 wif() (swap.providers.xinfin.wallet.Wallet method), 188
 WithdrawSignature (class in swap.providers.bitcoin.signature), 59
 WithdrawSignature (class in swap.providers.bytom.signature), 94
 WithdrawSignature (class in swap.providers.ethereum.signature), 133
 WithdrawSignature (class in swap.providers.vapor.signature), 170
 WithdrawSignature (class in swap.providers.xinfin.signature), 209
 WithdrawSolver (class in swap.providers.bitcoin.solver), 53
 WithdrawSolver (class in swap.providers.bytom.solver), 88
 WithdrawSolver (class in swap.providers.ethereum.solver), 127
 WithdrawSolver (class in swap.providers.vapor.solver), 164
 WithdrawSolver (class in swap.providers.xinfin.solver), 203
 WithdrawTransaction (class in swap.providers.bitcoin.transaction), 49
 WithdrawTransaction (class in swap.providers.bytom.transaction), 84
 WithdrawTransaction (class in swap.providers.ethereum.transaction), 124
 WithdrawTransaction (class in swap.providers.vapor.transaction), 160
 WithdrawTransaction (class in swap.providers.xinfin.transaction), 200

X

xprivate_key() (swap.providers.bitcoin.wallet.Wallet method), 38
 xprivate_key() (swap.providers.bytom.wallet.Wallet method), 71
 xprivate_key() (swap.providers.ethereum.wallet.Wallet method), 110
 xprivate_key() (swap.providers.vapor.wallet.Wallet method), 147
 xprivate_key() (swap.providers.xinfin.wallet.Wallet method), 186
 xpublic_key() (swap.providers.bitcoin.wallet.Wallet method), 38
 xpublic_key() (swap.providers.bytom.wallet.Wallet method), 71
 xpublic_key() (swap.providers.ethereum.wallet.Wallet method), 110
 xpublic_key() (swap.providers.vapor.wallet.Wallet method), 147
 xpublic_key() (swap.providers.xinfin.wallet.Wallet method), 186
 xrc20_balance() (swap.providers.xinfin.htlc.HTLC method), 194
 xrc20_balance() (swap.providers.xinfin.wallet.Wallet method), 188