
PyStratis

Release 1.1.0.0

Tjaden Froyda

Oct 24, 2021

CONTENTS:

1	pyStratis	1
1.1	API	1
1.1.1	Subpackages	1
1.1.1.1	AddressBook	1
1.1.1.2	Balances	2
1.1.1.3	BlockStore	2
1.1.1.4	ColdStaking	9
1.1.1.5	Collateral	16
1.1.1.6	CollateralVoting	17
1.1.1.7	ConnectionManager	17
1.1.1.8	Consensus	19
1.1.1.9	Dashboard	21
1.1.1.10	Diagnostic	21
1.1.1.11	Federation	24
1.1.1.12	FederationGateway	26
1.1.1.13	FederationWallet	31
1.1.1.14	Interop	34
1.1.1.15	Mempool	39
1.1.1.16	Mining	39
1.1.1.17	Multisig	40
1.1.1.18	Network	41
1.1.1.19	Node	42
1.1.1.20	Notifications	49
1.1.1.21	RPC	49
1.1.1.22	SignalR	50
1.1.1.23	SmartContracts	51
1.1.1.24	SmartContractWallet	60
1.1.1.25	Staking	64
1.1.1.26	Voting	66
1.1.1.27	Wallet	69
1.1.1.28	Global_ResponseModels	89
1.1.2	FeatureInitializationState	101
1.1.3	FullNodeState	101
1.1.4	LogRule	101
1.1.5	APIError	102
1.2	Nodes	102
1.2.1	StraxNode	102
1.2.2	CirrusNode	104
1.2.3	InterfluxStraxNode	107
1.2.4	InterfluxCirrusNode	109

1.2.5	StraxMasternode	113
1.2.6	CirrusMasternode	115
1.3	Core	118
1.3.1	Subpackages	118
1.3.1.1	Networks	118
1.3.1.2	Types	120
1.3.2	CoinType	123
1.3.3	ContractTransactionItemType	123
1.3.4	ConversionRequestStatus	124
1.3.5	ConversionRequestType	124
1.3.6	CrosschainTransferStatus	124
1.3.7	Deposit	124
1.3.8	DepositRetrievalType	125
1.3.9	DestinationChain	126
1.3.10	ExtKey	126
1.3.11	ExtPubKey	127
1.3.12	Key	127
1.3.13	MultisigSecret	128
1.3.14	Outpoint	128
1.3.15	PubKey	128
1.3.16	Recipient	129
1.3.17	SmartContractParameter	129
1.3.18	SmartContractParameterType	129
1.3.19	TransactionItemType	130
1.3.20	WalletSecret	130
2	Indices and tables	131
	Python Module Index	133
	Index	137

1.1 API

1.1.1 Subpackages

1.1.1.1 AddressBook

AddressBook

class `AddressBook(**kwargs)`

Implements the addressbook api endpoints.

add(*address: Union[str, Address], label: str, **kwargs*) → *AddressBookEntryModel*

Adds an entry to the address book.

Parameters

- **address** (*str*, *Address*) – The address to add to the address book.
- **label** (*str*) – The address label.

Returns The address book entry.

Return type *AddressBookEntryModel*

Raises *APIError* – Error thrown by node API. See message for details.

remove(*label: str, **kwargs*) → *AddressBookEntryModel*

Removes an entry from the address book.

Parameters **label** (*str*) – The label to remove.

Returns The address book entry.

Return type *AddressBookEntryModel*

Raises *APIError* – Error thrown by node API. See message for details.

__call__(*skip: Optional[int] = None, take: Optional[int] = None, **kwargs*) →
List[AddressBookEntryModel]

Gets the address book entries with option to implement pagination.

Parameters

- **skip** (*int*, *optional*) – The number of entries to skip.
- **take** (*int*, *optional*) – The maximum number of entries to take.

Returns A list of address book entries.

Return type *List*[AddressBookEntryModel]

Raises *APIError* – Error thrown by node API. See message for details.

Note: If neither skip or take arguments are specified, returns the entire address book. An address book can be accessed from a wallet, but it is a standalone feature, which is not attached to any wallet.

AddressBookEntryModel

class AddressBookEntryModel(*, address: Address, label: str)

A pydantic model representing an address book entry.

address: Address

An address validated for the current network.

label: str

A label for the given address.

1.1.1.2 Balances

Balances

class Balances(**kwargs)

Implements the balances api endpoints.

over_amount_at_height(block_height: int, amount: Union[Money, int, float, decimal.Decimal], **kwargs) → List[Address]

Returns a list of addresses with balance over specified amount at the given chain height.

Parameters

- **block_height** (int) – The specified chain height.
- **amount** (Money, int, float, Decimal) – The specified amount in coin units.

Returns A list of addresses meeting the criteria.

Return type *List*[Address]

Raises *APIError* – Error thrown by node API. See message for details.

1.1.1.3 BlockStore

BlockStore

class BlockStore(**kwargs)

Implements the blockstore api endpoints.

addressindexer_tip(**kwargs) → AddressIndexerTipModel

Retrieves the address indexer tip.

Parameters ****kwargs** – Extra keyword arguments.

Returns The address indexer tip hash and height.

Return type AddressIndexerTipModel

Raises *APIError* – Error thrown by node API. See message for details.

block(*block_hash: Union[uint256, str]*, *show_transaction_details: bool = True*, *output_json: bool = True*, ***kwargs*) → *Union[BlockModel, BlockTransactionDetailsModel, hexstr, str]*

Retrieves the block which matches the supplied block hash.

Parameters

- **block_hash** (*uint256, str*) – The hash of the required block.
- **show_transaction_details** (*bool, optional*) – A flag that indicates whether to return each block transaction complete with details or simply return transaction hashes. Default=True.
- **output_json** (*bool*) – Output json or hex block. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The representation of the block.

Return type (*BlockModel, BlockTransactionDetailsModel, hexstr, str*)

Raises *APIError* – Error thrown by node API. See message for details.

get_block_count(***kwargs*) → *int*

Gets the current block count.

Parameters ***kwargs* – Extra keyword arguments.

Returns The current block count.

Return type *int*

Raises *APIError* – Error thrown by node API. See message for details.

get_addresses_balances(*addresses: Union[List[Union[Address, str]], Address, str]*, *min_confirmations: int = 0*, ***kwargs*) → *GetAddressesBalancesModel*

Provides balance of the given addresses confirmed with at least *min_confirmations* confirmations.

Requires *addressindex=1* in node configuration.

Parameters

- **addresses** (*List(Address, str), Address, str*) – A list of addresses or single address to query.
- **min_confirmations** (*int, optional*) – Only blocks below consensus tip less this parameter will be considered. Default=0.
- ****kwargs** – Extra keyword arguments.

Returns The balances of the given address(es).

Return type *GetAddressesBalancesModel*

Raises *APIError* – Error thrown by node API. See message for details.

get_verbose_addresses_balances(*addresses: Union[List[Union[Address, str]], Address, str]*, ***kwargs*) → *GetVerboseAddressesBalancesModel*

Provides verbose balance data of the given addresses.

Parameters

- **addresses** (*List(Address, str), Address, str*) – A list of addresses or single address to query.
- ****kwargs** – Extra keyword arguments.

Returns A verbose accounting of the balances of the specified address(es).

Return type `GetVerboseAddressesBalancesModel`

Raises `APIError` – Error thrown by node API. See message for details.

`get_utxo_set(at_block_height: int, **kwargs) → List[UTXOModel]`

Gets the UTXO set at the specified block height.

Parameters

- `at_block_height` (`int`) – The specified block height.
- `**kwargs` – Extra keyword arguments.

Returns A list of UTXO at the given height.

Return type `List[UTXOModel]`

Raises `APIError` – Error thrown by node API. See message for details.

`get_last_balance_update_transaction(address: Union[Address, str], **kwargs) → Optional[GetLastBalanceUpdateTransactionModel]`

Gets the transaction information for a transaction last updating the given address.

Parameters

- `address` (`Address`) – An address to query.
- `**kwargs` – Extra keyword arguments.

Returns Returns the information on the last transaction if the transaction exists, otherwise `None`.

Return type (`GetLastBalanceUpdateTransactionModel`, `None`)

Raises `APIError` – Error thrown by node API. See message for details.

`get_utxoset_for_address(address: Union[Address, str], **kwargs) → Optional[GetUTXOsForAddressModel]`

Gets the utxoset and balance information for the given address.

Parameters

- `address` (`Address`) – An address to query.
- `**kwargs` – Extra keyword arguments.

Returns Returns the current balance and utxo set for the given address.

Return type (`GetUTXOsForAddressModel`, `None`)

Raises `APIError` – Error thrown by node API. See message for details.

AddressBalanceModel

`class AddressBalanceModel(*, address: Address, balance: Money)`

A pydantic model representing an address and balance.

address: `Address`

An address validated on the current network.

balance: `Money`

The balance in coin units.

AddressIndexerTipModel

class AddressIndexerTipModel(* , tipHash: uint256, tipHeight: int)

A pydantic model for the address indexer tip.

tip_hash: *uint256*

The block hash of the block at the address indexer tip.

tip_height: *int*

The block height of the address indexer tip.

BalanceChangesModel

class BalanceChangesModel(* , deposited: bool, satoshi: int, balanceChangedHeight: int)

A pydantic model representing changes in balance at a given address.

deposited: *bool*

If true, amount was received. False if value was withdrawn.

satoshi: *int*

The value of the amount changed, in satoshi.

balance_changed_height: *int*

The height of the block containing the transaction.

BlockModel

class BlockModel(* , hash: uint256, confirmations: int, size: int, weight: int, height: int, version: int, versionHex: str, merkleroot: hexstr, tx: List[uint256] = None, time: datetime.datetime, mediantime: datetime.datetime, nonce: int, bits: str, difficulty: float, chainwork: str, nTx: int, previousblockhash: uint256 = None, nextblockhash: uint256 = None, signature: str = None, modifyerv2: str = None, flags: str = None, hashproof: str = None, blocktrust: str = None, chaintrust: str = None)

A pydantic model of a block.

hash: *uint256*

The block hash.

confirmations: *int*

The number of confirmations.

size: *int*

The size of the block.

weight: *int*

The weight of the block.

height: *int*

The height of the block.

version: *int*

The block version.

version_hex: *str*

The block version in hex.

merkleroot: *hexstr*

The block merkleroot.

tx: `Optional[List[uint256]]`
A list of transactions in the block.

time: `datetime.datetime`
The time the block was produced.

median_time: `datetime.datetime`
The median time.

nonce: `int`
The block's nonce.

bits: `str`
The block bits.

difficulty: `float`
The block difficulty.

chainwork: `str`
The chain work.

n_tx: `int`
The number of transactions in the block.

previous_blockhash: `Optional[uint256]`
The previous block hash.

next_blockhash: `Optional[uint256]`
The next block hash.

signature: `Optional[str]`
The signature.

modifier_v2: `Optional[str]`
The block modifier.

flags: `Optional[str]`
Block flags.

hashproof: `Optional[str]`
Block hashproof.

blocktrust: `Optional[str]`
Blocktrust.

chaintrust: `Optional[str]`
Chaintrust.

BlockTransactionDetailsModel

```
class BlockTransactionDetailsModel(*, hash: uint256, confirmations: int, size: int, weight: int, height: int,
    version: int, versionHex: str, merkleroot: hexstr, tx: List[uint256] =
    None, time: datetime.datetime, mediantime: datetime.datetime, nonce:
    int, bits: str, difficulty: float, chainwork: str, nTx: int,
    previousblockhash: uint256 = None, nextblockhash: uint256 = None,
    signature: str = None, modifyerv2: str = None, flags: str = None,
    hashproof: str = None, blocktrust: str = None, chaintrust: str = None,
    Transactions: List[TransactionModel])
```

A pydantic model for block transaction details.

transactions: `List[TransactionModel]`
A list of transactions.

hash: `uint256`
The block hash.

confirmations: `int`
The number of confirmations.

size: `int`
The size of the block.

weight: `int`
The weight of the block.

height: `int`
The height of the block.

version: `int`
The block version.

version_hex: `str`
The block version in hex.

merkleroot: `hexstr`
The block merkleroot.

tx: `Optional[List[uint256]]`
A list of transactions in the block.

time: `datetime`
The time the block was produced.

median_time: `datetime`
The median time.

nonce: `int`
The block's nonce.

bits: `str`
The block bits.

difficulty: `float`
The block difficulty.

chainwork: `str`
The chain work.

n_tx: `int`
The number of transactions in the block.

previous_blockhash: `Optional[uint256]`
The previous block hash.

next_blockhash: `Optional[uint256]`
The next block hash.

signature: `Optional[str]`
The signature.

modifier_v2: `Optional[str]`
The block modifier.

flags: `Optional[str]`
Block flags.

hashproof: `Optional[str]`
Block hashproof.

blocktrust: `Optional[str]`
Blocktrust.

chaintrust: `Optional[str]`
Chaintrust.

GetAddressesBalancesModel

class `GetAddressesBalancesModel`(*, *balances*: `List[AddressBalanceModel]`, *reason*: `str = None`)
A pydantic model for retrieving multiple address balances.

balances: `List[AddressBalanceModel]`
A list of addresses with current balances.

reason: `Optional[str]`
If query failed, a reason is given.

GetLastBalanceUpdateTransactionModel

class `GetLastBalanceUpdateTransactionModel`(*, *transaction*: `TransactionModel`, *blockHeight*: `int`)
A pydantic model containing information about there last transaction where a balance changed for an address.

transaction: `TransactionModel`
The transaction model containing the last balance update of the given address.

block_height: `int`
The block height for the last transaction for the given address.

GetUTXOsForAddressModel

class `GetUTXOsForAddressModel`(*, *balance*: `Money`, *utxos*: `List[UTXOModel]`)
A pydantic model representing unspent transaction outputs (utxos) and balance for a given address.

balance: `Money`
The address balance.

utxos: `List[UTXOModel]`
A list of utxos for the given address.

GetVerboseAddressesBalancesModel

```
class GetVerboseAddressesBalancesModel(*, balancesData: List[VerboseAddressBalanceModel],
                                       consensusTipHeight: int, reason: str = None)
```

A pydantic model representing verbose address balance information.

balances_data: List[[VerboseAddressBalanceModel](#)]

A list of verbose address balance data for the given addresses.

consensus_tip_height: int

The current consensus tip height.

reason: Optional[str]

If query failed, a reason is given.

UtxoModel

```
class UTXOModel(*, txId: uint256, index: int, scriptPubKey: str, value: Money)
```

A pydantic model representing a unspent transaction output (utxo).

transaction_id: [uint256](#)

The transaction hash of the transaction containing the utxo.

index: int

The output index of the utxo.

script_pubkey: str

The scriptpubkey of the utxo.

value: [Money](#)

The value of the utxo in coin units.

VerboseAddressBalanceModel

```
class VerboseAddressBalanceModel(*, address: Address, balanceChanges: List[BalanceChangesModel])
```

A pydantic model that verbosely lists balance changes for a given address.

address: [Address](#)

The given address.

balance_changes: List[[BalanceChangesModel](#)]

A list of balance changes for the given address.

1.1.1.4 ColdStaking

ColdStaking

```
class ColdStaking(**kwargs)
```

Implements the coldstaking api endpoints.

info(wallet_name: str, **kwargs) → [InfoModel](#)

Gets general information related to cold staking.

Parameters

- **wallet_name** (str) – The wallet name.

- ****kwargs** – Extra keyword arguments.

Returns The cold staking account information for the given wallet.

Return type `InfoModel`

Raises `APIError` – Error thrown by node API. See message for details.

account(*wallet_name: str, wallet_password: str, is_cold_wallet_account: bool = False, extpubkey: Optional[Union[ExtPubKey, str]] = None, **kwargs*) → `AccountModel`

Create a cold staking account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **wallet_password** (*str*) – The wallet password.
- **is_cold_wallet_account** (*bool, optional*) – If this account is for a cold wallet. Default=False.
- **extpubkey** (`ExtPubKey`, *str, optional*) – The extpubkey for the cold wallet.
- ****kwargs** – Extra keyword arguments.

Returns Information about the cold staking account.

Return type `AccountModel`

Raises `APIError` – Error thrown by node API. See message for details.

address(*wallet_name: str, is_cold_wallet_address: bool = False, segwit: bool = False, **kwargs*) → `AddressModel`

Gets a cold staking address.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **is_cold_wallet_address** (*bool, optional*) – If this address is for a cold wallet. Default=False.
- **segwit** (*bool, optional*) – If this is a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns Information about the cold staking address.

Return type `AddressModel`

Raises `APIError` – Error thrown by node API. See message for details.

setup(*cold_wallet_address: Union[Address, str], hot_wallet_address: Union[Address, str], wallet_name: str, wallet_account: str, wallet_password: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, split_count: int = 1, segwit_change_address: bool = False, **kwargs*) → `SetupModel`

Spends funds from a normal wallet addresses to the cold staking script.

Parameters

- **cold_wallet_address** (`Address`, *str*) – The cold wallet address.
- **hot_wallet_address** (`Address`, *str*) – The hot wallet address.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*) – The wallet account.

- **wallet_password** (*str*) – The wallet password.
- **amount** (*Money, int, float, Decimal*) – The amount to send to the old wallet.
- **fees** (*Money, int, float, Decimal*) – The transaction fee.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- **split_count** (*int, optional*) – Number of transactions to split over. Default=1.
- **segwit_change_address** (*bool, optional*) – If change address is a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns The transaction hex for the cold staking setup transaction.

Return type `SetupModel`

Raises `APIError` – Error thrown by node API. See message for details.

setup_offline(*cold_wallet_address: Union[Address, str], hot_wallet_address: Union[Address, str], wallet_name: str, wallet_account: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, split_count: int = 1, segwit_change_address: bool = False, **kwargs*) → `BuildOfflineSignModel`

Creates a cold staking setup transaction in an unsigned state.

Parameters

- **cold_wallet_address** (*Address, str*) – The cold wallet address.
- **hot_wallet_address** (*Address, str*) – The hot wallet address.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*) – The wallet account.
- **amount** (*Money, int, float, Decimal*) – The amount to send to the old wallet.
- **fees** (*Money, int, float, Decimal*) – The transaction fee.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- **split_count** (*int, optional*) – Number of transactions to split over. Default=1.
- **segwit_change_address** (*bool, optional*) – If change address is a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns The built transaction for signing offline.

Return type `BuildOfflineSignModel`

Raises `APIError` – Error thrown by node API. See message for details.

estimate_setup_tx_fee(*cold_wallet_address: Union[Address, str], hot_wallet_address: Union[Address, str], wallet_name: str, wallet_account: str, wallet_password: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, split_count: int = 1, segwit_change_address: bool = False, **kwargs*) → `Money`

Estimate the cold staking setup tx fee.

Parameters

- **cold_wallet_address** (*Address, str*) – The cold wallet address.
- **hot_wallet_address** (*Address, str*) – The hot wallet address.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*) – The wallet account.
- **wallet_password** (*str*) – The wallet password.
- **amount** (*Money, int, float, Decimal*) – The amount to send to the old wallet.
- **fees** (*Money, int, float, Decimal*) – The transaction fee.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- **split_count** (*int, optional*) – Number of transactions to split over. Default=1.
- **segwit_change_address** (*bool, optional*) – If change address is a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns The cold staking fee estimate.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

estimate_offline_setup_tx_fee(*cold_wallet_address: Union[Address, str], hot_wallet_address: Union[Address, str], wallet_name: str, wallet_account: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, split_count: int = 1, segwit_change_address: bool = False, **kwargs*)
→ *Money*

Estimate the cold staking offline setup tx fee.

Parameters

- **cold_wallet_address** (*Address, str*) – The cold wallet address.
- **hot_wallet_address** (*Address, str*) – The hot wallet address.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*) – The wallet account.
- **amount** (*Money, int, float, Decimal*) – The amount to send to the old wallet.
- **fees** (*Money, int, float, Decimal*) – The transaction fee.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- **split_count** (*int, optional*) – Number of transactions to split over. Default=1.
- **segwit_change_address** (*bool, optional*) – If change address is a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns The offline cold staking fee estimate.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

withdrawal(*receiving_address: Union[Address, str], wallet_name: str, wallet_password: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, **kwargs*) → *WithdrawalModel*

Spends funds from the cold staking wallet account back to a normal wallet account.

Parameters

- **receiving_address** (*Address, str*) – The receiving address.
- **wallet_password** (*str*) – The wallet password.
- **wallet_name** (*str*) – The wallet name.
- **amount** (*Money, int, float, Decimal*) – The amount to withdraw to the receiving address.
- **fees** (*Money, int, float, Decimal, optional*) – The amount paid in fees.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The withdrawal transaction model.

Return type *WithdrawalModel*

Raises *APIError* – Error thrown by node API. See message for details.

offline_withdrawal(*receiving_address: Union[Address, str], wallet_name: str, account_name: str, amount: Union[Money, int, float, decimal.Decimal], fees: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, **kwargs*) → *BuildOfflineSignModel*

Builds a request to spend funds from a cold staking wallet account back to a normal wallet account.

Parameters

- **receiving_address** (*Address, str*) – The receiving address.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str*) – The account name.
- **amount** (*Money, int, float, Decimal*) – The amount to withdraw to the receiving address.
- **fees** (*Money, int, float, Decimal*) – The amount paid in fees.
- **subtract_fee_from_amount** (*bool, optional*) – If fee should be subtracted from amount. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The built withdrawal transaction model for offline signing.

Return type *BuildOfflineSignModel*

Raises *APIError* – Error thrown by node API. See message for details.

estimate_offline_withdrawal_tx_fee(*wallet_name: str, account_name: str, receiving_address: Union[Address, str], amount: Union[Money, int, float, decimal.Decimal], subtract_fee_from_amount: bool = True, **kwargs*) → *Money*

Estimate the fee for an offline cold staking withdrawal transaction.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str*) – The account name.
- **receiving_address** (*Address*, *str*) – The receiving address.
- **amount** (*Money*, *int*, *float*, *Decimal*) – The amount to withdraw to the receiving address.
- **subtract_fee_from_amount** (*bool*, *optional*) – If fee should be subtracted from amount. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The estimate for offline withdrawal transaction fee.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

estimate_withdrawal_tx_fee(*receiving_address: Union[Address, str]*, *wallet_name: str*,
wallet_password: str, *amount: Union[Money, int, float, decimal.Decimal]*,
fees: Union[Money, int, float, decimal.Decimal],
subtract_fee_from_amount: bool = True, ***kwargs*) → *Money*

Estimate the fee for a cold staking withdrawal transaction.

Parameters

- **receiving_address** (*Address*, *str*) – The receiving address.
- **wallet_password** (*str*) – The wallet password.
- **wallet_name** (*str*) – The wallet name.
- **amount** (*Money*, *int*, *float*, *Decimal*) – The amount to withdraw to the receiving address.
- **fees** (*Money*, *int*, *float*, *Decimal*, *optional*) – The amount paid in fees.
- **subtract_fee_from_amount** (*bool*, *optional*) – If fee should be subtracted from amount. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The estimate for the withdrawal transaction fee.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

retrieve_filtered_utxos(*wallet_name: str*, *wallet_password: str*, *wallet_account: str*, *trx_hex: hexstr*,
broadcast: bool = False, ***kwargs*) → *List[hexstr]*

Estimate the fee for a cold staking withdrawal transaction.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **wallet_password** (*str*) – The wallet password.
- **wallet_account** (*str*) – The wallet account.
- **trx_hex** (*hexstr*) – The transaction id hex.
- **broadcast** (*bool*) – If true, broadcast the transaction to the network after being built. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A list of hex encoded coldstaking transactions.

Return type *List*[*hexstr*]

Raises *APIError* – Error thrown by node API. See message for details.

AccountModel

```
class AccountModel(*, accountName: str)
    A pydantic model for a cold staking account.

    account_name: str
        The cold staking account name.
```

AddressModel

```
class AddressModel(*, address: Address)
    A pydantic model for a cold staking address.

    address: Address
        The cold staking address.
```

BuildOfflineSignModel

```
class BuildOfflineSignModel(*, walletName: str, walletAccount: str, unsignedTransaction: hexstr, fee:
    Money, utxos: List[UtxoDescriptor], addresses: List[AddressDescriptor])
    A pydantic model for a built offline sign request.

    wallet_name: str
        The wallet name.

    wallet_account: str
        The wallet account.

    unsigned_transaction: hexstr
        The unsigned transaction hex.

    fee: Money
        The transaction fee.

    utxos: List[UtxoDescriptor]
        The utxos included in the transaction.

    addresses: List[AddressDescriptor]
        The addresses and amounts receiving outputs.
```

InfoModel

class InfoModel(*, *coldWalletAccountExists: bool, hotWalletAccountExists: bool*)

A pydantic model for cold wallet information.

cold_wallet_account_exists: bool

True if cold wallet account exists.

hot_wallet_account_exists: bool

True if hot wallet account exists.

SetupModel

class SetupModel(*, *transactionHex: hexstr*)

A pydantic model for a cold staking wallet setup transaction.

transaction_hex: hexstr

The hex serialized cold staking wallet setup transaction.

WithdrawalModel

class WithdrawalModel(*, *transactionHex: hexstr*)

A pydantic model for a cold staking withdrawal transaction.

transaction_hex: hexstr

A hex serialized cold staking wallet withdrawal transaction.

1.1.1.5 Collateral

Collateral

class Collateral(***kwargs*)

Implements the collateral api endpoints.

join_federation(*collateral_address: Union[Address, str], collateral_wallet_name: str, collateral_wallet_password: str, wallet_password: str, wallet_name: str, wallet_account: str = 'account 0', **kwargs*) → *JoinFederationResponseModel*

Called by a miner wanting to join the federation.

Parameters

- **collateral_address** (*Address, str*) – The collateral address.
- **collateral_wallet_name** (*str*) – The collateral wallet name.
- **collateral_wallet_password** (*SecretStr*) – The collateral wallet password.
- **wallet_password** (*SecretStr*) – The wallet password.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str, optional*) – The wallet account. Default='account 0'.
- ****kwargs** – Extra keyword arguments.

Returns The response to the join-federation API request.

Return type *JoinFederationResponseModel*

Raises ***APIError*** – Error thrown by node API. See message for details.

JoinFederationResponseModel

class `JoinFederationResponseModel`(**MinerPublicKey*: `PubKey`)

A pydantic model for the join federation response.

miner_public_key: `PubKey`

The miner public key for a new federation member, if successful.

1.1.1.6 CollateralVoting

CollateralVoting

class `CollateralVoting`(***kwargs*)

Implements the collateralvoting api endpoints.

schedulevote_kickfedmember(*pubkey_hex*: `Union[str, hexstr, PubKey]`, *collateral_amount_satoshis*: `int`, *collateral_mainchain_address*: `Union[Address, str]`, ***kwargs*) → `None`

Schedule a vote to kick an existing federation member.

Parameters

- **pubkey_hex** (`PubKey`) – The fedmember pubkey hex value.
- **collateral_amount_satoshis** (`int`) – The collateral amount in satoshis.
- **collateral_mainchain_address** (`Address`, `str`) – The mainchain address holding the collateral.
- ****kwargs** – Extra keyword arguments.

Returns `None`

Raises ***APIError*** – Error thrown by node API. See message for details.

1.1.1.7 ConnectionManager

ConnectionManager

class `ConnectionManager`(***kwargs*)

Implements the connectionmanager api endpoints.

addnode(*ipaddr*: `str`, *command*: `str`, ***kwargs*) → `bool`

Interface for the addnode command. Can continuously try to addnode, remove node, or onetry a node connection.

Parameters

- **ipaddr** (`str`) – The endpoint.
- **command** (`str`) – Allowed commands [add, remove, onetry]
- ****kwargs** – Extra keyword arguments.

Returns If command was successful.

Return type `bool`

Raises *APIError* – Error thrown by node API. See message for details.

getpeerinfo(**kwargs) → List[*PeerInfoModel*]

Gets the peer info.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of connected peers.

Return type List[*PeerInfoModel*]

Raises *APIError* – Error thrown by node API. See message for details.

PeerInfoModel

```
class PeerInfoModel(*, id: int, addr: str, addrlocal: str, services: str, relaytxes: bool, lastsend: int, lastrecv: int, bytessent: int, bytesrecv: int, conntime: int, timeoffset: int, pingtime: int, minping: int, pingwait: int, version: int, subver: str, inbound: bool, addnode: bool, startingheight: int, banscore: int, synced_headers: int, synced_blocks: int, whitelisted: bool, inflight: bool = None, bytessent_per_msg: int = None, bytesrecv_per_msg: int = None)
```

A pydantic model with power information..

peer_id: int

The peer id.

addr: str

The peer address.

addrlocal: str

The local peer address.

services: str

Peer services.

relaytxes: bool

Relay transactions.

lastsend: int

Last send.

lastrecv: int

Last received.

bytessent: int

Bytes sent.

bytesrecv: int

Bytes received.

conntime: int

Connection time in seconds.

timeoffset: int

The peer time offset in seconds.

pingtime: int

The ping time in ms.

minping: int

The minimum ping time.

pingwait: int
The point wait time.

version: int
The peer version

subver: str
The peer subversion.

inbound: bool
Inbound connected peer.

addnode: bool
If true, peer was connected by addnode.

starting_height: int
Connection starting height.

banscore: int
The peer's ban score.

synced_headers: int
The number of synced headers.

synced_blocks: int
The number of synced blocks.

whitelisted: bool
If true, peer is whitelisted.

inflight: Optional[bool]
Inflight peer.

bytessent_per_msg: Optional[int]
Bytes sent per message.

bytesrecv_per_msg: Optional[int]
Bytes received per message.

1.1.1.8 Consensus

Consensus

class Consensus(**kwargs)

Implements the consensus api endpoints.

deployment_flags(**kwargs) → List[*DeploymentFlagsModel*]

Get the active deployment flags.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of active deployment flags..

Return type List[*DeploymentFlagsModel*]

Raises *APIError* – Error thrown by node API. See message for details.

get_best_blockhash(**kwargs) → *uint256*

Gets the best block hash.

Parameters ****kwargs** – Extra keyword arguments.

Returns The block hash.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

get_blockhash(*height: int, **kwargs*) → `uint256`

Gets the block hash at the specified block

Parameters

- **height** (`int`) – The requested height for block hash retrieval.
- ****kwargs** – Extra keyword arguments.

Returns The block hash.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

DeploymentFlagsModel

```
class DeploymentFlagsModel(*, deploymentName: str, deploymentIndex: int, stateValue: int, thresholdState: str, height: int, sinceHeight: int, confirmationPeriod: int, periodStartHeight: int, periodEndHeight: int, votes: int, blocks: int, versions: dict, threshold: int, timeStart: str, timeTimeout: str)
```

A pydantic model representing deployment flags.

deployment_name: str

The deployment flag name.

deployment_index: int

The deployment index.

state_value: int

The deployment flag state value.

threshold_state: str

The deployment flag threshold state.

height: int

The block height.

since_height: int

The flag's activation height.

confirmation_period: int

The confirmation period size.

period_start_height: int

The activation period start height.

period_end_height: int

The activation period ending height.

votes: int

The number of votes.

blocks: int

Blocks.

versions: dict

Versions.

threshold: int
The threshold.

time_start: str
The start time.

time_time_out: str
The timeout time.

1.1.1.9 Dashboard

Dashboard

class Dashboard(kwargs)**
Implements the dashboard api endpoints.

stats(kwargs) → str**
Gets the dashboard of node stats.

Parameters ****kwargs** – Extra keyword arguments.

Returns The dashboard text.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

asyncloops_stats(kwargs) → str**
Gets the dashboard of asyncloop stats.

Parameters ****kwargs** – Extra keyword arguments.

Returns The async loop stats.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

1.1.1.10 Diagnostic

Diagnostic

class Diagnostic(kwargs)**
Implements the diagnostic api endpoints.

get_connectedpeers_info(kwargs) → *GetConnectedPeersInfoModel***
Get connected peers info.

Parameters ****kwargs** – Extra keyword arguments.

Returns Information on connected peers.

Return type *GetConnectedPeersInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

get_status(kwargs) → *GetStatusModel***
Get the diagnostic feature status.

Parameters ****kwargs** – Extra keyword arguments.

Returns The feature status.

Return type `GetStatusModel`

Raises `APIError` – Error thrown by node API. See message for details.

get_peer_statistics(*connected_only: bool, **kwargs*) → `List[PeerStatisticsModel]`
Gets statistics for connected peers.

Parameters

- **connected_only** (*bool*) – To show data for only connected nodes.
- ****kwargs** – Extra keyword arguments.

Returns A list of statistics on the connected peers.

Return type `List[PeerStatisticsModel]`

Raises `APIError` – Error thrown by node API. See message for details.

start_collecting_peerstatistics(***kwargs*) → `str`
Start collecting peer statistics.

Parameters ****kwargs** – Extra keyword arguments.

Returns The status of the feature.

Return type `str`

Raises `APIError` – Error thrown by node API. See message for details.

stop_collecting_peerstatistics(***kwargs*) → `str`
Stop collecting peer statistics.

Parameters ****kwargs** – Extra keyword arguments.

Returns The status of the feature.

Return type `str`

Raises `APIError` – Error thrown by node API. See message for details.

ConnectedPeerInfoModel

class ConnectedPeerInfoModel(**, isConnected: bool, disconnectReason: str = None, state: int, endPoint: str*)
A pydantic model representing connected peer information.

is_connected: bool
True if the peer is connected

disconnect_reason: Optional[str]
The reason for disconnection.

state: int
The connection state.

endpoint: str
The peer's endpoint.

GetConnectedPeersInfoModel

```
class GetConnectedPeersInfoModel(*, peersByPeerId: List[ConnectedPeerInfoModel], connectedPeers:
    List[ConnectedPeerInfoModel], connectedPeersNotInPeersByPeerId:
    List[ConnectedPeerInfoModel])
```

A pydantic model for connected peer information.

peers_by_peer_id: List[ConnectedPeerInfoModel]

A list of peers by id.

connected_peers: List[ConnectedPeerInfoModel]

A list of connected peers.

connected_peers_not_in_peers_by_peer_id: List[ConnectedPeerInfoModel]

A list of known peer IDs not connected to.

GetStatusModel

```
class GetStatusModel(*, peerStatistics: str)
```

A pydantic model for status of diagnostics collection service.

peer_statistics: str

The status of the service.

PeerStatisticsModel

```
class PeerStatisticsModel(*, peerEndPoint: str, connected: bool, inbound: bool, bytesSent: int,
    bytesReceived: int, receivedMessages: int, sentMessages: int, latestEvents:
    List[str])
```

A pydantic model for peer statistics.

peer_endpoint: str

The peer endpoint.

connected: bool

If true, peer is connected.

inbound: bool

If true, peer is inbound connection.

bytes_sent: int

Bytes sent to peer.

bytes_received: int

Bytes received from peer.

received_messages: int

The number of received messages from peer.

send_messages: int

The number of sent messages to peer.

latest_events: List[str]

A list of peer events.

1.1.1.11 Federation

Federation

class Federation(**kwargs)

Implements the federation api endpoints.

reconstruct(**kwargs) → str

Signals the node to rebuild the federation. Will need to restart node when complete.

Parameters **kwargs – Extra keyword arguments.

Returns The node response of the request.

Return type str

Raises *APIError* – Error thrown by node API. See message for details.

members_current(**kwargs) → *FederationMemberDetailedModel*

Retrieves the information for the current federation member's voting status and mining estimates.

Parameters **kwargs – Extra keyword arguments.

Returns Information on the current member.

Return type *FederationMemberDetailedModel*

Raises *APIError* – Error thrown by node API. See message for details.

members(**kwargs) → List[*FederationMemberModel*]

Retrieves a list of active federation members and last active times.

Parameters **kwargs – Extra keyword arguments.

Returns Information on each of federation members.

Return type List[*FederationMemberModel*]

Raises *APIError* – Error thrown by node API. See message for details.

miner_at_height(block_height: int, **kwargs) → *PubKey*

Gets the federation pubkey that mined the block at the specified height.

Parameters

- **block_height** – The height to query
- **kwargs –

Returns The pubkey that produced the block at the specified height.

Return type *PubKey*

Raises *APIError* – Error thrown by node API. See message for details.

federation_at_height(block_height: int, **kwargs) → List[*PubKey*]

Gets the federation membership at the specified height.

Parameters

- **block_height** – The height to query
- **kwargs –

Returns The pubkeys of federation members at the specified height.

Return type List[*PubKey*]

Raises [APIError](#) – Error thrown by node API. See message for details.

FederationMemberDetailedModel

```
class FederationMemberDetailedModel(*, pubkey: PubKey, collateralAmount: Money, lastActiveTime: str,
    periodOfInactivity: str, pollStartBlockHeight: int = None,
    pollNumberOfVotesAcquired: int = None, pollFinishedBlockHeight:
    int = None, pollWillFinishInBlocks: int = None,
    pollExecutedBlockHeight: int = None,
    memberWillStartMiningAtBlockHeight: int = None,
    memberWillStartEarningRewardsEstimateHeight: int = None,
    pollType: str = None, rewardEstimatePerBlock: Money = None,
    federationSize: int, miningStats:
    pystratis.api.federation.responsemodels.miningstats.MiningStats)
```

A pydantic model for details on federation members.

poll_start_block_height: Optional[int]

The poll start block height.

poll_number_of_votes_acquired: Optional[int]

The number of poll votes acquired.

poll_finished_block_height: Optional[int]

The poll finished block height.

poll_will_finish_in_blocks: Optional[int]

The number of blocks until the poll finishes.

poll_executed_block_height: Optional[int]

The block height where poll was executed, if it has occurred.

member_will_start_mining_at_block_height: Optional[int]

Height when the member will start mining.

pubkey: PubKey

The federation member's pubkey.

collateral_amount: Money

The federation member's collateral amount.

last_active_time: str

The federation member's last active time.

period_of_inactivity: str

The federation member's period of inactivity.

member_will_start_earning_rewards_estimate_height: Optional[int]

Height when member will start earning rewards.

poll_type: Optional[str]

The poll type.

reward_estimate_per_block: Optional[Money]

The reward estimate per block.

federation_size: int

The size of the federation.

mining_stats: pystratis.api.federation.responsemodels.miningstats.MiningStats

The mining stats for the federation member.

FederationMemberModel

```
class FederationMemberModel(*, pubkey: PubKey, collateralAmount: Money, lastActiveTime: str,
                             periodOfInactivity: str)
```

A pydantic model for a federation member.

pubkey: *PubKey*

The federation member's pubkey.

collateral_amount: *Money*

The federation member's collateral amount.

last_active_time: *str*

The federation member's last active time.

period_of_inactivity: *str*

The federation member's period of inactivity.

1.1.1.12 FederationGateway

FederationGateway

```
class FederationGateway(**kwargs)
```

Implements the federationgateway api endpoints.

deposits(*block_height: int, **kwargs*) → List[*MaturedBlockDepositsModel*]

Retrieves block deposits

Parameters

- **block_height** (*int*) – The block height at which to obtain deposits.
- ****kwargs** – Extra keyword arguments.

Returns A list of matured block deposits.

Return type List[*MaturedBlockDepositsModel*]

Raises *APIError* – Error thrown by node API. See message for details.

pending_transfer(*deposit_id: Union[str, uint256], transaction_id: Union[str, uint256], **kwargs*) → List[*CrossChainTransferModel*]

Gets pending transfers.

Parameters

- **deposit_id** (*uint256, str*) – The deposit id hash.
- **transaction_id** (*uint256, str*) – The transaction id hash.
- ****kwargs** – Extra keyword arguments.

Returns A list of cross chain transfers.

Return type List[*CrossChainTransferModel*]

Raises *APIError* – Error thrown by node API. See message for details.

fullysigned_transfer(*deposit_id: Union[str, uint256], transaction_id: Union[str, uint256], **kwargs*) → List[*CrossChainTransferModel*]

Get fully signed transfers.

Parameters

- **deposit_id** (*uint256*, *str*) – The deposit id hash.
- **transaction_id** (*uint256*, *str*) – The transaction id hash.
- ****kwargs** – Extra keyword arguments.

Returns A list of cross chain transfers.

Return type *List*[*CrossChainTransferModel*]

Raises *APIError* – Error thrown by node API. See message for details.

member_info(***kwargs*) → *FederationMemberInfoModel*

Gets info on the state of a multisig member.

Parameters ****kwargs** – Extra keyword arguments.

Returns Information on the current multisig member.

Return type *FederationMemberInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

info(***kwargs*) → *FederationGatewayInfoModel*

Gets info on the state of the federation.

Parameters ****kwargs** – Extra keyword arguments.

Returns Information on the federation gateway.

Return type *FederationGatewayInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

ip_add(*ipaddr: str*, ***kwargs*) → *str*

Add a federation member's IP address to the federation IP list

Parameters

- **ipaddr** (*str*) – The endpoint.
- ****kwargs** – Extra keyword arguments.

Returns Response to ip add request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

ip_remove(*ipaddr: str*, ***kwargs*) → *str*

Remove a federation member's IP address to the federation IP list

Parameters

- **ipaddr** (*str*) – The endpoint.
- ****kwargs** – Extra keyword arguments.

Returns response to ip remove request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

ip_replace(*ipaddr: str*, *ipaddr: str*, ***kwargs*) → *str*

Replace a federation member's IP from the federation IP list with another.

Parameters

- **ipaddr** (*str*) – The new endpoint.

- **ipaddr** (*str*) – The endpoint being replaced.
- ****kwargs** – Extra keyword arguments.

Returns Response to ip replace request.

Return type *str*

Raises **APIError** – Error thrown by node API. See message for details.

verify_transfer(*deposit_id_transaction_id: Union[str, uint256], **kwargs*) → Union[str, *ValidateTransactionResultModel*]

Validate a transfer transaction.

Parameters

- **deposit_id_transaction_id** (*uint256, str*) – The transaction id containing the deposit with the deposit id.
- ****kwargs** – Extra keyword arguments.

Returns A model describing the validity of the transfer.

Return type *Union[str, ValidateTransactionResultModel]*

Raises **APIError** – Error thrown by node API. See message for details.

transfers_delete_suspended(***kwargs*) → *str*

Delete a suspended transfer transaction.

Parameters ****kwargs** – Extra keyword arguments.

Returns A message about the deletion request.

Return type *str*

Raises **APIError** – Error thrown by node API. See message for details.

transfer(*deposit_id: Union[str, uint256], **kwargs*) → *CrossChainTransferModel*

Gets pending transfers.

Parameters

- **deposit_id** (*uint256, str*) – The deposit id hash.
- ****kwargs** – Extra keyword arguments.

Returns A cross chain transfer.

Return type *CrossChainTransferModel*

Raises **APIError** – Error thrown by node API. See message for details.

CrosschainTransferModel

```
class CrossChainTransferModel(*, depositAmount: Money, depositId: uint256, depositHeight: int,
                              transferStatus: CrossChainTransferStatus, tx: TransactionModel)
```

A pydantic model of a cross chain transfer.

deposit_amount: *Money*

The amount deposited.

deposit_id: *uint256*

The hash of the deposit transaction.

deposit_height: int
The height of the deposit transaction.

transfer_status: *CrossChainTransferStatus*
The transfer status.

tx: *TransactionModel*
The transaction model of the cross chain transaction.

FederationGatewayInfoModel

```
class FederationGatewayInfoModel(*, active: bool, mainchain: bool, endpoints: List[str], multisigPubKey:
    PubKey, federationMultisigPubKeys: List[PubKey], miningPubKey:
    PubKey = "", federationMiningPubKeys: List[PubKey] = [],
    multisigAddress: Address, multisigRedeemScript: str = None,
    multisigRedeemScriptPaymentScript: str = None, minconfsmalldeposits:
    int, minconfnormaldeposits: int, minconflargedeposits: int,
    minconfdistributiondeposits: int)
```

A pydantic model representing information on the federation gateway.

active: bool
True if federation gateway is active.

mainchain: bool
True if called on mainchain node.

multisig_pubkey: *PubKey*
The multisig pubkey.

federation_multisig_pubkeys: List[*PubKey*]
A list of federation multisig pubkeys.

mining_pubkey: *PubKey*
The mining pubkey.

federation_mining_pubkeys: Optional[List[*PubKey*]]
The federation mining pubkeys. Only active if federation mining is active.

multisig_address: *Address*
The multisig address.

multisig_redeem_script: Optional[str]
The multisig redeem script.

multisig_redeem_script_payment_script: Optional[str]
The multisig redeem script payment script.

min_conf_small_deposits: int
The minimum confirmations for small deposits.

min_conf_normal_deposits: int
The minimum confirmations for normal deposits.

min_conf_large_deposits: int
The minimum confirmations for large deposits.

min_conf_distribution_deposits: int
The minimum confirmations for distribution deposits.

FederationMemberConnectionInfoModel

```
class FederationMemberConnectionInfoModel(*, federationMemberIp: str, isConnected: bool, isBanned: bool)
```

A pydantic model for federation member connections.

federation_member_ip: str

The federation member ip.

is_connected: bool

If true, federation member is connected.

is_banned: bool

If true, federation member is banned.

FederationMemberInfoModel

```
class FederationMemberInfoModel(*, asyncLoopState: str, consensusHeight: int, cctsHeight: int, cctsNextDepositHeight: int, cctsPartials: int, cctsSuspended: int, federationWalletActive: bool, federationWalletHeight: int, nodeVersion: str, pubKey: PubKey = None, federationConnectionState: str, federationMemberConnections: List[FederationMemberConnectionInfoModel])
```

A pydantic model representing information about the current federation member.

async_loop_state: str

The async loop state.

consensus_height: int

The current consensus height.

ccts_height: int

The node's CCTS height.

ccts_next_deposit_height: int

The next CCTS deposit height.

ccts_partials: int

The number of partial CCTS transactions.

ccts_suspended: int

The number of suspended CCTS transactions.

federation_wallet_active: bool

If true, the federation wallet is active.

federation_wallet_height: int

The local federation wallet height.

node_version: str

The node version.

pubkey: Optional[PubKey]

The member's pubkey. Could be None.

federation_connection_state: str

The federation connection state.

federation_member_connections: List[FederationMemberConnectionInfoModel]

A list of connected federation members.

MaturedBlockDepositsModel

class MaturedBlockDepositsModel(* , deposits: List[Deposit], blockInfo: MaturedBlockInfoModel)

A pydantic model for matured block deposits.

deposits: List[Deposit]

A list of deposits.

block_info: MaturedBlockInfoModel

Matured block information model.

SerializableResult

class SerializableResult(* , value: Any = None, message: str = "")

A pydantic model for a serializable result.

value: Optional[Any]

message: Optional[str]

ValidateTransactionResultModel

class ValidateTransactionResultModel(* , isValid: bool, errors: List[str] = None)

A pydantic model for a validate transaction result.

is_valid: bool

If true, transaction is valid.

errors: Optional[List[str]]

Transaction validation errors.

1.1.1.13 FederationWallet

FederationWallet

class FederationWallet(**kwargs)

Implements the federationwallet api endpoints.

general_info(**kwargs) → WalletGeneralInfoModel

Retrieves general information about the wallet.

Parameters ****kwargs** – Extra keyword arguments.

Returns General information about the federation wallet.

Return type WalletGeneralInfoModel

Raises **APIError** – Error thrown by node API. See message for details.

balance(**kwargs) → WalletBalanceModel

Retrieves wallet balances.

Parameters ****kwargs** – Extra keyword arguments.

Returns Federation wallet balance information.

Return type WalletBalanceModel

Raises **APIError** – Error thrown by node API. See message for details.

history(*max_entries_to_return: int, **kwargs*) → List[WithdrawalModel]

Retrieves a withdrawal history for the wallet.

Parameters

- **max_entries_to_return** (*int*) – The maximum number of history entries to return.
- ****kwargs** – Extra keyword arguments.

Returns The federation wallet history.

Return type List[WithdrawalModel]

Raises **APIError** – Error thrown by node API. See message for details.

sync(*block_hash: Union[str, uint256], **kwargs*) → None

Starts sending block to wallet for synchronisation. Demo/testing use only.

Parameters

- **block_hash** (*uint256, str*) – The block hash at which to start sync.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises **APIError** – Error thrown by node API. See message for details.

enable_federation(*mnemonic: str, password: str, passphrase: Optional[str] = None, timeout_seconds: int = 60, **kwargs*) → Union[None, str]

Enables the federation so that it can sign transactions.

Parameters

- **mnemonic** (*str*) – The mnemonic.
- **password** (*str*) – The password.
- **passphrase** (*str, optional*) – The passphrase.
- **timeout_seconds** (*int, optional*) – Seconds to timeout. Default=60.
- ****kwargs** – Extra keyword arguments.

Returns None if successful, str if error.

Return type (*str, None*)

Raises **APIError** –

remove_transactions(*resync: bool = True, **kwargs*) → List[RemovedTransactionModel]

Remove all transactions from the wallet.

Parameters

- **resync** (*bool, optional*) – A flag to resync the wallet after transactions are removed. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns A list of removed transactions from the federation wallet.

Return type List[RemovedTransactionModel]

Raises **APIError** – Error thrown by node API. See message for details.

RemovedTransactionModel

```
class RemovedTransactionModel(*, transactionId: uint256, creationTime: datetime.datetime)
```

A pydantic model for a removed transaction.

```
transaction_id: uint256
```

The removed transaction hash.

```
creation_time: datetime.datetime
```

The creation time of the removed transaction.

WalletBalanceModel

```
class WalletBalanceModel(*, balances: List[AccountBalanceModel])
```

A pydantic model for a wallet balance.

```
balances: List[AccountBalanceModel]
```

A list of account balances.

WalletGeneralInfoModel

```
class WalletGeneralInfoModel(*, walletName: str = None, network: str, creationTime: datetime.datetime,
                             isDecrypted: bool, lastBlockSyncedHeight: int, chainTip: int, isChainSynced:
                             bool, connectedNodes: int)
```

A model representing general wallet info.

```
wallet_name: Optional[str]
```

The name of the wallet. Will be None for multisig.

```
network: str
```

The name of the network the wallet is operating on.

```
creation_time: datetime.datetime
```

The datetime of wallet creation

```
is_decrypted: bool
```

If true, wallet is decrypted.

```
last_block_synced_height: int
```

The height of last block synced by wallet.

```
chain_tip: int
```

The height off the chain tip.

```
is_chain_synced: bool
```

If true, chain is synced.

```
connected_nodes: int
```

The number of connected nodes.

WithdrawalModel

```
class WithdrawalModel(*, id: uint256, depositId: uint256, amount: Money, payingTo: Union[str, Address],
                      blockHeight: int, blockHash: uint256, signatureCount: int, spendingOutputDetails: str,
                      transferStatus: CrossChainTransferStatus)
```

A pydantic model for a withdrawal processed by the multisig federation.

trx_id: *uint256*

The transaction hash.

deposit_id: *uint256*

The deposit transaction hash.

amount: *Money*

The amount of the withdrawal.

paying_to: *Union[str, Address]*

The address receiving the withdrawal.

block_height: *int*

The block height of the withdrawal.

block_hash: *uint256*

The hash of the block containing the withdrawal.

signature_count: *int*

The number of signatures on the withdrawal.

spending_output_details: *str*

Spending output details.

transfer_status: *CrossChainTransferStatus*

The cross chain transfer status.

1.1.1.14 Interop

Interop

```
class Interop(**kwargs)
```

Implements the interop api endpoints.

status_burns(**kwargs) → List[*ConversionRequestModel*]

Gets the current interop status of burns.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of burn conversion requests.

Return type List[*ConversionRequestModel*]

Raises *APIError* – Error thrown by node API. See message for details.

status_mints(**kwargs) → List[*ConversionRequestModel*]

Gets the current interop status of mints.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of mint conversion requests.

Return type List[*ConversionRequestModel*]

Raises *APIError* – Error thrown by node API. See message for details.

status_votes(***kwargs*) → dict

Gets the current interop status of votes.

Parameters ***kwargs* – Extra keyword arguments.

Returns A dictionary of votes with {request_id: [pubkeys_that_voted]}

Return type *dict*

Raises *APIError* – Error thrown by node API. See message for details.

owners(*destination_chain: int, **kwargs*) → List[str]

Retrieves the list of current owners for the multisig wallet contract.

Parameters

- **destination_chain** (*int*) – The destination chain.
- ***kwargs* – Extra keyword arguments.

Returns A list of owners of the multisig contract.

Return type *List[str]*

Raises *APIError* – Error thrown by node API. See message for details.

add_owner(*destination_chain: int, new_owner_address: Union[str, Address], gas_price: int, **kwargs*) → *uint256*

Creates and broadcasts an addOwner contract call on the multisig wallet contract. This can only be done by one of the current owners of the contract, and needs to be confirmed by a sufficient number of the other owners.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **new_owner_address** (*str, Address*) – The address to add to multisig ownership.
- **gas_price** (*int*) – The gas price.
- ***kwargs* – Extra keyword arguments.

Returns The transactionId of the multisig wallet contract transaction, which is then used to confirm the transaction.

Return type *uint256*

Raises *APIError* – Error thrown by node API. See message for details.

remove_owner(*destination_chain: int, existing_owner_address: Union[str, Address], gas_price: int, **kwargs*) → *uint256*

Creates and broadcasts a removeOwner contract call on the multisig wallet contract. This can only be done by one of the current owners of the contract, and needs to be confirmed by a sufficient number of the other owners.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **existing_owner_address** (*str, Address*) – The address to add to multisig ownership.
- **gas_price** (*int*) – The gas price.
- ***kwargs* – Extra keyword arguments.

Returns The transactionId of the multisig wallet contract transaction, which is then used to confirm the transaction.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

confirm_transaction(*destination_chain: int, transaction_id: int, gas_price: int, **kwargs*) → `uint256`

Explicitly confirms a given multisig wallet contract transactionId by submitting a contract call transaction to the network.

This can only be called once per multisig owner. Additional calls by the same owner account will simply fail and waste gas.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **transaction_id** (*int*) – The multisig wallet transactionId (int, not transaction hash).
- **gas_price** (*int*) – The gas price.
- ****kwargs** – Extra keyword arguments.

Returns The on-chain transaction hash of the contract call transaction.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

change_requirement(*destination_chain: int, requirement: int, gas_price: int, **kwargs*) → `uint256`

Creates and broadcasts a ‘changeRequirement()’ contract call on the multisig wallet contract. This can only be done by one of the current owners of the contract, and needs to be confirmed by a sufficient number of the other owners.

This should only be done once all owner modifications are complete to save gas and orchestrating confirmations.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **requirement** (*int*) – The new threshold for confirmations on the multisig wallet contract. Can usually be numOwners / 2 rounded up.
- **gas_price** (*int*) – The gas price.
- ****kwargs** – Extra keyword arguments.

Returns The multisig wallet transactionId of the changerequirement call.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

multisig_transaction(*destination_chain: int, transaction_id: int, raw: bool, **kwargs*) → Union[`hexstr`, `TransactionResponseModel`]

Retrieves a multisig wallet transaction.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **transaction_id** (*int*) – The multisig wallet transactionId (int, not transaction hash).
- **raw** (*bool*) – Indicates whether to partially decode the transaction or leave it in raw hex format.
- ****kwargs** – Extra keyword arguments.

Returns The multisig wallet transaction data.

Return type *Union[hexstr, TransactionResponseModel]*

Raises *APIError* – Error thrown by node API. See message for details.

multisig_confirmations(*destination_chain: int, transaction_id: int, **kwargs*) → List[str]

Returns the list of contract owners that confirmed a particular multisig transaction.

Parameters

- **destination_chain** (*int*) – The destination chain.
- **transaction_id** (*int*) – The multisig wallet transactionId (int, not transaction hash).
- ****kwargs** – Extra keyword arguments.

Returns A list of owner addresses that confirmed the transaction.

Return type *List[str]*

Raises *APIError* – Error thrown by node API. See message for details.

balance(*destination_chain: int, account: str, **kwargs*) → *Money*

Retrieves the wSTRAX balance of a given account.

Parameters

- **destination_chain** (*int*) – The chain the wSTRAX ERC20 contract is deployed to.
- **account** (*str*) – The account to retrieve the balance for.
- ****kwargs** – Extra keyword arguments.

Returns The wSTRAX account balance.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

requests_delete(***kwargs*) → *Money*

Deletes conversion requests.

Parameters ****kwargs** – Extra keyword arguments.

Returns A message about the status of the request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

requests_setoriginator(*request_id: int, **kwargs*) → *str*

Endpoint that allows the multisig operator to set itself as the originator (submitter) for a given request id.

Parameters

- **request_id** (*int*) – The requestId in question.
- ****kwargs** – Extra keyword arguments.

Returns A message about the status of the request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

requests_setnotoriginator(*request_id: int, **kwargs*) → *str*

Endpoint that allows the multisig operator to reset the request as NotOriginator.

Parameters

- **request_id** (*int*) – The requestId in question.

- ****kwargs** – Extra keyword arguments.

Returns A message about the status of the request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

requests_reprocess_burn(*request_id: int, height: int, **kwargs*) → *str*
Endpoint that allows the multisig operator to reprocess a burn.

Parameters

- **request_id** (*int*) – The requestId to reprocess burn.
- **height** (*int*) – The height at which to reprocess the burn.
- ****kwargs** – Extra keyword arguments.

Returns A message about the status of the request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

requests_pushvote(*request_id: int, vote_id: int, **kwargs*) → *str*
Endpoint that allows the multisig operator to manually add a vote if they are originator of the request.

Parameters

- **request_id** (*int*) – The request id.
- **vote_id** (*int*) – The vote id.
- ****kwargs** – Extra keyword arguments.

Returns A message about the status of the request.

Return type *str*

Raises *APIError* – Error thrown by node API. See message for details.

ConversionRequestModel

```
class ConversionRequestModel(*, requestId: uint256, requestType: ConversionRequestType, requestStatus:
    int, blockHeight: int, destinationAddress: Address, destinationChain:
    DestinationChain, amount: Money, processed: bool)
```

A pydantic model of a conversion request.

request_id: *uint256*

The hash of the conversion request.

request_type: *ConversionRequestType*

The conversion request type.

request_status: *int*

The conversion request status.

block_height: *int*

The block height of the transaction.

destination_address: *Address*

The destination address.

destination_chain: *DestinationChain*

The destination chain.

amount: *Money*

The amount converted.

processed: **bool**

True if the conversion has been processed.

TransactionResponseModel

class TransactionResponseModel(**data*: *hexstr*, *destination*: *DestinationChain*, *value*: *Money*, *executed*: *bool*)

A pydantic model of a multisig transaction response.

data: *hexstr*

The transaction hexstr.

destination: *DestinationChain*

The destination chain.

value: *Money*

The amount converted.

executed: **bool**

True if the transaction has been processed.

1.1.1.15 Mempool

Mempool

class Mempool(**kwargs*)

Implements the mempool api endpoints.

get_raw_mempool(**kwargs*) → *List[uint256]*

Gets a list of transaction hashes in the mempool.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of transactions in the mempool.

Return type *List[uint256]*

Raises *APIError* – Error thrown by node API. See message for details.

1.1.1.16 Mining

Mining

class Mining(**kwargs*)

Implements the mining api endpoints.

generate(*block_count*: *int*, ***kwargs*) → *GenerateBlocksModel*

Generate blocks by mining.

Parameters

- **block_count** (*int*) – The number of blocks to mine.

- ****kwargs** – Extra keyword arguments.

Returns A list of generated blocks.

Return type `GenerateBlocksModel`

Raises `APIError` – Error thrown by node API. See message for details.

stop_mining(**kwargs) → None
Stop mining.

Parameters ****kwargs** – Extra keyword arguments.

Returns None

Raises `APIError` – Error thrown by node API. See message for details.

GenerateBlocksModel

class `GenerateBlocksModel`(*, blocks: List[uint256])

A pydantic model for generated blocks.

blocks: List[uint256]

A list of hashes of generated blocks.

1.1.1.17 Multisig

Multisig

class `Multisig`(**kwargs)

Implements the multisig api endpoints.

build_transaction(recipients: List[Recipient], secrets: List[MultisigSecret], **kwargs) →
`BuildTransactionModel`

Builds a transaction.

Parameters

- **recipients** (List[Recipient]) – A list of recipient objects.
- **secrets** (List[MultisigSecret]) – A list of corresponding multisig secrets.
- ****kwargs** – Extra keyword arguments.

Returns A built multisig transaction.

Return type `BuildTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

BuildTransactionModel

class BuildTransactionModel(**fee*: Money = 0, *hex*: hexstr, *transactionId*: uint256)

A pydantic model for a built transaction.

fee: Money

The transaction fee.

hex: hexstr

The transaction hex.

transaction_id: uint256

The transaction hash.

1.1.1.18 Network

Network

class Network(***kwargs*)

Implements the network api endpoints.

disconnect(*peer_address*: str, ***kwargs*) → None

Disconnect from a node.

Parameters

- **peer_address** (*str*) – The peer endpoint.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises **APIError** – Error thrown by node API. See message for details.

set_ban(*ban_command*: str, *ban_duration_seconds*: int, *peer_address*: str, ***kwargs*) → None

Set a banned node.

Parameters

- **ban_command** (*str*) – Allowed commands [add, remove].
- **ban_duration_seconds** (*int*) – The ban duration in seconds.
- **peer_address** (*str*) – The peer address to ban/unban.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises **APIError** – Error thrown by node API. See message for details.

get_bans(***kwargs*) → List[BannedPeerModel]

Get a list of banned peers.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of banned peers with information on duration and reason for ban.

Return type List[BannedPeerModel]

Raises **APIError** – Error thrown by node API. See message for details.

clear_banned(***kwargs*) → None

Clear banned node list.

Parameters ****kwargs** – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

BannedPeerModel

class BannedPeerModel(*, *endPoint: str, banUntil: str, banReason: str*)

A pydantic model describing a banned peer.

endpoint: str

The banned peer endpoint.

ban_until: str

The time when ban end.

ban_reason: str

The reason for the ban.

1.1.1.19 Node

Node

class Node(***kwargs*)

Implements the node api endpoints.

status(*publish: bool = False, **kwargs*) → *StatusModel*

Gets the node status.

Parameters

- **publish** (*bool*) – If true, publish a full node event with the status.
- ****kwargs** – Extra keyword arguments.

Returns Information about the node status.

Return type *StatusModel*

Raises *APIError* – Error thrown by node API. See message for details.

get_blockheader(*block_hash: Union[str, uint256], is_json_format: bool = True, **kwargs*) → *BlockHeaderModel*

Gets the specified block header.

Parameters

- **block_hash** (*str, uint256*) – The specified block hash.
- **is_json_format** (*bool, optional*) – If block header should be returned as json. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The block headers.

Return type *BlockHeaderModel*

Raises *APIError* – Error thrown by node API. See message for details.

get_raw_transaction(*txid: Union[uint256, str]*, *verbose: bool = False*, ***kwargs*) → Union[*hexstr*, *TransactionModel*]

Gets a raw transaction from a transaction id.

Requires txindex=1 in node configuration.

Parameters

- **txid** (*uint256*, *str*) – The transaction hash.
- **verbose** (*bool*, *optional*) – If output should include verbose transaction data. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A raw transaction.

Return type Union[*hexstr*, *TransactionModel*]

Raises *APIError* – Error thrown by node API. See message for details.

decode_raw_transaction(*raw_hex: Union[str, hexstr]*, ***kwargs*) → *TransactionModel*

Decodes raw transaction hex into a transaction model.

Parameters

- **raw_hex** (*hexstr*, *str*) – The transaction hexstring.
- ****kwargs** – Extra keyword arguments.

Returns A transaction model.

Return type *TransactionModel*

Raises *APIError* – Error thrown by node API. See message for details.

validate_address(*address: str*, ***kwargs*) → *ValidateAddressModel*

Validate an address

Parameters

- **address** (*str*) – The address to validate.
- ****kwargs** – Extra keyword arguments.

Returns Information on the validity of the provided address, and if valid, if it is a witness or script address.

Return type *ValidateAddressModel*

Raises *APIError* – Error thrown by node API. See message for details.

get_txout(*txid: Union[uint256, str]*, *vout: int*, *include_mempool: bool = True*, ***kwargs*) → *GetTxOutModel*

Gets a specified txout from a given transaction.

Parameters

- **txid** (*uint256*, *str*) – The txid to check.
- **vout** (*int*) – The vout.
- **include_mempool** (*bool*, *optional*) – Include mempool in check. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns The specified txout.

Return type *GetTxOutModel*

Raises **APIError** – Error thrown by node API. See message for details.

get_txout_proof(*txids: List[Union[str, uint256]]*, *block_hash: Union[uint256, str]*, ***kwargs*) → *hexstr*
The merkle proof that the specified transaction exist in a given block.

Should have txindex enabled if not specifying blockhash.

Parameters

- **txids** (*List[uint256]*) – A list of transaction hashes.
- **block_hash** (*uint256, optional*) – The block hash to check.
- ****kwargs** – Extra keyword arguments.

Returns The merkle proof.

Return type *hexstr*

Raises **APIError** – Error thrown by node API. See message for details.

shutdown(***kwargs*) → *None*
Triggers node shutdown.

Parameters ****kwargs** – Extra keyword arguments.

Returns *None*

Raises **APIError** – Error thrown by node API. See message for details.

stop(***kwargs*) → *None*
Triggers node shutdown.

Parameters ****kwargs** – Extra keyword arguments.

Returns *None*

Raises **APIError** – Error thrown by node API. See message for details.

log_levels(*log_rules: List[LogRule]*, ***kwargs*) → *None*
Changes log level for the specified loggers.

Parameters

- **log_rules** (*List[LogRule]*) – A list of log rules to change.
- ****kwargs** – Extra keyword arguments.

Returns *None*

Raises **APIError** – Error thrown by node API. See message for details.

log_rules(***kwargs*) → *List[LogRule]*
Returns the enabled log rules.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of active log rules.

Return type *List[LogRule]*

Raises **APIError** – Error thrown by node API. See message for details.

async_loops(***kwargs*) → *List[AsyncLoopsModel]*
Gets the currently running async loops.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of active asynchronous loops.

Return type *List*[*AsyncLoopsModel*]

Raises *APIError* – Error thrown by node API. See message for details.

delete_datafolder_chain(***kwargs*) → None

Schedules data folder storing chain state in the datafolder for deletion on the next graceful shutdown.

Parameters ***kwargs* – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

rewind(*height: int, **kwargs*) → str

Signals the node to rewind to the specified height. This will be done via writing a flag to the .conf file so that on startup it be executed.

Parameters

- **height** (*int*) – The specified height to rewind to on restart.
- ***kwargs* – Extra keyword arguments.

Returns str

Raises *APIError* – Error thrown by node API. See message for details.

AsyncLoopsModel

class AsyncLoopsModel(**, loopName: str, status: str*)

A pydantic model for async loops.

loop_name: str

The loop name.

status: str

The loop status.

BlockHeaderModel

class BlockHeaderModel(**, version: int, merkleroot: hexstr, nonce: int, bits: str, previousblockhash: uint256 = None, time: datetime.datetime*)

A pydantic model representing the block header.

version: int

The version of the block.

merkleroot: *hexstr*

The merkle root of the block.

nonce: int

The nonce.

bits: str

The block's bits.

previous_blockhash: *Optional*[*uint256*]

The previous blockhash.

time: *datetime.datetime*

The time of the block.

ConnectedPeerModel

class ConnectedPeerModel(**version: str, remoteSocketEndpoint: str, tipHeight: int*)

A pydantic model representing a connected peer.

version: str

The connectedpeer's version.

remote_socket_endpoint: str

The connected peer's remote socket endpoint.

tip_height: int

The connected peer's tip height.

FeaturesDataModel

class FeaturesDataModel(**namespace: str, state: FeatureInitializationState*)

A pydantic model for features data.

namespace: str

The feature namespace.

state: FeatureInitializationState

The feature initialization state.

GetTxOutModel

class GetTxOutModel(**bestblock: uint256, confirmations: int, value: Money, scriptPubKey: ScriptPubKey, coinbase: bool*)

A pydantic model for get tx out response.

best_block: uint256

The highest block where utxo was present.

confirmations: int

The numberof confirmations.

value: Money

The value of the utxo.

script_pubkey: ScriptPubKey

The utxo scriptpubkey.

coinbase: bool

If true, the utxo originated as a coinbase transaction.

StatusModel

class StatusModel(**agent: str, version: str, externalAddress: str, network: str, coinTicker: str, processId: str, consensusHeight: int = None, blockStoreHeight: int, bestPeerHeight: int = None, inboundPeers: List[ConnectedPeerModel], outboundPeers: List[ConnectedPeerModel], featuresData: List[FeaturesDataModel], dataDirectoryPath: str, runningTime: str, difficulty: float = None, protocolVersion: int, testnet: bool, relayFee: Money, state: FullNodeState, inIbd: bool, headerHeight: int*)

A pydantic model for node status information.

agent: str
The node's agent string.

version: str
The node version.

external_address: str
The node external address.

network: str
The network.

coin_ticker: str
The ticker string of the current network's coin.

process_id: str
The process id of the node.

consensus_height: Optional[int]
The current consensus height.

blockstore_height: int
The current blockstore height.

best_peer_height: Optional[int]
The highest block height among connected peers.

inbound_peers: List[[ConnectedPeerModel](#)]
A list of inbound connected peers.

outbound_peers: List[[ConnectedPeerModel](#)]
A list of outbound connected peers.

features_data: List[[FeaturesDataModel](#)]
A list of features active on the node.

data_directory_path: str
The path to the node's data directory.

running_time: str
The node's current running time.

difficulty: Optional[float]
The current difficulty, if applicable.

protocol_version: int
The current protocol version.

testnet: bool
If true, node is on testnet.

relay_fee: [Money](#)
The network transaction relay fee.

state: [FullNodeState](#)
The node state model.

in_ibd: bool
If true, the node is in IBD state.

header_height: int
The header height.

TransactionModel

```
class TransactionModel(*, hex: hexstr, txid: uint256, hash: uint256, version: int, size: int, vsize: int, weight: int, locktime: int, vin: List[VIn], vout: List[VOut], blockhash: uint256 = None, confirmations: int = None, time: datetime.datetime = None, blocktime: datetime.datetime = None)
```

A pydantic model for a transaction.

hex: *hexstr*

The transaction hex.

txid: *uint256*

The transaction hash.

hash: *uint256*

The transaction hash.

version: *int*

The transaction version.

size: *int*

The transaction size.

vsize: *int*

The transaction vsize.

weight: *int*

The transaction weight.

locktime: *int*

The transaction locktime.

vin: *List[VIn]*

A list of VIn.

vout: *List[VOut]*

A list of VOut.

blockhash: *Optional[uint256]*

The hash of the block containing the transaction.

confirmations: *Optional[int]*

The number of confirmations of the transaction.

time: *Optional[datetime.datetime]*

The transaction time.

blocktime: *Optional[datetime.datetime]*

The blocktime.

ValidateAddressModel

```
class ValidateAddressModel(*, isvalid: bool, address: Address, scriptPubKey: str, isscript: bool, iswitness: bool)
```

A pydantic model for address validation.

isvalid: *bool*

True if the address is valid.

address: *Address*

The address.

scriptPubKey: str
The scriptPubKey.

isscript: bool
True if is a script address.

iswitness: bool
True if is a witness address.

1.1.1.20 Notifications

Notifications

class Notifications(**kwargs)
Implements the notifications api endpoints.

sync(sync_from: Union[uint256, str], **kwargs) → None
Begin synchronizing the chain from the provided block height or block hash.

Parameters

- **sync_from** (uint256, str) – The block hash to start syncing at.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

1.1.1.21 RPC

RPC

class RPC(**kwargs)
Implements the rpc api endpoints.

call_by_name(command: str, **kwargs) → *RPCCommandResponseModel*
Calls the specified RPC command.

Parameters

- **command** (str) – The complete RPC command.
- ****kwargs** – Extra keyword arguments.

Returns The command output.

Return type The *RPCCommandResponse*

Raises *APIError* – Error thrown by node API. See message for details.

list_methods(**kwargs) → List[*RPCCommandListModel*]
List available RPC call methods on this node.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of valid RPC commands.

Return type List[*RPCCommandListModel*]

Raises *APIError* – Error thrown by node API. See message for details.

RPCCommandListModel

class `RPCCommandListModel`(*, *command: str, description: str*)

A pydantic model for a RPC command.

command: `str`

The RPC command.

description: `str`

The command description.

RPCCommandResponseModel

class `RPCCommandResponseModel`(*, *value: dict*)

A pydantic model for a RPC response.

value: `dict`

The response.

1.1.1.22 SignalR

SignalR

class `SignalR`(***kwargs*)

Implements the signalr api endpoints.

get_connection_info(***kwargs*) → *GetConnectionInfoModel*

Returns the signalr connection info.

Parameters ****kwargs** – Extra keyword arguments.

Returns Information on the SignalR service.

Return type *GetConnectionInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

GetConnectionInfoModel

class `GetConnectionInfoModel`(*, *signalRUri: str, signalRPort: int*)

A pydantic model for SignalR connection information.

signalr_uri: `str`

The SignalR uri.

signalr_port: `int`

The SignalR port.

1.1.1.23 SmartContracts

SmartContracts

class `SmartContracts(**kwargs)`

Implements the smartcontracts api endpoints.

code(*address: Union[Address, str], **kwargs*) → *GetCodeModel*

Gets the bytecode for a smart contract as a hexadecimal string.

Parameters

- **address** (*Address, str*) – The smart contract address containing the contract bytecode.
- ****kwargs** – Extra keyword arguments.

Returns The smart contract code.

Return type *GetCodeModel*

Raises *APIError* – Error thrown by node API. See message for details.

balance(*address: Union[Address, str], **kwargs*) → *Money*

Gets the balance of a smart contract in strax or sidechain coin.

Parameters

- **address** (*Address, str*) – The smart contract address.
- ****kwargs** – Extra keyword arguments.

Returns The smart contract balance.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

storage(*contract_address: Union[Address, str], storage_key: str, data_type: int, **kwargs*) → Union[bool, bytes, str, *uint32, uint64, int32, int64, Address, bytearray, uint128, uint256*]

Gets a single piece of smart contract data. Returns a serialized string, if exists.

Parameters

- **contract_address** (*Address, str*) – The smart contract address being called.
- **storage_key** (*str*) – The key in the key-value store.
- **data_type** – The data type. Allowed values: [1,12]
- ****kwargs** – Extra keyword arguments.

Returns The smart contract information retrieved from storage.

Return type *Union[bool, bytes, str, uint32, uint64, int32, int64, Address, bytearray, uint128, uint256]*

Raises

- *APIError* – Error thrown by node API. See message for details.
- *RuntimeError* –

receipt(*tx_hash: Union[uint256, str], **kwargs*) → *ReceiptModel*

Gets a smart contract transaction receipt.

Parameters

- **tx_hash** (`uint256`, `str`) – The transaction hash of the smart contract receipt.
- ****kwargs** – Extra keyword arguments.

Returns The smart contract transaction receipt.

Return type `ReceiptModel`

Raises `APIError` – Error thrown by node API. See message for details.

receipt_search(*contract_address: Union[Address, str], topics: Optional[List[str]] = None, event_name: Optional[str] = None, from_block: int = 0, to_block: Optional[int] = None, **kwargs*) → List[`ReceiptModel`]

Searches a smart contract's receipts for those which match a specific event.

Parameters

- **contract_address** (`Address`, `str`) – The address for the smart contract.
- **event_name** (`str`, *optional*) – The event to search for.
- **topics** (`List[str]`, *optional*) – A list of topics to search for.
- **from_block** (`int`) – Block to start search from.
- **to_block** (`int`, *optional*) – Block to search up to.
- ****kwargs** – Extra keyword arguments.

Returns A list of receipts.

Return type List[`ReceiptModel`]

Raises `APIError` – Error thrown by node API. See message for details.

build_create(*wallet_name: str, fee_amount: Union[Money, int, float, decimal.Decimal], password: str, contract_code: Union[hexstr, str], gas_price: int, gas_limit: int, sender: Union[Address, str], amount: Union[Money, int, float, decimal.Decimal], outpoints: Optional[List[Outpoint]] = None, account_name: str = 'account 0', parameters: Optional[List[Union[str, SmartContractParameter]]] = None, **kwargs*) → `BuildCreateContractTransactionModel`

Builds a transaction to create a smart contract.

Parameters

- **wallet_name** (`str`) – The wallet name.
- **account_name** (`str`, *optional*) – The wallet name. Default='account 0'
- **outpoints** (`List[Outpoint]`, *optional*) – A list of the outpoints used to construct the transaction.
- **amount** (`Money`, `int`, `float`, `Decimal`, *optional*) – The amount being sent.
- **fee_amount** (`Money`, `int`, `float`, `Decimal`) – The fee amount.
- **password** (`SecretStr`) – The password.
- **contract_code** (`hexstr`, `str`) – The smart contract code hexstring.
- **gas_price** (`int`) – The amount of gas being used in satoshis.
- **gas_limit** (`int`) – The maximum amount of gas that can be used in satoshis.
- **sender** (`Address`, `str`) – The address of the sending address.
- **parameters** (`List[Union[SmartContractParameter, str]]`, *optional*) – A list of parameters for the smart contract.

- ****kwargs** – Extra keyword arguments.

Returns A built create smart contract transaction.

Return type `BuildCreateContractTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

build_call(*wallet_name: str, fee_amount: Union[Money, int, float, decimal.Decimal], password: str, contract_address: Union[Address, str], method_name: str, gas_price: int, gas_limit: int, sender: Union[Address, str], amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, outpoints: Optional[List[Outpoint]] = None, account_name: str = 'account 0', parameters: Optional[List[Union[str, SmartContractParameter]]] = None, **kwargs*) → `BuildContractTransactionModel`

Builds a transaction to call a smart contract method.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The wallet name. Default='account 0'
- **outpoints** (*List[Outpoint], optional*) – A list of the outpoints used to construct the transaction.
- **contract_address** (*Address, str*) – The smart contract address being called.
- **method_name** (*str*) – The method name being called.
- **amount** (*Money, int, float, Decimal, optional*) – The amount being sent.
- **fee_amount** (*Money, int, float, Decimal*) – The fee amount.
- **password** (*SecretStr*) – The password.
- **gas_price** (*int*) – The amount of gas being used in satoshis.
- **gas_limit** (*int*) – The maximum amount of gas that can be used in satoshis.
- **sender** (*Address, str*) – The address of the sending address.
- **parameters** (*List[Union[SmartContractParameter, str]], optional*) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns A built smart contract transaction.

Return type `BuildContractTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

build_transaction(*sender: Union[Address, str], password: str, wallet_name: str, recipients: List[Recipient], op_return_data: Optional[str] = None, outpoints: Optional[List[Outpoint]] = None, op_return_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, fee_type: Optional[str] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, change_address: Optional[Union[Address, str]] = None, account_name: str = 'account 0', segwit_change_address: bool = False, fee_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, **kwargs*) → `BuildContractTransactionModel`

Build a transaction to transfer funds on a smart contract network.

Parameters

- **sender** (*Address*) – The sender address.

- **fee_amount** (*Money, int, float, Decimal, optional*) – The fee amount.
- **password** (*SecretStr*) – The password.
- **segwit_change_address** (*bool, optional*) – If the change address is a segwit address. Default=False.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint]*) – A list of the outpoints used to construct the transaction.
- **recipients** (*List[Recipient]*) – A list of the recipients, including amounts, for the transaction.
- **op_return_data** (*str, optional*) – OP_RETURN data to include with the transaction.
- **op_return_amount** (*Money, int, float, Decimal, optional*) – Amount to burn in the OP_RETURN transaction.
- **fee_type** (*str, optional*) – low, medium, or high.
- **allow_unconfirmed** (*bool, optional*) – If True, allow unconfirmed transactions in the estimation. Default=False
- **shuffle_outputs** (*bool, optional*) – If True, shuffles outputs. Default=False.
- **change_address** (*Address, optional*) – Sends output sum less amount sent to recipients to this designated change address, if provided.
- ****kwargs** – Extra keyword arguments.

Returns A built smart contract transaction.

Return type `BuildContractTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

`estimate_fee(sender: Union[Address, str], wallet_name: str, recipients: List[Recipient], fee_type: str, outpoints: Optional[List[Outpoint]] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, op_return_data: Optional[str] = None, op_return_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, account_name: str = 'account 0', change_address: Optional[Union[Address, str]] = None, **kwargs) → Money`

Gets a fee estimate for a specific smart contract account-based transfer transaction.

Parameters

- **sender** (*Address, str*) – The sender address.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint], optional*) – A list of the outpoints used to construct the transaction.
- **recipients** (*List[Recipient]*) – A list of the recipients, including amounts, for the transaction.
- **op_return_data** (*str, optional*) – OP_RETURN data to include with the transaction.
- **op_return_amount** (*Money, int, float, Decimal, optional*) – Amount to burn in the OP_RETURN transaction.
- **fee_type** (*str, optional*) – low, medium, or high.

- **allow_unconfirmed** (*bool, optional*) – If True, allow unconfirmed transactions in the estimation. Default=False
- **shuffle_outputs** (*bool, optional*) – If True, shuffles outputs. Default=False.
- **change_address** (*Address, str, optional*) – Sends output sum less amount sent to recipients to this designated change address, if provided.
- ****kwargs** – Extra keyword arguments.

Returns The fee estimate.

Return type `Money`

Raises `APIError` – Error thrown by node API. See message for details.

```
build_and_send_create(wallet_name: str, fee_amount: Union[Money, int, float, decimal.Decimal],
    password: str, contract_code: Union[hexstr, str], gas_price: int, gas_limit: int,
    sender: Union[Address, str], amount: Optional[Union[Money, int, float,
    decimal.Decimal]] = None, outpoints: Optional[List[Outpoint]] = None,
    account_name: str = 'account 0', parameters: Optional[List[Union[str,
    SmartContractParameter]]] = None, **kwargs) →
    BuildCreateContractTransactionModel
```

Builds a transaction to create a smart contract and then broadcasts.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The wallet name. Default='account 0'
- **outpoints** (*List[Outpoint], optional*) – A list of the outpoints used to construct the transaction.
- **amount** (*Money, int, float, Decimal, optional*) – The amount being sent.
- **fee_amount** (*Money, int, float, Decimal*) – The fee amount.
- **password** (*SecretStr*) – The password.
- **contract_code** (*hexstr, str*) – The smart contract code hexstring.
- **gas_price** (*int*) – The amount of gas being used in satoshis.
- **gas_limit** (*int*) – The maximum amount of gas that can be used in satoshis.
- **sender** (*Address, str*) – The address of the sending address.
- **parameters** (*List[Union[SmartContractParameter, str]], optional*) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns A built create smart contract transaction.

Return type `BuildCreateContractTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

build_and_send_call(*wallet_name: str, fee_amount: Union[Money, int, float, decimal.Decimal], password: str, contract_address: Union[Address, str], method_name: str, gas_price: int, gas_limit: int, sender: Union[Address, str], amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, outpoints: Optional[List[Outpoint]] = None, account_name: str = 'account 0', parameters: Optional[List[Union[str, SmartContractParameter]]] = None, **kwargs*) → *BuildContractTransactionModel*

Builds a transaction to call a smart contract method and then broadcasts.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The wallet name. Default='account 0'
- **outpoints** (*List[Outpoint], optional*) – A list of the outpoints used to construct the transaction.
- **contract_address** (*Address, str*) – The smart contract address being called.
- **method_name** (*str*) – The method name being called.
- **amount** (*Money, int, float, Decimal, optional*) – The amount being sent.
- **fee_amount** (*Money, int, float, Decimal*) – The fee amount.
- **password** (*SecretStr*) – The password.
- **gas_price** (*int*) – The amount of gas being used in satoshis.
- **gas_limit** (*int*) – The maximum amount of gas that can be used in satoshis.
- **sender** (*Address, str*) – The address of the sending address.
- **parameters** (*List[Union[SmartContractParameter, str]], optional*) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns A built smart contract transaction.

Return type *BuildContractTransactionModel*

Raises *APIError* – Error thrown by node API. See message for details.

local_call(*contract_address: Union[Address, str], method_name: str, amount: Union[Money, int, float, decimal.Decimal], gas_price: int, gas_limit: int, sender: Union[Address, str], block_height: Optional[int] = None, parameters: Optional[List[Union[str, SmartContractParameter]]] = None, **kwargs*) → *LocalExecutionResultModel*

Makes a local call to a method on a smart contract that has been successfully deployed. The purpose is to query and test methods.

Parameters

- **contract_address** (*Address, str*) – The smart contract address being called.
- **method_name** (*str*) – The smart contract method being called.
- **amount** (*Money, int, float, Decimal*) – The amount being sent.
- **gas_price** (*int*) – The amount of gas being used in satoshis.
- **gas_limit** (*int*) – The maximum amount of gas that can be used in satoshis.
- **sender** (*Address, str*) – The address of the sending address.

- **block_height** (*int, optional*) – The height at which to query the contract’s state. If unset, will default to the current chain tip.
- **parameters** (*List[Union[SmartContractParameter, str]], optional*) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns The results of a local contract execution.

Return type `LocalExecutionResultModel`

Raises `APIError` – Error thrown by node API. See message for details.

address_balances (*wallet_name: str, **kwargs*) → `List[AddressBalanceModel]`

Gets all addresses owned by a wallet which have a balance associated with them.

Parameters

- **wallet_name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns A list of addresses with balance information.

Return type `List[AddressBalanceModel]`

Raises `APIError` – Error thrown by node API. See message for details.

AddressBalanceModel

class `AddressBalanceModel`(*, *address: Address, sum: Money*)

A pydantic model for an address balance.

address: `Address`

The address.

sum: `Money`

The amount present at the address.

BuildContractTransactionModel

class `BuildContractTransactionModel`(*, *fee: Money, hex: hexstr, message: str = None, success: bool = None, transactionId: uint256 = None*)

A pydantic model for building a smart contact transaction.

fee: `Money`

The transaction fee.

hex: `hexstr`

The hex serialized transaction.

message: `Optional[str]`

The build transaction message.

success: `Optional[bool]`

True if build was successful.

transaction_id: `Optional[uint256]`

The transaction hash, if build successful.

BuildCreateContractTransactionModel

```
class BuildCreateContractTransactionModel(*, fee: Money, hex: hexstr, message: str = None, success:
    bool = None, transactionId: uint256 = None,
    newContractAddress: Address)
```

A pydantic model for a create smart contract transaction.

new_contract_address: *Address*

The new address associated with the smart contract.

fee: *Money*

The transaction fee.

hex: *hexstr*

The hex serialized transaction.

message: *Optional[str]*

The build transaction message.

success: *Optional[bool]*

True if build was successful.

transaction_id: *Optional[uint256]*

The transaction hash, if build successful.

GetCodeModel

```
class GetCodeModel(*, type: str, bytecode: str, csharp: str, message: str = None)
```

A pydantic model for the smart contract code request.

type: *str*

The code type.

bytecode: *str*

The contract bytecode.

csharp: *str*

The csharp code.

message: *Optional[str]*

A message from response.

LocalExecutionResultModel

```
class LocalExecutionResultModel(*, internalTransfers:
    List[pystratis.api.smartcontracts.responsemodels.transferinfomodel.TransferInfoModel]
    = None, gasConsumed: int, revert: bool = None, errorMessage: str =
    None, return: Any = None, logs:
    List[pystratis.api.smartcontracts.responsemodels.logmodel.LogModel] =
    None)
```

A pydantic model representing the result of a local smart contract execution call.

internal_transfers: *Optional[List[TransferInfoModel]]*

A list of internal transfers.

gas_consumed: *int*

The amount of gas consumed by the call.

revert: `Optional[bool]`
If true, call was not successful.

error_message: `Optional[str]`
An error message, if thrown.

return_obj: `Optional[Any]`
An optional return object.

logs: `Optional[List[LogModel]]`
An optional list of logs returned.

LogModel

class LogModel(**, address: Address = None, topics: List[str] = None, data: str = None, log: dict = None*)
A pydantic model of a smart contact log.

address: `Optional[Address]`
The smart contact address.

topics: `Optional[List[str]]`
A list of topics.

data: `Optional[str]`
Log data.

log: `Optional[dict]`
Log dict object response.

ReceiptModel

class ReceiptModel(**, transactionHash: pystratis.core.types.uint256.uint256, blockHash: pystratis.core.types.uint256.uint256, postState: pystratis.core.types.uint256.uint256 = None, gasUsed: int = None, from: pystratis.core.types.address.Address = None, to: pystratis.core.types.address.Address = None, newContractAddress: pystratis.core.types.address.Address = None, success: bool, returnValue: str = None, bloom: pystratis.core.types.hexstr.hexstr = None, error: str = None, logs: List[pystratis.api.smartcontracts.responsemodels.logmodel.LogModel] = None, blockNumber: int*)

A pydantic model of a smart contact receipt.

transaction_hash: `uint256`
The transaction hash.

block_hash: `uint256`
The hash of the block containing the transaction.

post_state: `Optional[uint256]`
The smart contact state after execution.

gas_used: `Optional[int]`
The amount of gas used.

from_address: `Optional[Address]`
Sending address, if applicable.

to_address: `Optional[Address]`
Receiving address, if applicable.

new_contract_address: `Optional[Address]`
A new contract address, if creation transaction.

success: `bool`
True if transaction successful.

return_value: `Optional[str]`
Transaction return value, if applicable.

bloom: `Optional[hexstr]`
The bloom filter.

error: `Optional[str]`
Error message, if present.

logs: `Optional[List[LogModel]]`
Smart contact log model data, if present.

block_number: `int`
The block number.

TransferInfoModel

class TransferInfoModel(**from*: *pystratis.core.types.address.Address*, *to*:
pystratis.core.types.address.Address, *value*: *pystratis.core.types.money.Money*)
A pydantic model of a smart contact transfer.

from_address: `Address`
The sending address.

to_address: `Address`
The receiving address.

value: `Money`
The amount sent.

1.1.1.24 SmartContractWallet

SmartContractWallet

class SmartContractWallet(***kwargs*)
Implements the smartcontractwallet api endpoints.

account_addresses(*wallet_name*: *str*, ***kwargs*) → `List[Address]`
Gets a smart contract account address.

Parameters

- **wallet_name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns A list of smart contract addresses.

Return type `List[Address]`

Raises `APIError` – Error thrown by node API. See message for details.

address_balance(*address*: `Union[Address, str]`, ***kwargs*) → `Money`
Gets the balance at a specific wallet address in STRAX (or the sidechain coin).

Parameters

- **address** (*Address*, *str*) – The smart contract address being queried.
- ****kwargs** – Extra keyword arguments.

Returns The smart contract address balance.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

history(*wallet_name: str*, *address: Union[Address, str]*, *skip: int = 0*, *take: Optional[int] = None*, ***kwargs*) → *List[ContractTransactionItemModel]*

Gets the history of a specific smart contract wallet address.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **address** (*Address*, *str*) – The address to query the history.
- **skip** (*int*, *optional*) – Skip this many items. Default=0.
- **take** (*int*, *optional*) – Take this many items.
- ****kwargs** – Extra keyword arguments.

Returns A history of a smart contract wallet address.

Return type *List[ContractTransactionItemModel]*

Raises *APIError* – Error thrown by node API. See message for details.

create(*wallet_name: str*, *fee_amount: Union[Money, int, float, decimal.Decimal]*, *password: str*, *contract_code: Union[hexstr, str]*, *gas_price: int*, *gas_limit: int*, *sender: Union[Address, str]*, *amount: Optional[Union[Money, int, float, decimal.Decimal]] = None*, *outpoints: Optional[List[Outpoint]] = None*, *account_name: str = 'account 0'*, *parameters: Optional[List[str]] = None*, ***kwargs*) → *uint256*

Builds a transaction to create a smart contract and then broadcasts the transaction to the network.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str*, *optional*) – The wallet name. Default='account 0'
- **outpoints** (*List[Outpoint]*, *optional*) – A list of the outpoints used to construct the transaction.
- **amount** (*Money*, *int*, *float*, *Decimal*, *optional*) – The amount being sent.
- **fee_amount** (*Money*, *int*, *float*, *Decimal*) – The fee amount.
- **password** (*SecretStr*) – The password.
- **contract_code** (*hexstr*, *str*) – The smart contract code hexstring.
- **gas_price** (*int*) – The amount of gas being used in satoshis.
- **gas_limit** (*int*) – The maximum amount of gas that can be used in satoshis.
- **sender** (*Address*, *str*) – The address of the sending address.
- **parameters** (*List[Union[SmartContractParameter, str]]*, *optional*) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns The transaction hash.

Return type `uint256`

Raises `APIError` – Error thrown by node API. See message for details.

`call(wallet_name: str, fee_amount: Union[Money, int, float, decimal.Decimal], password: str, contract_address: Union[Address, str], method_name: str, gas_price: int, gas_limit: int, sender: Union[Address, str], amount: Union[Money, int, float, decimal.Decimal], outpoints: Optional[List[Outpoint]] = None, account_name: str = 'account 0', parameters: Optional[List[str]] = None, **kwargs) → BuildContractTransactionModel`

Builds a transaction to call a smart contract method and then broadcasts the transaction to the network.

Parameters

- **wallet_name** (`str`) – The wallet name.
- **account_name** (`str`, `optional`) – The wallet name. Default='account 0'
- **outpoints** (`List[Outpoint]`, `optional`) – A list of the outpoints used to construct the transaction.
- **contract_address** (`Address`, `str`) – The smart contract address being called.
- **method_name** (`str`) – The method name being called.
- **amount** (`Money`, `int`, `float`, `Decimal`, `optional`) – The amount being sent.
- **fee_amount** (`Money`, `int`, `float`, `Decimal`) – The fee amount.
- **password** (`SecretStr`) – The password.
- **gas_price** (`int`) – The amount of gas being used in satoshis.
- **gas_limit** (`int`) – The maximum amount of gas that can be used in satoshis.
- **sender** (`Address`, `str`) – The address of the sending address.
- **parameters** (`List[Union[SmartContractParameter, str]]`, `optional`) – A list of parameters for the smart contract.
- ****kwargs** – Extra keyword arguments.

Returns A built smart contract transaction.

Return type `BuildContractTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

`send_transaction(transaction_hex: Union[hexstr, str], **kwargs) → WalletSendTransactionModel`

Broadcasts a transaction, which either creates a smart contract or calls a method on a smart contract.

Parameters

- **transaction_hex** (`hexstr`, `str`) – The transaction hex string.
- ****kwargs** – Extra keyword arguments.

Returns Information about the sent transaction.

Return type `WalletSendTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

BuildContractTransactionModel

```
class BuildContractTransactionModel(*, fee: Money, hex: hexstr, message: str = None, success: bool =
    None, transactionId: uint256 = None)
```

A pydantic model for building a smart contract transaction.

fee: *Money*

The transaction fee.

hex: *hexstr*

The hex serialized transaction.

message: *Optional[str]*

The build transaction message.

success: *Optional[bool]*

True if build was successful.

transaction_id: *Optional[uint256]*

The transaction hash, if build successful.

ContractTransactionItemModel

```
class ContractTransactionItemModel(*, blockHeight: int, type: ContractTransactionItemType, hash: uint256,
    to: Address, amount: Money, transactionFee: Money, gasFee:
    Money)
```

A pydantic model representing a contract transaction.

block_height: *int*

The block height of block containing the transaction.

item_type: *ContractTransactionItemType*

The contract transaction item type.

hash: *uint256*

The transaction hash.

to_address: *Address*

The to address of the contact.

amount: *Money*

The transaction amount.

transaction_fee: *Money*

The transaction fee.

gas_fee: *Money*

The transaction's gas fee.

TransactionOutputModel

```
class TransactionOutputModel(*, address: Optional[Union[int, Address]] = None, amount: Money,
                             opReturnData: str = None)
```

A pydantic model of a transaction output.

address: `Optional[Union[int, Address]]`

The address receiving the output.

amount: `Money`

The output amount.

op_return_data: `Optional[str]`

The OP_RETURN data, if present.

WalletSendTransactionModel

```
class WalletSendTransactionModel(*, transactionId: uint256, outputs: List[TransactionOutputModel])
```

A pydantic model for a send transaction response.

transaction_id: `uint256`

The transaction hash.

outputs: `List[TransactionOutputModel]`

A list of transaction outputs.

1.1.1.25 Staking

Staking

```
class Staking(**kwargs)
```

Implements the staking api endpoints.

get_staking_info(**kwargs) → *GetStakingInfoModel*

Gets current staking information.

Parameters ****kwargs** – Extra keyword arguments.

Returns Information about current staking status.

Return type *GetStakingInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

start_staking(name: str, password: str, **kwargs) → None

Start staking

Parameters

- **name** (*str*) – The wallet name.
- **password** (*str*) – The wallet password.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

start_multistaking(wallet_credentials: List[WalletSecret], **kwargs) → None

Start staking for multiple wallets simultaneously

Parameters

- **wallet_credentials** (*List[WalletSecret]*) – A list of wallet credentials to launch staking of multiple wallets with one command.
- ****kwargs** – Extra keyword arguments.

Returns None**Raises** *APIError* – Error thrown by node API. See message for details.**stop_staking**(***kwargs*) → None

Stop staking.

Parameters ****kwargs** – Extra keyword arguments.**Returns** None**Raises** *APIError* – Error thrown by node API. See message for details.**GetStakingInfoModel**

```
class GetStakingInfoModel(*, enabled: bool, staking: bool, errors: str = None, currentBlockSize: int,
                           currentBlockTx: int, pooledTx: int, difficulty: float, searchInterval: int, weight: int,
                           netStakeWeight: int = None, immature: int, expectedTime: int)
```

A pydantic model for staking information.

enabled: bool

If true, staking is enabled.

staking: bool

If true, is currently staking.

errors: Optional[str]

Error messages, if present.

current_blocksize: int

The current block size.

current_block_tx: int

The current number of block transactions.

pooled_tx: int

The number of pooled transactions.

difficulty: float

The current difficulty.

search_interval: int

The search interval.

weight: int

The current staking weight.

net_stake_weight: Optional[int]

The network staking weight.

immature: int

The number of immature coins that can't stake.

expected_time: int

The expected number of seconds between stakes.

1.1.1.26 Voting

Voting

class Voting(**kwargs)

Implements the voting api endpoints.

pending_polls(vote_type: int, pubkey_of_member_being_voted_on: Union[hexstr, str], **kwargs) → List[PollViewModel]

Gets a list of pending polls.

Parameters

- **vote_type** (VoteKey, optional) – The type of vote to query.
- **pubkey_of_member_being_voted_on** (PubKey, optional) – The pubkey to query.
- ****kwargs** – Extra keyword arguments.

Returns A list of pending polls.

Return type List[PollViewModel]

Raises **APIError** – Error thrown by node API. See message for details.

finished_polls(vote_type: int, pubkey_of_member_being_voted_on: Union[hexstr, str], **kwargs) → List[PollViewModel]

Gets a list of finished polls.

Parameters

- **vote_type** (VoteKey, optional) – The type of vote to query.
- **pubkey_of_member_being_voted_on** (PubKey, optional) – The pubkey to query.
- ****kwargs** – Extra keyword arguments.

Returns A list of finished polls.

Return type List[PollViewModel]

Raises **APIError** – Error thrown by node API. See message for details.

executed_polls(vote_type: int, pubkey_of_member_being_voted_on: Union[hexstr, str], **kwargs) → List[PollViewModel]

Gets a list of executed polls.

Parameters

- **vote_type** (VoteKey, optional) – The type of vote to query.
- **pubkey_of_member_being_voted_on** (hexstr, str, optional) – The pubkey to query.
- ****kwargs** – Extra keyword arguments.

Returns A list of executed polls.

Return type List[PollViewModel]

Raises **APIError** – Error thrown by node API. See message for details.

whitelisted_hashes(**kwargs) → List[WhitelistedHashesModel]

Gets a list of whitelisted hashes.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of whitelisted hashes.

Return type `List[WhitelistedhashesModel]`

Raises `APIError` – Error thrown by node API. See message for details.

schedulevote_whitelisthash(*hash_id: Union[uint256, str], **kwargs*) → None

Vote to add a hash from whitelist.

Parameters

- **hash_id** (`uint256`, `str`) – The hash to whitelist.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises `APIError` – Error thrown by node API. See message for details.

schedulevote_removehash(*hash_id: Union[uint256, str], **kwargs*) → None

Vote to remove a hash from whitelist.

Parameters

- **hash_id** (`uint256`, `str`) – The hash to remove.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises `APIError` – Error thrown by node API. See message for details.

schedulevote_kickmember(*pubkey: Union[hexstr, str], **kwargs*) → None

Vote to remove a hash from whitelist.

Parameters

- **pubkey** (`hexstr`, `str`) – The pubkey to vote on kicking.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises `APIError` – Error thrown by node API. See message for details.

scheduled_votes(***kwargs*) → `List[VotingDataModel]`

Gets the scheduled voting data.

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of voting data.

Return type `List[VotingDataModel]`

Raises `APIError` – Error thrown by node API. See message for details.

polls_tip(***kwargs*) → `int`

Gets the tip of the polls repository.

Parameters ****kwargs** – Extra keyword arguments.

Returns The polls repository tip.

Return type `int`

Raises `APIError` – Error thrown by node API. See message for details.

PollViewModel

```
class PollViewModel(*, IsPending: bool, IsExecuted: bool, Id: int, PollVotedInFavorBlockDataHash: uint256 = None, PollVotedInFavorBlockDataHeight: int = None, PollStartFavorBlockDataHash: uint256 = None, PollStartFavorBlockDataHeight: int = None, PollExecutedBlockDataHash: uint256 = None, PollExecutedBlockDataHeight: int = None, PubKeysHexVotedInFavor: List[PubKey], VotingDataString: str)
```

A pydantic model for polling data.

is_pending: **bool**

If true, poll is pending.

is_executed: **bool**

If true, poll has been executed.

poll_id: **int**

The poll id.

poll_voted_in_favor_blockdata_hash: **Optional[uint256]**

If voted in favor, the block of the vote.

poll_voted_in_favor_blockdata_height: **Optional[int]**

If voted in favor, the height of the block.

poll_start_favor_blockdata_hash: **Optional[uint256]**

The block hash when polling started.

poll_start_favor_blockdata_height: **Optional[int]**

The block height when polling started.

poll_executed_blockdata_hash: **Optional[uint256]**

The block hash when poll was executed, if executed.

poll_executed_blockdata_height: **Optional[int]**

The block height when poll was executed, if executed.

pubkeys_hex_voted_in_favor: **List[PubKey]**

A list of pubkeys voting in favor of poll.

voting_data_string: **str**

Voting data.

VotingDataModel

```
class VotingDataModel(*, key: PubKey, hash: uint256)
```

A pydantic model representing voting data.

key: **PubKey**

The pubkey.

hash: **uint256**

The hash voted upon.

WhitelistedHashesModel

class `WhitelistedHashesModel`(*, *hash*: `uint256`)

A pydantic model for a whitelisted hash.

hash: `uint256`

A whitelisted hash.

1.1.1.27 Wallet

Wallet

class `Wallet`(***kwargs*)

Implements the wallet api endpoints.

mnemonic(*language*: *str* = 'English', *word_count*: *int* = 12, ***kwargs*) → List[str]

Generates a mnemonic to use for an HD wallet. For more information about mnemonics, see [BIP39](#).

Parameters

- **language** (*str*) – The language used to generate mnemonic.
- **word_count** (*int*) – Count of words needs to be generated.
- ****kwargs** – Extra keyword arguments.

Returns The generated mnemonic.

Return type List[str]

Raises [APIError](#) – Error thrown by node API. See message for details.

create(*name*: *str*, *password*: *str*, *passphrase*: *str*, *mnemonic*: Optional[str] = None, ***kwargs*) → List[str]

Creates a new wallet on this full node.

Parameters

- **mnemonic** (*str*, *optional*) – The mnemonic used to create a HD wallet. If not specified, it will be randomly generated underhood.
- **password** (*str*) – The password for a wallet to be created.
- **passphrase** (*str*) – The passphrase used in master key generation.
- **name** – (str): The name for a wallet to be created.
- ****kwargs** – Extra keyword arguments.

Returns The mnemonic used to generate this HD wallet.

Return type List[str]

Raises [APIError](#) – Error thrown by node API. See message for details.

sign_message(*wallet_name*: *str*, *password*: *str*, *external_address*: Union[Address, str], *message*: *str*, ***kwargs*) → str

Signs a message and returns the signature.

Parameters

- **wallet_name** (*str*) – The name of the wallet to sign message with.
- **password** (*str*) – The password of the wallet to sign message with.

- **external_address** ([Address](#), *str*) – The external address of a private key used to sign message.
- **message** (*str*) – The message to be signed.
- ****kwargs** – Extra keyword arguments.

Returns The signature of the message.

Return type *str*

Raises [APIError](#) – Error thrown by node API. See message for details.

pubkey(*wallet_name: str, external_address: Union[Address, str], **kwargs*) → [PubKey](#)
Gets the public key for an address.

Parameters

- **wallet_name** (*str*) – The name of the wallet to search for pubkey in.
- **external_address** ([Address](#), *str*) – The external address of a wanted pubkey.
- ****kwargs** – Extra keyword arguments.

Returns The requested public key.

Return type [PubKey](#)

Raises [APIError](#) – Error thrown by node API. See message for details.

verify_message(*signature: str, external_address: Union[Address, str], message: str, **kwargs*) → *bool*
Verifies the signature of a message.

Parameters

- **signature** (*str*) – The signature to be verified.
- **external_address** ([Address](#), *str*) – The address of the signer.
- **message** (*str*) – The message that was signed.
- ****kwargs** – Extra keyword arguments.

Returns True if signature is verified, False otherwise.

Return type *bool*

Raises [APIError](#) – Error thrown by node API. See message for details.

load(*name: str, password: str, **kwargs*) → *None*
Loads a previously created wallet.

Parameters

- **name** (*str*) – The wallet name to load.
- **password** (*str*) – The wallet password.
- ****kwargs** – Extra keyword arguments.

Returns *None*

Raises [APIError](#) – Error thrown by node API. See message for details.

recover(*mnemonic: str, password: str, passphrase: str, name: str, creation_date: Optional[str] = None, **kwargs*) → *None*
Recovers an existing wallet.

Parameters

- **mnemonic** (*str*) – A mnemonic for initializing a wallet.
- **password** (*str*) – The password for the wallet.
- **passphrase** (*str*) – The passphrase for the wallet.
- **name** (*str*) – The name for the wallet.
- **creation_date** (*str*, *datetime*, *optional*) – An estimate of the wallet creation date.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

recover_via_extpubkey(*extpubkey: Union[ExtPubKey, str, hexstr]*, *account_index: int*, *name: str*, *creation_date: str*, ***kwargs*) → None

Recovers a wallet using its extended public key.

Parameters

- **extpubkey** (*ExtPubKey*, *str*, *hexstr*) – The extpubkey for the recovered wallet.
- **account_index** (*int*) – The account index.
- **name** (*str*) – The wallet name.
- **creation_date** (*str*, *datetime*, *optional*) – An estimate of the wallet creation date.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises *APIError* – Error thrown by node API. See message for details.

general_info(*name: str*, ***kwargs*) → *WalletGeneralInfoModel*

Gets some general information about a wallet.

Parameters

- **name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns General information about the wallet.

Return type *WalletGeneralInfoModel*

Raises *APIError* – Error thrown by node API. See message for details.

transaction_count(*wallet_name: str*, *account_name: str = 'account 0'*, ***kwargs*) → int

Get the transaction count for the specified Wallet and Account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str*, *optional*) – The account name. Default='account 0'.
- ****kwargs** – Extra keyword arguments.

Returns The number of transactions.

Return type *int*

Raises *APIError* – Error thrown by node API. See message for details.

history(*wallet_name: str, account_name: str = 'account 0', address: Optional[Union[Address, str]] = None, skip: int = 0, take: Optional[int] = None, prev_output_tx_time: Optional[int] = None, prev_output_index: Optional[int] = None, search_query: Optional[str] = None, **kwargs*) → *WalletHistoryModel*

Gets the history of a wallet.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **address** (*Address, str, optional*) – The address to query the history.
- **skip** (*conint (ge=0), optional*) – The number of history items to skip.
- **take** (*conint (ge=0), optional*) – The number of history items to take.
- **prev_output_tx_time** (*conint (ge=0), optional*) – The previous output transaction time.
- **prev_output_index** (*conint (ge=0), optional*) – The previous output transaction index.
- **search_query** (*str, optional*) – A search query.
- ****kwargs** – Extra keyword arguments.

Returns The wallet history.

Return type *WalletHistoryModel*

Raises *APIError* – Error thrown by node API. See message for details.

balance(*wallet_name: str, account_name: str = 'account 0', include_balance_by_address: bool = False, **kwargs*) → *WalletBalanceModel*

Gets the balance of a wallet in STRAX (or sidechain coin). Both the confirmed and unconfirmed balance are returned.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **include_balance_by_address** (*bool, optional*) – If true, includes detailed information about balances by address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns The wallet balance.

Return type *WalletBalanceModel*

Raises *APIError* – Error thrown by node API. See message for details.

received_by_address(*address: Union[Address, str], **kwargs*) → *AddressBalanceModel*

Retrieves transactions received by the specified address.

Parameters

- **address** (*Address*) – The address to query.
- ****kwargs** – Extra keyword arguments.

Returns The transactions associated with the address.

Return type *AddressBalanceModel*

Raises ***APIError*** – Error thrown by node API. See message for details.

max_balance(*wallet_name: str, fee_type: str, account_name: str = 'account 0', allow_unconfirmed: bool = False, **kwargs*) → *MaxSpendableAmountModel*

Gets the maximum spendable balance for an account along with the fee required to spend it.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **fee_type** (*str*) – The fee type. Allowed [low, medium, high]
- **allow_unconfirmed** (*bool, optional*) – If True, allow unconfirmed utxo in request. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns Information about the maximum spendable amount and fee to send.

Return type *MaxSpendableAmountModel*

Raises ***APIError*** – Error thrown by node API. See message for details.

spendable_transactions(*wallet_name: str, account_name: str = 'account 0', min_confirmations: int = 0, **kwargs*) → *SpendableTransactionsModel*

Gets the spendable transactions for an account with the option to specify how many confirmations a transaction needs to be included.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **min_confirmations** (*int, optional*) – Get spendable transactions less this value from chain tip. Default=0.
- ****kwargs** – Extra keyword arguments.

Returns Spendable transactions.

Return type *SpendableTransactionsModel*

Raises ***APIError*** – Error thrown by node API. See message for details.

estimate_txfee(*wallet_name: str, outpoints: List[Outpoint], recipients: List[Recipient], op_return_data: Optional[str] = None, op_return_amount: Optional[Money] = None, fee_type: Optional[str] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, change_address: Optional[Address] = None, account_name: str = 'account 0', **kwargs*) → *Money*

Gets a fee estimate for a specific transaction.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint]*) – A list of outpoints to include in the transaction.
- **recipients** (*List[Recipient]*) – A list of recipients with amounts.
- **op_return_data** (*str, optional*) – The OP_RETURN data.

- **op_return_amount** (*Money, int, float, Decimal, optional*) – The amount to burn in OP_RETURN.
- **fee_type** (*str, optional*) – The fee type. Allowed [low, medium, high]
- **allow_unconfirmed** (*bool, optional*) – If True, includes unconfirmed outputs. Default=False.
- **shuffle_outputs** (*bool, optional*) – If True, shuffle outputs. Default=False.
- **change_address** (*Address, str, optional*) – Specify a change address. If not set, a new change address is used.
- ****kwargs** – Extra keyword arguments.

Returns An estimate of the transaction fee.

Return type *Money*

Raises *APIError* – Error thrown by node API. See message for details.

build_transaction(*wallet_name: str, password: str, outpoints: List[Outpoint], recipients: List[Recipient], fee_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, segwit_change_address: bool = False, op_return_data: Optional[str] = None, op_return_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, fee_type: Optional[str] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, account_name: str = 'account 0', change_address: Optional[Union[Address, str]] = None, **kwargs*) → *BuildTransactionModel*

Builds a transaction and returns the hex to use when executing the transaction.

Parameters

- **fee_amount** (*Money, int, float, Decimal, optional*) – The fee amount. Cannot be set with fee_type.
- **password** (*str*) – The password.
- **segwit_change_address** (*bool, optional*) – If True, the change address is a segwit address. Default=False.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint]*) – A list of outpoints to include in the transaction.
- **recipients** (*List[Recipient]*) – A list of recipients with amounts.
- **op_return_data** (*str, optional*) – The OP_RETURN data.
- **op_return_amount** (*Money, int, float, Decimal, optional*) – The amount to burn in OP_RETURN.
- **fee_type** (*str, optional*) – The fee type. Allowed [low, medium, high]
- **allow_unconfirmed** (*bool, optional*) – If True, includes unconfirmed outputs. Default=False.
- **shuffle_outputs** (*bool, optional*) – If True, shuffle outputs. Default=False.
- **change_address** (*Address, str, optional*) – Specify a change address. If not set, a new change address is used.
- ****kwargs** – Extra keyword arguments.

Returns A built transaction.

Return type `BuildTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

build_interflux_transaction(*wallet_name: str, password: str, destination_chain: int, destination_address: Union[Address, str], outpoints: List[Outpoint], recipients: List[Recipient], fee_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, segwit_change_address: bool = False, op_return_data: Optional[str] = None, op_return_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, fee_type: Optional[str] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, account_name: str = 'account 0', change_address: Optional[Union[Address, str]] = None, **kwargs*) → `BuildTransactionModel`

Builds a transaction and returns the hex to use when executing the transaction.

Parameters

- **destination_chain** (*int*) – Enumeration representing the destination chain.
- **destination_address** (*Address, str*) – The destination address.
- **fee_amount** (*Money, int, float, Decimal, optional*) – The fee amount. Cannot be set with `fee_type`.
- **password** (*str*) – The password.
- **segwit_change_address** (*bool, optional*) – If True, the change address is a segwit address.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint]*) – A list of outpoints to include in the transaction.
- **recipients** (*List[Recipient]*) – A list of recipients with amounts.
- **op_return_data** (*str, optional*) – The OP_RETURN data.
- **op_return_amount** (*Money, int, float, Decimal, optional*) – The amount to burn in OP_RETURN.
- **fee_type** (*str, optional*) – The fee type. Allowed [low, medium, high]
- **allow_unconfirmed** (*bool, optional*) – If True, includes unconfirmed outputs. Default=False.
- **shuffle_outputs** (*bool, optional*) – If True, shuffle outputs. Default=False.
- **change_address** (*Address, str, optional*) – Specify a change address. If not set, a new change address is used.
- ****kwargs** – Extra keyword arguments.

Returns A built interflux transaction.

Return type `BuildTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

send_transaction(*transaction_hex: Union[str, hexstr], **kwargs*) → `WalletSendTransactionModel`
Sends a transaction that has already been built.

Parameters

- **transaction_hex** (*hexstr*, *str*) – The hexified transaction.
- ****kwargs** – Extra keyword arguments.

Returns Information about a sent transaction.

Return type `WalletSendTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

list_wallets(***kwargs*) → dict

Lists all the files found in the database

Parameters ****kwargs** – Extra keyword arguments.

Returns A list of wallets.

Return type *dict*

Raises `APIError` – Error thrown by node API. See message for details.

account(*wallet_name: str, password: str, **kwargs*) → str

Creates a new account for a wallet.

Parameters

- **password** (*str*) – The wallet password.
- **wallet_name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns The newly created account name.

Return type *str*

Raises `APIError` – Error thrown by node API. See message for details.

accounts(*wallet_name: str, **kwargs*) → List[str]

Gets a list of accounts for the specified wallet.

Parameters

- **wallet_name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns A list of accounts.

Return type *List[str]*

Raises `APIError` – Error thrown by node API. See message for details.

unused_address(*wallet_name: str, account_name: str = 'account 0', segwit: bool = False, **kwargs*) → `Address`

Gets an unused address (in the Base58 format) for a wallet account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **segwit** (*bool, optional*) – If True, get a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns An unused address.

Return type `Address`

Raises **APIError** – Error thrown by node API. See message for details.

unused_addresses(*wallet_name: str, count: int, account_name: str = 'account 0', segwit: bool = False, **kwargs*) → List[Address]

Gets a specified number of unused addresses (in the Base58 format) for a wallet account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **count** (*int*) – The number of addresses to get.
- **segwit** (*bool, optional*) – If True, get a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A list of unused addresses.

Return type List[Address]

Raises **APIError** – Error thrown by node API. See message for details.

new_addresses(*wallet_name: str, count: int, account_name: str = 'account 0', segwit: bool = False, **kwargs*) → List[Address]

Gets a specified number of new addresses (in the Base58 format) for a wallet account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **count** (*conint (ge=1)*) – The number of addresses to get.
- **segwit** (*bool, optional*) – If True, get a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A new address.

Return type List[Address]

Raises **APIError** – Error thrown by node API. See message for details.

addresses(*wallet_name: str, account_name: str = 'account 0', segwit: bool = False, **kwargs*) → AddressesModel

Gets all addresses for a wallet account.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **segwit** (*bool, optional*) – If True, gets a segwit address. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns All addresses associated with the account given.

Return type AddressesModel

Raises **APIError** – Error thrown by node API. See message for details.

remove_transactions(*wallet_name*: str, *ids*: Optional[List[Union[str, uint256]]] = None, *from_date*: Optional[str] = None, *remove_all*: bool = False, *resync*: bool = True, ****kwargs**) → List[RemovedTransactionModel]

Removes transactions from the wallet.

Parameters

- **wallet_name** (str) – The wallet name.
- **ids** (List[uint256, str], optional) – A list of transaction ids to remove.
- **from_date** (str, optional) – An option to remove transactions after given date.
- **remove_all** (bool, optional) – An option to remove all transactions. Default=False.
- **resync** (bool, optional) – If True, resyncs wallet after items removed. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns A list of removed transactions.

Return type List[RemovedTransactionModel]

Raises **APIError** – Error thrown by node API. See message for details.

remove_wallet(*wallet_name*: str, ****kwargs**) → None

Remove a wallet

Parameters

- **wallet_name** (str) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises **APIError** – Error thrown by node API. See message for details.

extpubkey(*wallet_name*: str, *account_name*: str = 'account 0', ****kwargs**) → ExtPubKey

Gets the extended public key of a specified wallet account.

Parameters

- **wallet_name** (str) – The wallet name.
- **account_name** (str, optional) – The account name. Default='account 0'.
- ****kwargs** – Extra keyword arguments.

Returns ExtPubKey

Raises **APIError** – Error thrown by node API. See message for details.

private_key(*password*: str, *wallet_name*: str, *address*: Union[Address, str], ****kwargs**) → Key

Gets the private key of a specified wallet address.

Parameters

- **password** (str) – The wallet password.
- **wallet_name** (str) – The wallet name.
- **address** (Address, str) – The address to request a private key for.
- ****kwargs** – Extra keyword arguments.

Returns The private key.

Return type Key

Raises [APIError](#) – Error thrown by node API. See message for details.

sync(*block_hash: Union[uint256, str], **kwargs*) → None

Requests the node resyncs from a block specified by its block hash.

Parameters

- **block_hash** (*uint256, str*) – The hash to start syncing from.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises [APIError](#) – Error thrown by node API. See message for details.

sync_from_date(*wallet_name: str, date: str, all_transactions: bool = True, **kwargs*) → None

Request the node resyncs starting from a given date and time.

Parameters

- **date** (*str*) – The date to sync from in YYYY-MM-DDTHH:MM:SS format.
- **all_transactions** (*bool, optional*) – If True, sync all transactions. Default=True.
- **wallet_name** (*str*) – The wallet name.
- ****kwargs** – Extra keyword arguments.

Returns None

Raises [APIError](#) – Error thrown by node API. See message for details.

wallet_stats(*wallet_name: str, account_name: str = 'account 0', min_confirmations: int = 0, verbose: bool = True, **kwargs*) → [WalletStatsModel](#)

Retrieves information about the wallet.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **min_confirmations** (*int, optional*) – Include transaction less this amount from the chain tip. Default=0.
- **verbose** (*bool, optional*) – If True, give verbose response. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns Wallet statistical information.

Return type [WalletStatsModel](#)

Raises [APIError](#) – Error thrown by node API. See message for details.

split_coins(*wallet_name: str, wallet_password: str, total_amount_to_split: Union[Money, int, float, decimal.Decimal], utxos_count: int, account_name: str = 'account 0', **kwargs*) → [WalletSendTransactionModel](#)

Creates requested amount of UTXOs each of equal value and sends the transaction.

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **wallet_password** (*str*) – The wallet password.

- **total_amount_to_split** (*Money, int, float, Decimal*) – The total amount to split.
- **utxos_count** (*int*) – The number of utxos to create. (Must be greater than 2).
- ****kwargs** – Extra keyword arguments.

Returns Information about the sent transaction.

Return type [WalletSendTransactionModel](#)

Raises [APIError](#) – Error thrown by node API. See message for details.

distribute_utxos(*wallet_name: str, wallet_password: str, utxos_count: int, utxo_per_transaction: int, outpoints: List[Outpoint], account_name: str = 'account 0', use_unique_address_per_utxo: bool = True, reuse_addresses: bool = True, use_change_addresses: bool = False, timestamp_difference_between_transactions: int = 0, min_confirmations: int = 0, dry_run: bool = True, **kwargs*) → [DistributeUtxoModel](#)

Splits and distributes UTXOs across wallet addresses

Parameters

- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str, optional*) – The account name. Default='account 0'.
- **wallet_password** (*str*) – The wallet password.
- **use_unique_address_per_utxo** (*bool, optional*) – If True, uses a unique address for each utxo. Default=True.
- **reuse_addresses** (*bool, optional*) – If True, reuses addresses. Default=True.
- **use_change_addresses** (*bool, optional*) – If True, use change addresses. Default=False.
- **utxos_count** (*int*) – The number of utxos to create.
- **utxo_per_transaction** (*int*) – The number of utxos per transaction.
- **timestamp_difference_between_transactions** (*int, optional*) – The number of seconds between transactions. Default=0.
- **min_confirmations** (*int, optional*) – The minimum number of confirmations to include in transaction. Default=0.
- **outpoints** (*List[Outpoint]*) – A list of outpoints to include in the transaction.
- **dry_run** (*bool, optional*) – If True, simulate transaction. Default=True.
- ****kwargs** – Extra keyword arguments.

Returns Information about the distribute utxo transaction.

Return type [DistributeUtxoModel](#)

Raises [APIError](#) – Error thrown by node API. See message for details.

sweep(*private_keys: List[Union[Key, str]], destination_address: Union[Address, str], broadcast: bool = False, **kwargs*) → [List\[uint256\]](#)

Sweeps a wallet to specified address.

Parameters

- **private_keys** (*List[Key, str]*) – A list of private keys to sweep.

- **destination_address** (*Address*, *str*) – The address to sweep the coins to.
- **broadcast** (*bool*, *optional*) – Broadcast transaction after creation. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A list of transactions for the sweep.

Return type *List[uint256]*

Raises *APIError* – Error thrown by node API. See message for details.

build_offline_sign_request(*wallet_name: str, outpoints: List[Outpoint], recipients: List[Recipient], fee_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, op_return_data: Optional[str] = None, op_return_amount: Optional[Union[Money, int, float, decimal.Decimal]] = None, fee_type: Optional[str] = None, allow_unconfirmed: bool = False, shuffle_outputs: bool = False, account_name: str = 'account 0', change_address: Optional[Union[Address, str]] = None, **kwargs*) → *BuildOfflineSignModel*

Builds an offline sign request for a transaction.

Parameters

- **fee_amount** (*Money*, *int*, *float*, *Decimal*) – The fee amount. Cannot be set with *fee_type*.
- **wallet_name** (*str*) – The wallet name.
- **account_name** (*str*, *optional*) – The account name. Default='account 0'.
- **outpoints** (*List[Outpoint]*) – A list of outputs to use for the transaction.
- **recipients** (*List[Recipient]*) – A list of recipients, including amounts.
- **op_return_data** (*str*, *optional*) – The OP_RETURN data.
- **op_return_amount** (*Money*, *optional*) – The amount to burn in OP_RETURN.
- **fee_type** (*str*, *optional*) – The fee type. Allowed [low, medium, high]
- **allow_unconfirmed** (*bool*, *optional*) – If True, includes unconfirmed outputs. Default=False.
- **shuffle_outputs** (*bool*, *optional*) – If True, shuffle outputs. Default=False.
- **change_address** (*Address*, *optional*) – Specify a change address. If not set, a new change address is used.
- ****kwargs** – Extra keyword arguments.

Returns A built transaction that can be signed offline.

Return type *BuildOfflineSignModel*

Raises *APIError* – Error thrown by node API. See message for details.

offline_sign_request(*wallet_password: str, wallet_name: str, unsigned_transaction: Union[str, hexstr], fee: Union[Money, int, float, decimal.Decimal], utxos: List[UtxoDescriptor], addresses: List[AddressDescriptor], wallet_account: str = 'account 0', **kwargs*) → *BuildTransactionModel*

Build an offline sign request for a transaction. The resulting transaction hex can be broadcast.

Parameters

- **wallet_password** (*str*) – The wallet password.

- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*, *optional*) – The account name. Default='account 0'.
- **unsigned_transaction** (*hexstr*, *str*) – The unsigned transaction hexstr.
- **fee** (*Money*, *int*, *float*, *Decimal*) – The fee.
- **utxos** (*List*[*UtxoDescriptor*]) – A list of utxodescriptors.
- **addresses** (*List*[*AddressDescriptor*]) – A list of addresses to send transactions.
- ****kwargs** – Extra keyword arguments.

Returns A signed transaction that can be broadcast.

Return type `BuildTransactionModel`

Raises `APIError` – Error thrown by node API. See message for details.

consolidate(*wallet_password: str*, *wallet_name: str*, *destination_address: Union[Address, str]*, *utxo_value_threshold_in_satoshis: int*, *wallet_account: str = 'account 0'*, *broadcast: bool = False*, ***kwargs*) → *hexstr*

Consolidate a wallet.

utxo_value_threshold looks to consolidate any utxo amount below the threshold.

Parameters

- **wallet_password** (*str*) – The wallet password.
- **wallet_name** (*str*) – The wallet name.
- **wallet_account** (*str*, *optional*) – The account name. Default='account 0'.
- **destination_address** (*Address*, *str*) – The destination address.
- **utxo_value_threshold_in_satoshis** (*int*) – The threshold where amounts below this amount will be consolidated. (min 1e8)
- **broadcast** (*bool*, *optional*) – If True, broadcast consolidation transaction. Default=False.
- ****kwargs** – Extra keyword arguments.

Returns A consolidation transaction ready for broadcast.

Return type `hexstr`

Raises `APIError` – Error thrown by node API. See message for details.

AccountHistoryModel

```
class AccountHistoryModel(*, accountName: str, accountHdPath: str, coinType: CoinType,
                           transactionsHistory: List[TransactionItemModel])
```

An pydantic model for account history.

account_name: `str`
The account name.

account_hd_path: `str`
The account HD path.

coin_type: `CoinType`
The coin type.

transactions_history: `List[TransactionItemModel]`

A list of transactions composing the history.

AddressBalanceModel

class AddressBalanceModel(**address: Address, coinType: CoinType, amountConfirmed: Money, amountUnconfirmed: Money, spendableAmount: Money*)

A pydantic model for an address balance.

address: `Address`

The address.

coin_type: `CoinType`

The coin type.

amount_confirmed: `Money`

The confirmed amount.

amount_unconfirmed: `Money`

The unconfirmed amount.

spendable_amount: `Money`

The spendable amount.

AddressesModel

class AddressesModel(**addresses: List[AddressModel]*)

A pydantic model for a list of addressmodels.

addresses: `List[AddressModel]`

The list of address models.

BuildOfflineSignModel

class BuildOfflineSignModel(**walletName: str, walletAccount: str, unsignedTransaction: hexstr, fee: Money, utxos: List[UtxoDescriptor], addresses: List[AddressDescriptor]*)

A pydantic model for a built offline sign request.

wallet_name: `str`

The wallet name.

wallet_account: `str`

The wallet account.

unsigned_transaction: `hexstr`

The unsigned transaction hex.

fee: `Money`

The transaction fee.

utxos: `List[UtxoDescriptor]`

The utxos included in the transaction.

addresses: `List[AddressDescriptor]`

The addresses and amounts receiving outputs.

BuildTransactionModel

```
class BuildTransactionModel(*, fee: Money = 0, hex: hexstr, transactionId: uint256)
```

A pydantic model for a built transaction.

fee: *Money*

The transaction fee.

hex: *hexstr*

The transaction hex.

transaction_id: *uint256*

The transaction hash.

DistributeUtxoModel

```
class DistributeUtxoModel(*, walletName: str, useUniqueAddressPerUtxo: bool, utxosCount: int,
                           utxoPerTransaction: int, timestampDifferenceBetweenTransactions: int,
                           minConfirmations: int, dryRun: bool, walletSendTransaction:
                           List[WalletSendTransactionModel] = None)
```

A pydantic model for the distribute utxo method.

wallet_name: *str*

The wallet name.

use_unique_address_per_utxo: *bool*

If true, a different address used for each utxo.

utxos_count: *int*

The number of utxos.

utxo_per_transaction: *int*

The number of utxos per transaction.

timestamp_difference_between_transactions: *int*

The number of seconds between transactions.

min_confirmations: *int*

The minimum number of confirmations to include utxo in transaction.

dry_run: *bool*

If true, simulate the transaction.

wallet_send_transaction: *Optional[List[WalletSendTransactionModel]]*

A list of send transactions (if not simulated).

MaxSpendableAmountModel

```
class MaxSpendableAmountModel(*, maxSpendableAmount: Money, Fee: Money)
```

A pydantic model for the maximum spendable amount.

max_spendable_amount: *Money*

The maximum spendable amount.

fee: *Money*

The to spend the maximum amount.

PaymentDetailModel

class PaymentDetailModel(**, destinationAddress: Address, amount: Money, isChange: bool*)

A pydantic model for payment details.

destination_address: Address

The destination address.

amount: Money

The amount sent to this address.

is_change: bool

If true, destination address is a change address.

RemovedTransactionModel

class RemovedTransactionModel(**, transactionId: uint256, creationTime: datetime.datetime*)

A pydantic model for a removed transaction.

transaction_id: uint256

The removed transaction hash.

creation_time: datetime.datetime

The creation time of the removed transaction.

SpendableTransactionModel

class SpendableTransactionModel(**, id: uint256, index: int, address: Address, isChange: bool, amount: Money, creationTime: datetime.datetime, confirmations: int*)

A pydantic model representing spendable transactions.

transaction_id: uint256

The transaction hash with spendable output.

index: int

The index of the spendable output.

address: Address

The address holding the spendable output.

is_change: bool

If true, address is a change address.

amount: Money

The amount in the unspent output.

creation_time: datetime.datetime

The output creation time.

confirmations: int

The number of confirmations for three output.

SpendableTransactionsModel

class SpendableTransactionsModel(* , transactions: List[SpendableTransactionModel])

A pydantic model for a list of spendable transactions.

transactions: List[SpendableTransactionModel]

A list of spendable transactions.

TransactionItemModel

class TransactionItemModel(* , type: TransactionItemType, toAddress: Address, id: uint256, amount: Money, payments: List[PaymentDetailModel] = None, fee: Money = None, confirmedInBlock: int, timestamp: datetime.datetime, txOutputTime: datetime.datetime, txOutputIndex: int, blockIndex: int = None)

A pydantic model for a transaction item.

transaction_type: TransactionItemType

The transaction type.

to_address: Address

The address receiving the transaction.

transaction_id: uint256

The transaction hash.

amount: Money

The transaction value.

payments: Optional[List[PaymentDetailModel]]

A list of payment detail models.

fee: Optional[Money]

The transaction fee.

confirmed_in_block: int

The block n height where transaction was confirmed.

timestamp: datetime.datetime

The transaction timestamp.

tx_output_time: datetime.datetime

The transaction output time.

tx_output_index: int

The transaction output index.

block_index: Optional[int]

The block index.

TransactionOutputModel

```
class TransactionOutputModel(*, address: Optional[Union[int, Address]] = None, amount: Money,
                             opReturnData: str = None)
```

A pydantic model of a transaction output.

```
address: Optional[Union[int, Address]]
```

The address receiving the output.

```
amount: Money
```

The output amount.

```
op_return_data: Optional[str]
```

The OP_RETURN data, if present.

UtxoAmountModel

```
class UtxoAmountModel(*, Amount: Money, Count: int)
```

A pydantic model representing a utxo amount.

```
amount: Money
```

The total amount in the utxos.

```
count: int
```

The number of utxos included in the count.

UtxoPerBlockModel

```
class UtxoPerBlockModel(*, utxoPerBlock: int, Count: int)
```

A pydantic model representing utxo per block.

```
utxo_per_block: int
```

The number of utxo per block.

```
count: int
```

The number of utxos.

UtxoPerTransactionModel

```
class UtxoPerTransactionModel(*, utxoPerTransaction: int, Count: int)
```

A pydantic model for utxo per transaction.

```
utxo_per_transaction: int
```

The utxo per transaction.

```
count: int
```

The total number of utxos.

WalletHistoryModel

class WalletHistoryModel(* , *History: List[AccountHistoryModel]*)

A pydantic model for a wallet history.

history: List[AccountHistoryModel]

A list of account histories.

WalletStatsModel

class WalletStatsModel(* , *walletName: str, totalUtxoCount: int, uniqueTransactionCount: int, uniqueBlockCount: int, countOfTransactionsWithAtLeastMaxReorgConfirmations: int, utxoAmounts: List[UtxoAmountModel], utxoPerTransaction: List[UtxoPerTransactionModel], utxoPerBlock: List[UtxoPerBlockModel]*)

A pydantic model for wallet stats.

wallet_name: str

The wallet name.

total_utxo_count: int

The total number of utxos.

unique_transaction_count: int

The number of unique transactions.

unique_block_count: int

The number of unique blocks containing wallet transactions.

finalized_transactions: int

The number of finalized transactions.

utxo_amounts: List[UtxoAmountModel]

A list of utxo amounts.

utxo_per_transaction: List[UtxoPerTransactionModel]

A list of utxo per transaction.

utxo_per_block: List[UtxoPerBlockModel]

A list of utxo per block.

WalletBalanceModel

class WalletBalanceModel(* , *balances: List[AccountBalanceModel]*)

A pydantic model for a wallet balance.

balances: List[AccountBalanceModel]

A list of account balances.

WalletGeneralInfoModel

```
class WalletGeneralInfoModel(*, walletName: str = None, network: str, creationTime: datetime.datetime,
                             isDecrypted: bool, lastBlockSyncedHeight: int, chainTip: int, isChainSynced:
                             bool, connectedNodes: int)
```

A model representing general wallet info.

wallet_name: Optional[str]

The name of the wallet. Will be None for multisig.

network: str

The name of the network the wallet is operating on.

creation_time: datetime.datetime

The datetime of wallet creation

is_decrypted: bool

If true, wallet is decrypted.

last_block_synced_height: int

The height of last block synced by wallet.

chain_tip: int

The height off the chain tip.

is_chain_synced: bool

If true, chain is synced.

connected_nodes: int

The number of connected nodes.

WalletSendTransactionModel

```
class WalletSendTransactionModel(*, transactionId: uint256, outputs: List[TransactionOutputModel])
```

A pydantic model for a send transaction response.

transaction_id: uint256

The transaction hash.

outputs: List[TransactionOutputModel]

A list of transaction outputs.

1.1.1.28 Global_ResponseModels

AccountBalanceModel

```
class AccountBalanceModel(*, accountName: str = None, accountHdPath: str = None, coinType: CoinType,
                           amountConfirmed: Money, amountUnconfirmed: Money, spendableAmount:
                           Money = None, addresses: List[AddressModel] = None)
```

A pydantic model for account balance.

account_name: Optional[str]

The account name. Will be None for multisig.

account_hd_path: Optional[str]

The account HD path. Will be None for multisig.

coin_type: *CoinType*
The coin type.

amount_confirmed: *Money*
The amount confirmed.

amount_unconfirmed: *Money*
The amount unconfirmed.

spendable_amount: *Optional[Money]*
The spendable amount. Will be None for multisig.

addresses: *Optional[List[AddressModel]]*
A list of addresses.

AddressBalanceModel

class AddressBalanceModel(**address: Address, coinType: CoinType, amountConfirmed: Money, amountUnconfirmed: Money, spendableAmount: Money*)

A pydantic model for an address balance.

address: *Address*
The address.

coin_type: *CoinType*
The coin type.

amount_confirmed: *Money*
The confirmed amount.

amount_unconfirmed: *Money*
The unconfirmed amount.

spendable_amount: *Money*
The spendable amount.

AddressDescriptor

class AddressDescriptor(**address: Address, keyPath: str, addressType: str*)

A pydantic model of an address descriptor.

address: *Address*
The address.

key_path: *str*
The key path.

address_type: *str*
The address type.

AddressesModel

```
class AddressesModel(*, addresses: List[AddressModel])
```

A pydantic model for a list of addressmodels.

```
addresses: List[AddressModel]
```

The list of address models.

AddressModel

```
class AddressModel(*, address: Address, isUsed: bool, isChange: bool, amountConfirmed: Money,
                  amountUnconfirmed: Money)
```

A pydantic model representing an address with balance.

```
address: Address
```

The address.

```
is_used: bool
```

If true, the address is used.

```
is_change: bool
```

If true, the address is a change address.

```
amount_confirmed: Money
```

The amount confirmed in the address.

```
amount_unconfirmed: Money
```

The amount unconfirmed in the address.

BlockModel

```
class BlockModel(*, hash: uint256, confirmations: int, size: int, weight: int, height: int, version: int, versionHex:
                 str, merkleroot: hexstr, tx: List[uint256] = None, time: datetime.datetime, mediantime:
                 datetime.datetime, nonce: int, bits: str, difficulty: float, chainwork: str, nTx: int,
                 previousblockhash: uint256 = None, nextblockhash: uint256 = None, signature: str = None,
                 modifyerv2: str = None, flags: str = None, hashproof: str = None, blocktrust: str = None,
                 chaintrust: str = None)
```

A pydantic model of a block.

```
hash: uint256
```

The block hash.

```
confirmations: int
```

The number of confirmations.

```
size: int
```

The size of the block.

```
weight: int
```

The weight of the block.

```
height: int
```

The height of the block.

```
version: int
```

The block version.

version_hex: str
The block version in hex.

merkleroot: hexstr
The block merkleroot.

tx: Optional[List[uint256]]
A list of transactions in the block.

time: datetime.datetime
The time the block was produced.

median_time: datetime.datetime
The median time.

nonce: int
The block's nonce.

bits: str
The block bits.

difficulty: float
The block difficulty.

chainwork: str
The chain work.

n_tx: int
The number of transactions in the block.

previous_blockhash: Optional[uint256]
The previous block hash.

next_blockhash: Optional[uint256]
The next block hash.

signature: Optional[str]
The signature.

modifier_v2: Optional[str]
The block modifier.

flags: Optional[str]
Block flags.

hashproof: Optional[str]
Block hashproof.

blocktrust: Optional[str]
Blocktrust.

chaintrust: Optional[str]
Chaintrust.

BlockTransactionDetailsModel

```
class BlockTransactionDetailsModel(*, hash: uint256, confirmations: int, size: int, weight: int, height: int,
    version: int, versionHex: str, merkleroot: hexstr, tx: List[uint256] =
    None, time: datetime.datetime, mediantime: datetime.datetime, nonce:
    int, bits: str, difficulty: float, chainwork: str, nTx: int,
    previousblockhash: uint256 = None, nextblockhash: uint256 = None,
    signature: str = None, modifyrv2: str = None, flags: str = None,
    hashproof: str = None, blocktrust: str = None, chaintrust: str = None,
    Transactions: List[TransactionModel])
```

A pydantic model for block transaction details.

transactions: List[TransactionModel]

A list of transactions.

hash: uint256

The block hash.

confirmations: int

The number of confirmations.

size: int

The size of the block.

weight: int

The weight of the block.

height: int

The height of the block.

version: int

The block version.

version_hex: str

The block version in hex.

merkleroot: hexstr

The block merkleroot.

tx: Optional[List[uint256]]

A list of transactions in the block.

time: datetime

The time the block was produced.

median_time: datetime

The median time.

nonce: int

The block's nonce.

bits: str

The block bits.

difficulty: float

The block difficulty.

chainwork: str

The chain work.

n_tx: int

The number of transactions in the block.

previous_blockhash: `Optional[uint256]`

The previous block hash.

next_blockhash: `Optional[uint256]`

The next block hash.

signature: `Optional[str]`

The signature.

modifier_v2: `Optional[str]`

The block modifier.

flags: `Optional[str]`

Block flags.

hashproof: `Optional[str]`

Block hashproof.

blocktrust: `Optional[str]`

Blocktrust.

chaintrust: `Optional[str]`

Chaintrust.

BuildContractTransactionModel

```
class BuildContractTransactionModel(*, fee: Money, hex: hexstr, message: str = None, success: bool = None, transactionId: uint256 = None)
```

A pydantic model for building a smart contract transaction.

fee: `Money`

The transaction fee.

hex: `hexstr`

The hex serialized transaction.

message: `Optional[str]`

The build transaction message.

success: `Optional[bool]`

True if build was successful.

transaction_id: `Optional[uint256]`

The transaction hash, if build successful.

BuildCreateContractTransactionModel

```
class BuildCreateContractTransactionModel(*, fee: Money, hex: hexstr, message: str = None, success: bool = None, transactionId: uint256 = None, newContractAddress: Address)
```

A pydantic model for a create smart contract transaction.

new_contract_address: `Address`

The new address associated with the smart contract.

fee: `Money`

The transaction fee.

hex: `hexstr`
The hex serialized transaction.

message: `Optional[str]`
The build transaction message.

success: `Optional[bool]`
True if build was successful.

transaction_id: `Optional[uint256]`
The transaction hash, if build successful.

BuildOfflineSignModel

```
class BuildOfflineSignModel(*, walletName: str, walletAccount: str, unsignedTransaction: hexstr, fee: Money, utxos: List[UtxoDescriptor], addresses: List[AddressDescriptor])
```

A pydantic model for a built offline sign request.

wallet_name: `str`
The wallet name.

wallet_account: `str`
The wallet account.

unsigned_transaction: `hexstr`
The unsigned transaction hex.

fee: `Money`
The transaction fee.

utxos: `List[UtxoDescriptor]`
The utxos included in the transaction.

addresses: `List[AddressDescriptor]`
The addresses and amounts receiving outputs.

BuildTransactionModel

```
class BuildTransactionModel(*, fee: Money = 0, hex: hexstr, transactionId: uint256)
```

A pydantic model for a built transaction.

fee: `Money`
The transaction fee.

hex: `hexstr`
The transaction hex.

transaction_id: `uint256`
The transaction hash.

MaturedBlockInfoModel

class MaturedBlockInfoModel(* , *blockHash: uint256, blockHeight: int, blockTime: datetime.datetime*)

A pydantic model representing a married block.

block_hash: uint256

The block hash.

block_height: int

The block height.

block_time: datetime.datetime

The time block was produced.

PollViewModel

class PollViewModel(* , *IsPending: bool, IsExecuted: bool, Id: int, PollVotedInFavorBlockDataHash: uint256 = None, PollVotedInFavorBlockDataHeight: int = None, PollStartFavorBlockDataHash: uint256 = None, PollStartFavorBlockDataHeight: int = None, PollExecutedBlockDataHash: uint256 = None, PollExecutedBlockDataHeight: int = None, PubKeysHexVotedInFavor: List[PubKey], VotingDataString: str*)

A pydantic model for polling data.

is_pending: bool

If true, poll is pending.

is_executed: bool

If true, poll has been executed.

poll_id: int

The poll id.

poll_voted_in_favor_blockdata_hash: Optional[uint256]

If voted in favor, the block of the vote.

poll_voted_in_favor_blockdata_height: Optional[int]

If voted in favor, the height of the block.

poll_start_favor_blockdata_hash: Optional[uint256]

The block hash when polling started.

poll_start_favor_blockdata_height: Optional[int]

The block height when polling started.

poll_executed_blockdata_hash: Optional[uint256]

The block hash when poll was executed, if executed.

poll_executed_blockdata_height: Optional[int]

The block height when poll was executed, if executed.

pubkeys_hex_voted_in_favor: List[PubKey]

A list of pubkeys voting in favor of poll.

voting_data_string: str

Voting data.

RemovedTransactionModel

class RemovedTransactionModel(**, transactionId: uint256, creationTime: datetime.datetime*)

A pydantic model for a removed transaction.

transaction_id: *uint256*

The removed transaction hash.

creation_time: *datetime.datetime*

The creation time of the removed transaction.

ScriptPubKey

class ScriptPubKey(**, asm: str, hex: str, type: str, reqSigs: int = None, addresses: List[str] = None*)

A ScriptPubKey.

A ScriptPubKey is a part of transaction's output, and is the second half of a script.

Note: Learn more about [transaction structure](#).

type: *str*

The type of script. The list of supported types can be found in [sources](#).

req_sigs: *Optional[int]*

The number of required signatures.

asm: *str*

The assembly representation of the script.

hex: *str*

The hex representation of the script.

addresses: *Optional[List[str]]*

A list of output addresses.

ScriptSig

class ScriptSig(**, asm: str, hex: str*)

Represents ScriptSig.

A ScriptSig is a part of transaction's input, and is the first half of a script.

Note: Learn more about [transaction structure](#).

asm: *str*

The assembly representation of the script.

hex: *str*

The hex representation of the script.

TransactionModel

```
class TransactionModel(*, hex: hexstr, txid: uint256, hash: uint256, version: int, size: int, vsize: int, weight: int, locktime: int, vin: List[VIn], vout: List[VOut], blockhash: uint256 = None, confirmations: int = None, time: datetime.datetime = None, blocktime: datetime.datetime = None)
```

A pydantic model for a transaction.

hex: *hexstr*

The transaction hex.

txid: *uint256*

The transaction hash.

hash: *uint256*

The transaction hash.

version: *int*

The transaction version.

size: *int*

The transaction size.

vsize: *int*

The transaction vsize.

weight: *int*

The transaction weight.

locktime: *int*

The transaction locktime.

vin: *List[VIn]*

A list of VIn.

vout: *List[VOut]*

A list of VOut.

blockhash: *Optional[uint256]*

The hash of the block containing the transaction.

confirmations: *Optional[int]*

The number of confirmations of the transaction.

time: *Optional[datetime.datetime]*

The transaction time.

blocktime: *Optional[datetime.datetime]*

The blocktime.

TransactionOutputModel

```
class TransactionOutputModel(*, address: Optional[Union[int, Address]] = None, amount: Money, opReturnData: str = None)
```

A pydantic model of a transaction output.

address: *Optional[Union[int, Address]]*

The address receiving the output.

amount: *Money*

The output amount.

op_return_data: Optional[str]
The OP_RETURN data, if present.

UtxoDescriptor

class UtxoDescriptor(*, *transactionId: uint256, index: int, scriptPubKey: str, amount: Money*)

A pydantic model of a utxo descriptor.

transaction_id: uint256
The transaction hash off the utxo.

index: int
The index of the utxo in the transaction.

script_pubkey: str
The scriptpubkey of the utxo.

amount: Money
The amount in the utxo.

VIn

class VIn(*, *coinbase: str = None, txid: str = None, vout: int = None, scriptSig: ScriptSig = None, sequence: int*)
Represents transaction's input.

Note: Learn more about [transaction input structure](#).

coinbase: Optional[str]
Three scriptSig off this was a coinbase transaction.

txid: Optional[str]
The transaction hash.

vout: Optional[int]
The index of the output.

script_sig: Optional[ScriptSig]
The scriptSig.

sequence: int
The transaction's sequence number.

VOut

class VOut(*, *value: Money, n: int, scriptPubKey: ScriptPubKey*)
Represents transaction's output.

Note: Learn more about [transaction output structure and scriptPubKey](#).

value: Money
The value of a transaction's output.

n: int
The index of the output.

script_pubkey: ScriptPubKey
The output's scriptPubKey.

WalletBalanceModel

class WalletBalanceModel(**, balances: List[AccountBalanceModel]*)
A pydantic model for a wallet balance.

balances: List[AccountBalanceModel]
A list of account balances.

WalletGeneralInfoModel

class WalletGeneralInfoModel(**, walletName: str = None, network: str, creationTime: datetime.datetime, isDecrypted: bool, lastBlockSyncedHeight: int, chainTip: int, isChainSynced: bool, connectedNodes: int*)

A model representing general wallet info.

wallet_name: Optional[str]
The name of the wallet. Will be None for multisig.

network: str
The name of the network the wallet is operating on.

creation_time: datetime.datetime
The datetime of wallet creation

is_decrypted: bool
If true, wallet is decrypted.

last_block_synced_height: int
The height of last block synced by wallet.

chain_tip: int
The height off the chain tip.

is_chain_synced: bool
If true, chain is synced.

connected_nodes: int
The number of connected nodes.

WalletSendTransactionModel

class WalletSendTransactionModel(**, transactionId: uint256, outputs: List[TransactionOutputModel]*)
A pydantic model for a send transaction response.

transaction_id: uint256
The transaction hash.

outputs: List[TransactionOutputModel]
A list of transaction outputs.

1.1.2 FeatureInitializationState

```
class FeatureInitializationState(value)
    Enum representing current state of feature initialization.
    Uninitialized = 'Uninitialized'
    Initializing = 'Initializing'
    Initialized = 'Initialized'
    Disposing = 'Disposing'
    Disposed = 'Disposed'
```

1.1.3 FullNodeState

```
class FullNodeState(value)
    Enum representing current state of Full Node.
    Created = 'Created'
    Initializing = 'Initializing'
    Initialized = 'Initialized'
    Starting = 'Starting'
    Started = 'Started'
    Disposing = 'Disposing'
    Disposed = 'Disposed'
    Failed = 'Failed'
```

1.1.4 LogRule

```
class LogRule(*, ruleName: str, logLevel: str, filename: str)
    A log rule model.
```

Note: More information about logging in Stratis Full Node can be found [here](#).

```
rule_name: str
    The name of the log rule.
log_level: str
    The log level.
filename: str
    The log file name.
```

1.1.5 APIError

exception `APIError`(*code: int, message: str*)

Represents an API error message response.

code

The API error code.

Type *int*

message

The error message.

Type *str*

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

1.2 Nodes

1.2.1 StraxNode

class `StraxNode`(*ipaddress: str = http://localhost, network: Union[StraxMain, StraxTest, StraxRegTest] = StraxMain()*)

A Strax Node.

property `coldstaking`: *ColdStaking*

The coldstaking route.

Returns A ColdStaking instance.

Return type *ColdStaking*

property `diagnostic`: *Diagnostic*

The diagnostic route.

Returns A Diagnostic instance.

Return type *Diagnostic*

property `externalapi`: `pystratis.api.externalapi.externalapi.ExternalAPI`

The externalapi route.

Returns A ExternalAPI instance.

Return type *ExternalAPI*

property `mining`: *Mining*

The mining route.

Returns A Mining instance.

Return type *Mining*

property `signalr`: *SignalR*

The signalr route.

Returns A SignalR instance.

Return type *SignalR*

property staking: *Staking*

The staking route.

Returns A Staking instance.

Return type *Staking*

property addressbook: *AddressBook*

The addressbook route.

Returns An addressbook instance.

Return type *AddressBook*

property blockchainnetwork: `pystratis.core.networks.basenetwork.BaseNetwork`

The node's network type.

Returns The node's network.

Return type *BaseNetwork*

property blockstore: *BlockStore*

The blockstore route.

Returns A BlockStore instance.

Return type *BlockStore*

check_all_endpoints_implemented() → bool

Queries a running node's swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type *bool*

property connection_manager: *ConnectionManager*

The connectionmanager route.

Returns A ConnectionManager instance.

Return type *ConnectionManager*

property consensus: *Consensus*

The consensus route.

Returns A Consensus instance.

Return type *Consensus*

property dashboard: *Dashboard*

The dashboard route.

Returns A Dashboard instance.

Return type *Dashboard*

property ipaddr: `str`

The node's ip address.

Returns The specified ip address of the node.

Return type *str*

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type `Mempool`

property name: `str`

The node's name.

Returns The node name.

Return type `str`

property network: `Network`

The network route.

Returns A Network instance.

Return type `Network`

property node: `Node`

The node route.

Returns A Node instance.

Return type `Node`

property rpc: `RPC`

The RPC route.

Returns A RPC instance.

Return type `RPC`

stop_node() → `bool`

Convenience method for stopping node.

property wallet: `Wallet`

The wallet route.

Returns A Wallet instance.

Return type `Wallet`

1.2.2 CirrusNode

```
class CirrusNode(ipaddress: str = http://localhost, network: Union[CirrusMain, CirrusTest, CirrusRegTest] =  
                CirrusMain())
```

A Cirrus Node.

property balances: `Balances`

The balances route.

Returns A Balances instance.

Return type `Balances`

property collateral: `Collateral`

The collateral route.

Returns A Collateral instance.

Return type `Collateral`

property contract_swagger:

```
pystratis.api.contract_swagger.contract_swagger.ContractSwagger
```

The contract_swagger route.

Returns A ContractSwagger instance.

Return type `ContractSwagger`

property diagnostic: *Diagnostic*

The diagnostic route.

Returns A `Diagnostic` instance.

Return type `Diagnostic`

property dynamic_contract:

`pystratis.api.dynamic_contract.dynamic_contract.DynamicContract`

The dynamic contract route.

Returns A `DynamicContract` instance.

Return type `DynamicContract`

property federation: *Federation*

The federation route.

Returns A `Federation` instance.

Return type `Federation`

property smart_contracts: *SmartContracts*

The smartcontracts route.

Returns A `SmartContracts` instance.

Return type `SmartContracts`

property smart_contract_wallet: *SmartContractWallet*

The smartcontractwallet route.

Returns A `SmartContractWallet` instance.

Return type `SmartContractWallet`

property signalr: *SignalR*

The signalr route.

Returns A `SignalR` instance.

Return type `SignalR`

property voting: *Voting*

The voting route.

Returns A `Voting` instance.

Return type `Voting`

property addressbook: *AddressBook*

The addressbook route.

Returns An `addressbook` instance.

Return type `AddressBook`

property blockchainnetwork: `pystratis.core.networks.basenetwork.BaseNetwork`

The node's network type.

Returns The node's network.

Return type `BaseNetwork`

property blockstore: *BlockStore*

The blockstore route.

Returns A BlockStore instance.

Return type BlockStore

check_all_endpoints_implemented() → bool

Queries a running node's swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type bool

property connection_manager: *ConnectionManager*

The connectionmanager route.

Returns A ConnectionManager instance.

Return type ConnectionManager

property consensus: *Consensus*

The consensus route.

Returns A Consensus instance.

Return type Consensus

property dashboard: *Dashboard*

The dashboard route.

Returns A Dashboard instance.

Return type Dashboard

property ipaddr: str

The node's ip address.

Returns The specified ip address of the node.

Return type str

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type Mempool

property name: str

The node's name.

Returns The node name.

Return type str

property network: *Network*

The network route.

Returns A Network instance.

Return type Network

property node: *Node*

The node route.

Returns A Node instance.

Return type Node

property rpc: *RPC*

The RPC route.

Returns A RPC instance.

Return type *RPC*

stop_node() → bool

Convenience method for stopping node.

property wallet: *Wallet*

The wallet route.

Returns A Wallet instance.

Return type *Wallet*

1.2.3 InterfluxStraxNode

class InterfluxStraxNode(*ipaddress: str = http://localhost, network: Union[StraxMain, StraxTest, StraxRegTest] = StraxMain()*)

A Interflux Strax Node.

property collateral: *Collateral*

The collateral route.

Returns A Collateral instance.

Return type *Collateral*

property collateral_voting: *CollateralVoting*

The collateralvoting route.

Returns A CollateralVoting instance.

Return type *CollateralVoting*

property federation_gateway: *FederationGateway*

The federationgateway route.

Returns A FederationGateway instance.

Return type *FederationGateway*

property federation_wallet: *FederationWallet*

The federationwallet route.

Returns A FederationWallet instance.

Return type *FederationWallet*

property mining: *Mining*

The mining route.

Returns A Mining instance.

Return type *Mining*

property multisig: *Multisig*

The multisig route.

Returns A Multisig instance.

Return type *Multisig*

property notifications: *Notifications*

The notifications route.

Returns A Notifications instance.

Return type *Notifications*

property staking: *Staking*

The staking route.”

Returns A Staking instance.

Return type *Staking*

property addressbook: *AddressBook*

The addressbook route.

Returns An addressbook instance.

Return type *AddressBook*

property blockchainnetwork: *pystratis.core.networks.basenetwork.BaseNetwork*

The node’s network type.

Returns The node’s network.

Return type *BaseNetwork*

property blockstore: *BlockStore*

The blockstore route.

Returns A BlockStore instance.

Return type *BlockStore*

check_all_endpoints_implemented() → bool

Queries a running node’s swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type *bool*

property connection_manager: *ConnectionManager*

The connectionmanager route.

Returns A ConnectionManager instance.

Return type *ConnectionManager*

property consensus: *Consensus*

The consensus route.

Returns A Consensus instance.

Return type *Consensus*

property dashboard: *Dashboard*

The dashboard route.

Returns A Dashboard instance.

Return type *Dashboard*

property ipaddr: *str*

The node’s ip address.

Returns The specified ip address of the node.

Return type *str*

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type *Mempool*

property name: *str*

The node's name.

Returns The node name.

Return type *str*

property network: *Network*

The network route.

Returns A Network instance.

Return type *Network*

property node: *Node*

The node route.

Returns A Node instance.

Return type *Node*

property rpc: *RPC*

The RPC route.

Returns A RPC instance.

Return type *RPC*

stop_node() → bool

Convenience method for stopping node.

property wallet: *Wallet*

The wallet route.

Returns A Wallet instance.

Return type *Wallet*

1.2.4 InterfluxCirrusNode

```
class InterfluxCirrusNode(ipaddress: str = http://localhost, network: Union[CirrusMain, CirrusTest,
    CirrusRegTest] = CirrusMain())
```

An Interflux Cirrus node.

property balances: *Balances*

The balances route.

Returns A Balances instance.

Return type *Balances*

property collateral: *Collateral*

The collateral route.

Returns A Collateral instance.

Return type *Collateral*

property collateral_voting: *CollateralVoting*

The collateralvoting route.

Returns A CollateralVoting instance.

Return type *CollateralVoting*

property contract_swagger:

`pystratis.api.contract_swagger.contract_swagger.ContractSwagger`

The contract_swagger route.

Returns A ContractSwagger instance.

Return type *ContractSwagger*

property dynamic_contract:

`pystratis.api.dynamic_contract.dynamic_contract.DynamicContract`

The dynamic contract route.

Returns A DynamicContract instance.

Return type *DynamicContract*

property externalapi: `pystratis.api.externalapi.externalapi.ExternalAPI`

The externalapi route.

Returns A ExternalAPI instance.

Return type *ExternalAPI*

property federation: *Federation*

The federation route.

Returns A Federation instance.

Return type *Federation*

property federation_gateway: *FederationGateway*

The federationgateway route.

Returns A FederationGateway instance.

Return type *FederationGateway*

property federation_wallet: *FederationWallet*

The federationwallet route.

Returns A FederationWallet instance.

Return type *FederationWallet*

property interop: *Interop*

The interop route.

Returns An Interop instance.

Return type *Interop*

property multisig: *Multisig*

The multisig route.

Returns A Multisig instance.

Return type *Multisig*

property smart_contracts: *SmartContracts*

The smartcontracts route.

Returns A SmartContracts instance.

Return type `SmartContracts`

property smart_contract_wallet: `SmartContractWallet`

The smartcontractwallet route.

Returns A SmartContractWallet instance.

Return type `SmartContractWallet`

property notifications: `Notifications`

The notifications route.

Returns A Notifications instance.

Return type `Notifications`

property voting: `Voting`

The voting route.

Returns A Voting instance.

Return type `Voting`

property addressbook: `AddressBook`

The addressbook route.

Returns An addressbook instance.

Return type `AddressBook`

property blockchainnetwork: `pystratis.core.networks.basenetwork.BaseNetwork`

The node's network type.

Returns The node's network.

Return type `BaseNetwork`

property blockstore: `BlockStore`

The blockstore route.

Returns A BlockStore instance.

Return type `BlockStore`

check_all_endpoints_implemented() → bool

Queries a running node's swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type `bool`

property connection_manager: `ConnectionManager`

The connectionmanager route.

Returns A ConnectionManager instance.

Return type `ConnectionManager`

property consensus: `Consensus`

The consensus route.

Returns A Consensus instance.

Return type `Consensus`

property dashboard: *Dashboard*

The dashboard route.

Returns A Dashboard instance.

Return type *Dashboard*

property ipaddr: *str*

The node's ip address.

Returns The specified ip address of the node.

Return type *str*

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type *Mempool*

property name: *str*

The node's name.

Returns The node name.

Return type *str*

property network: *Network*

The network route.

Returns A Network instance.

Return type *Network*

property node: *Node*

The node route.

Returns A Node instance.

Return type *Node*

property rpc: *RPC*

The RPC route.

Returns A RPC instance.

Return type *RPC*

stop_node() → bool

Convenience method for stopping node.

property wallet: *Wallet*

The wallet route.

Returns A Wallet instance.

Return type *Wallet*

1.2.5 StraxMasterNode

class StraxMasterNode(*ipaddress: str = http://localhost, network: Union[StraxMain, StraxTest, StraxRegTest] = StraxMain()*)

The Strax member of the masternode pair.

property addressbook: *AddressBook*

The addressbook route.

Returns An addressbook instance.

Return type *AddressBook*

property balances: *Balances*

The balances route.

Returns A Balances instance

Return type *Balances*

property blockchainnetwork: *pystratis.core.networks.basenetwork.BaseNetwork*

The node's network type.

Returns The node's network.

Return type *BaseNetwork*

property blockstore: *BlockStore*

The blockstore route.

Returns A BlockStore instance.

Return type *BlockStore*

check_all_endpoints_implemented() → *bool*

Queries a running node's swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type *bool*

property collateral: *Collateral*

The collateral route. Not available in devmode.

Returns A Collateral instance.

Return type *Collateral*

property connection_manager: *ConnectionManager*

The connectionmanager route.

Returns A ConnectionManager instance.

Return type *ConnectionManager*

property consensus: *Consensus*

The consensus route.

Returns A Consensus instance.

Return type *Consensus*

property contract_swagger:

pystratis.api.contract_swagger.contract_swagger.ContractSwagger

The contract_swagger route.

Returns A ContractSwagger instance.

Return type ContractSwagger

property dashboard: *Dashboard*

The dashboard route.

Returns A Dashboard instance.

Return type Dashboard

property dynamic_contract:

`pystratis.api.dynamic_contract.dynamic_contract.DynamicContract`

The dynamic contract route.

Returns A DynamicContract instance.

Return type DynamicContract

property federation: *Federation*

The federation route.

Returns A Federation instance.

Return type Federation

property ipaddr: `str`

The node's ip address.

Returns The specified ip address of the node.

Return type `str`

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type Mempool

property name: `str`

The node's name.

Returns The node name.

Return type `str`

property network: *Network*

The network route.

Returns A Network instance.

Return type Network

property node: *Node*

The node route.

Returns A Node instance.

Return type Node

property notifications: *Notifications*

The notifications route.

Returns A Notifications instance.

Return type Notifications

property rpc: *RPC*

The RPC route.

Returns A RPC instance.

Return type *RPC*

property signalr: *SignalR*

The signalr route.

Returns A SignalR instance.

Return type *SignalR*

property smart_contract_wallet: *SmartContractWallet*

The smartcontractwallet route.

Returns A SmartContractWallet instance.

Return type *SmartContractWallet*

property smart_contracts: *SmartContracts*

The smartcontracts route.

Returns A SmartContracts instance.

Return type *SmartContracts*

stop_node() → bool

Convenience method for stopping node.

property voting: *Voting*

The voting route.

Returns A Voting instance.

Return type *Voting*

property wallet: *Wallet*

The wallet route.

Returns A Wallet instance.

Return type *Wallet*

1.2.6 CirrusMasternode

class CirrusMasterNode(*ipaddress: str = http://localhost, network: Union[CirrusMain, CirrusTest, CirrusRegTest] = CirrusMain()*)

The Cirrus member of the masternode pair.

property addressbook: *AddressBook*

The addressbook route.

Returns An addressbook instance.

Return type *AddressBook*

property balances: *Balances*

The balances route.

Returns A Balances instance

Return type *Balances*

property blockchainnetwork: `pystratis.core.networks.basenetwork.BaseNetwork`

The node's network type.

Returns The node's network.

Return type `BaseNetwork`

property blockstore: `BlockStore`

The blockstore route.

Returns A `BlockStore` instance.

Return type `BlockStore`

check_all_endpoints_implemented() → `bool`

Queries a running node's swagger schema and compares the pystratis implemented endpoints with those defined by the swagger schema.

Returns True if all endpoints are implemented, otherwise False.

Return type `bool`

property collateral: `Collateral`

The collateral route. Not available in devmode.

Returns A `Collateral` instance.

Return type `Collateral`

property connection_manager: `ConnectionManager`

The connectionmanager route.

Returns A `ConnectionManager` instance.

Return type `ConnectionManager`

property consensus: `Consensus`

The consensus route.

Returns A `Consensus` instance.

Return type `Consensus`

property contract_swagger:

`pystratis.api.contract_swagger.contract_swagger.ContractSwagger`

The `contract_swagger` route.

Returns A `ContractSwagger` instance.

Return type `ContractSwagger`

property dashboard: `Dashboard`

The dashboard route.

Returns A `Dashboard` instance.

Return type `Dashboard`

property dynamic_contract:

`pystratis.api.dynamic_contract.dynamic_contract.DynamicContract`

The `dynamic_contract` route.

Returns A `DynamicContract` instance.

Return type `DynamicContract`

property federation: *Federation*

The federation route.

Returns A Federation instance.

Return type *Federation*

property ipaddr: *str*

The node's ip address.

Returns The specified ip address of the node.

Return type *str*

property mempool: *Mempool*

The mempool route.

Returns A Mempool instance.

Return type *Mempool*

property name: *str*

The node's name.

Returns The node name.

Return type *str*

property network: *Network*

The network route.

Returns A Network instance.

Return type *Network*

property node: *Node*

The node route.

Returns A Node instance.

Return type *Node*

property notifications: *Notifications*

The notifications route.

Returns A Notifications instance.

Return type *Notifications*

property rpc: *RPC*

The RPC route.

Returns A RPC instance.

Return type *RPC*

property signalr: *SignalR*

The signalr route.

Returns A SignalR instance.

Return type *SignalR*

property smart_contract_wallet: *SmartContractWallet*

The smartcontractwallet route.

Returns A SmartContractWallet instance.

Return type `SmartContractWallet`

property smart_contracts: `SmartContracts`

The smartcontracts route.

Returns A SmartContracts instance.

Return type `SmartContracts`

stop_node() → bool

Convenience method for stopping node.

property voting: `Voting`

The voting route.

Returns A Voting instance.

Return type `Voting`

property wallet: `Wallet`

The wallet route.

Returns A Wallet instance.

Return type `Wallet`

1.3 Core

1.3.1 Subpackages

1.3.1.1 Networks

StraxNetwork

class StraxMain(*DEFAULT_PORT: int = 17105, API_PORT: int = 17103, RPC_PORT: int = 17104, SIGNALR_PORT: int = 17102*)

Describes the StraxMain network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=17105.
- **RPC_PORT** (*int, optional*) – The rpc port, if active. Default=17104.
- **API_PORT** (*int, optional*) – The API port. Default=17103.
- **SIGNALR_PORT** (*int, optional*) – The SignalR port. Default=17102.

validate_address(*address: str*) → bool

Validates an address on this network.

class StraxTest(*DEFAULT_PORT: int = 27105, API_PORT: int = 27103, RPC_PORT: int = 27104, SIGNALR_PORT: int = 27102*)

Describes the StraxTest network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=27105.
- **RPC_PORT** (*int, optional*) – The rpc port, if active. Default=27104.
- **API_PORT** (*int, optional*) – The API port. Default=27103.

- **SIGNALR_PORT** (*int, optional*) – The SignalR port. Default=27102.

validate_address(*address: str*) → bool
Validates an address on this network.

class StraxRegTest(*DEFAULT_PORT: int = 37105, API_PORT: int = 37103, RPC_PORT: int = 37104, SIGNALR_PORT: int = 37102*)

Describes the StraxRegTest network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=37105.
- **RPC_PORT** (*int, optional*) – The rpc port, if active. Default=37104.
- **API_PORT** (*int, optional*) – The API port. Default=37103.
- **SIGNALR_PORT** (*int, optional*) – The SignalR port. Default=37102.

validate_address(*address: str*) → bool
Validates an address on this network.

CirrusNetwork

class CirrusMain(*DEFAULT_PORT: int = 16179, API_PORT: int = 37223, RPC_PORT: int = 16175, SIGNALR_PORT: int = 38823*)

Describes the CirrusMain network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=16179.
- **RPC_PORT** (*int, optional*) – The rpc port, if active. Default=16175.
- **API_PORT** (*int, optional*) – The API port. Default=37223.
- **SIGNALR_PORT** (*int, optional*) – The SignalR port. Default=38823.

validate_address(*address: str*) → bool
Validates an address on this network.

class CirrusTest(*DEFAULT_PORT: int = 26179, API_PORT: int = 38223, RPC_PORT: int = 26175, SIGNALR_PORT: int = 39823*)

Describes the CirrusTest network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=26179.
- **RPC_PORT** (*int, optional*) – The rpc port, if active. Default=26175.
- **API_PORT** (*int, optional*) – The API port. Default=38223.
- **SIGNALR_PORT** (*int, optional*) – The SignalR port. Default=39823.

validate_address(*address: str*) → bool
Validates an address on this network.

class CirrusRegTest(*DEFAULT_PORT: int = 26179, API_PORT: int = 38223, RPC_PORT: int = 26175, SIGNALR_PORT: int = 39823*)

Describes the CirrusRegTest network.

Parameters

- **DEFAULT_PORT** (*int, optional*) – The network communication port. Default=26179.

- `RPC_PORT` (*int*, *optional*) – The rpc port, if active. Default=26175.
- `API_PORT` (*int*, *optional*) – The API port. Default=38223.
- `SIGNALR_PORT` (*int*, *optional*) – The SignalR port. Default=39823.

`validate_address`(*address: str*) → bool
Validates an address on this network.

Ethereum

`class Ethereum`(*DEFAULT_PORT: int = 30303*)
Default settings for the ethereum network.

Parameters `DEFAULT_PORT` (*int*, *optional*) – The network communication port. Default=30303.

1.3.1.2 Types

Address

`class Address`(*address: str*, *network: pystratis.core.networks.basenetwork.BaseNetwork*)
A address model. Address is validated by the network.

address

network

static validate_values(*address: str*, *network: pystratis.core.networks.basenetwork.BaseNetwork*) → bool

hexstr

`class hexstr`(*content*, **args*, ***kwargs*)
Represents an hex string.

to_bytes() → bytes

int32

`class int32`(*value*)
Represents an int32.

classmethod hex_to_int(*v: Union[str, hexstr]*) → int

to_hex() → str

validate_value(*value*) → int

property value: int

int64**class** `int64`(*value*)

Represents an int64.

classmethod `hex_to_int`(*v*: *Union[str, hexstr]*) → int`to_hex`() → str`validate_value`(*value*) → int**property value:** int**Money****class** `Money`(*value*: *Union[Money, float, decimal.Decimal, int, str]*)

Represents Money.

In Stratis Platform, the money is represented by STRAX coin. A satoshi is the smallest unit of a STRAX. One STRAX is equivalent to 100 millionth of a satoasis (just like in Bitcoin).

Parameters *value* (`Money`, *float*, *Decimal*, *int*, *str*) – An amount of money. The value interpreted as a count of the STRAX coins.

Raises `ValueError` – Attempt to create Money with unsupported *value* type. Attempt to create Money with negative *value*.

property value: `decimal.Decimal`

The amount of money, represented by fixed-point STRAX amount.

Returns The amount of money.**Return type** `Decimal`**classmethod** `from_satoshi_units`(*value*: *int*) → *Money*

Convert satoasis to Money object. 1 STRAX is equivalent to 100 millionth of a satoasis.

Parameters *value* (*int*) – Amount of satoasis.**Returns** The Money object.**Return type** `Money``to_coin_unit`() → str

Represent Money object as a string.

Returns The string contains float representation of STRAX amount. For example, 1 STRAX will be represented as '1.00000000'.

Return type *str*

uint128

```
class uint128(value)
    Represents an uint128.
    classmethod hex_to_int(v: Union[str, hexstr]) → int
    to_hex() → str
    validate_value(value) → int
    property value: int
```

uint160

```
class uint160(value)
    Represents an uint160.
    classmethod hex_to_int(v: Union[str, hexstr]) → int
    to_hex() → str
    validate_value(value) → int
    property value: int
```

uint256

```
class uint256(value)
    Represents an uint256.
    classmethod hex_to_int(v: Union[str, hexstr]) → int
    to_hex() → str
    validate_value(value) → int
    property value: int
```

uint32

```
class uint32(value)
    Represents an uint32.
    classmethod hex_to_int(v: Union[str, hexstr]) → int
    to_hex() → str
    validate_value(value) → int
    property value: int
```

uint64

```

class uint64(value)
    Represents an uint64.

    classmethod hex_to_int(v: Union[str, hexstr]) → int
    to_hex() → str
    validate_value(value) → int
    property value: int

```

1.3.2 CoinType

```

class CoinType(value)
    Enum representing type of coin, as specified in BIP44. Registered cointypes specified in SLIP44.
    Corresponding type from StratisFullNode's implementation can be found here.

```

Note: Coin type for Cirrus mainnet is not a registered coin type (as well as testnets). According to SLIP44, ID 401 belongs to another coin, that has nothing to do with Stratis Platform.

```

Bitcoin = 0
Testnet = 1
CirrusTest = 400
Cirrus = 401
Strax = 105105

```

1.3.3 ContractTransactionItemType

```

class ContractTransactionItemType(value)
    Enum representing type of contract-related transaction item.
    Corresponding type from StratisFullNode's implementation can be found here.

```

```

Received = 0
Send = 1
Staked = 2
ContractCall = 3
ContractCreate = 4
GasRefund = 5

```

1.3.4 ConversionRequestStatus

class ConversionRequestStatus(*value*)

Enum representing status of interop conversion request.

Corresponding type from StratisFullNode's implementation can be found [here](#).

Unprocessed = 0

Submitted = 1

Processed = 2

OriginatorNotSubmitted = 3

OriginatorSubmitted = 4

VoteFinalised = 5

NotOriginator = 6

1.3.5 ConversionRequestType

class ConversionRequestType(*value*)

Enum representing type of interop conversion request.

Corresponding type from StratisFullNode's implementation can be found [here](#).

Mint = 0

Burn = 1

1.3.6 CrosschainTransferStatus

class CrossChainTransferStatus(*value*)

Enum representing status of cross chain status.

Corresponding type from StratisFullNode's implementation can be found [here](#).

Suspended = 'Suspended'

Partial = 'Partial'

FullySigned = 'FullySigned'

SeenInBlock = 'SeenInBlock'

Rejected = 'Rejected'

1.3.7 Deposit

class Deposit(**, id: uint256, amount: Money, targetAddress: Address, targetChain: DestinationChain = None, blockNumber: int, blockHash: uint256, retrievalType: DepositRetrievalType*)

A pydantic model representing a deposit made to a sidechain multisig, with the aim of triggering a cross chain transfer.

Note: Learn how to [acquire CRS token using GUI](#).

deposit_id: *uint256*

The hash of the source transaction that originates the fund transfer.

amount: *Money*

The amount of the requested funds transfer.

target_address: *Address*

The target address, on the target chain, for the fund deposited on the multisig.

target_chain: *Optional[DestinationChain]*

Chain on which STRAX minting or burning should occur.

block_number: *int*

The block number where the source deposit has been persisted.

block_hash: *uint256*

The hash of the block where the source deposit has been persisted.

retrieval_type: *DepositRetrievalType*

Whether the deposit is a “faster” or “normal” deposit.

1.3.8 DepositRetrievalType

class *DepositRetrievalType*(*value*)

Represents type of deposit retrieval.

Small deposits are processed after *IFederatedPegSettings.MinimumConfirmationsSmallDeposits* confirmations (blocks).

Normal deposits are processed after *IFederatedPegSettings.MinimumConfirmationsNormalDeposits* confirmations (blocks).

Large deposits are only processed after the height has increased past max re-org (*IFederatedPegSettings.MinimumConfirmationsLargeDeposits*) confirmations (blocks).

Conversion deposits are processed after similar intervals to the above, according to their size.

Reward distribution deposits are only processed after the height has increased past max re-org (*IFederatedPegSettings.MinimumConfirmationsDistributionDeposits*) confirmations (blocks).

Small = 0**Normal** = 1**Large** = 2**Distribution** = 3**ConversionSmall** = 4**ConversionNormal** = 5**ConversionLarge** = 6

1.3.9 DestinationChain

class `DestinationChain`(*value*)

Chains supported by InterFlux integration. Symbols are defined according to the [SLIP44](#) specification.

STRAX = 0

ETH = 1

BNB = 2

ETC = 3

AVAX = 4

ADA = 5

1.3.10 ExtKey

class `ExtKey`(*value: Union[bytes, str, Key]*)

Type representing extended private key, as specified in [BIP32](#).

Corresponding type from StratisFullNode's implementation can be found [here](#).

Parameters *value* (*bytes*, *str*, *Key*) – data for a private key.

Raises `ValueError` – Attempt to create `ExtKey` with unsupported *value* type. Attempt to create with incorrect length.

generate_private_key_bytes() → bytes

Get private key from this extended private key.

Returns private key, represented by the first 32 bytes of extended private key.

Return type *bytes*

generate_chain_code_bytes() → bytes

Get chain code from this extended private key.

Returns chain code, represented by the last 32 bytes of extended private key.

Return type *bytes*

generate_private_key_base58() → str

Get Base58-encoded private key from this extended private key.

Returns base58-encoded private key

Return type *str*

generate_private_key() → *Key*

Get private key from this extended private key.

Returns private key.

Return type *Key*

generate_wif_key() → str

Convert current key to [Wallet import format](#)

Returns WIF compilant key.

Return type *str*

get_bytes() → bytes
 Get private key bytes
Returns raw private key data
Return type *bytes*

1.3.11 ExtPubKey

class ExtPubKey(*extpubkey: str*)
 Type representing extended public key, as specified in [BIP32](#).
 Corresponding type from StratisFullNode's implementation can be found [here](#).

Parameters **extpubkey** (*str*) – encoded extended public key.

version

depth

parent_fingerprint

index

chain_code

key

checksum

decode_extpubkey(*extpubkey: str*)
 Deserialize extended public key, as specified in [BIP32](#).

Parameters **extpubkey** (*str*) – Base58 encoded serialized extended public key.

Raises **AssertionError** – extpubkey has incorrect format.

get_payload() → bytes
 Serialize extended public key, as specified in [BIP32](#).

Returns serialized extended public key.

Return type *bytes*

1.3.12 Key

class Key(*value: Union[bytes, str, Key]*)
 Type representing [private key](#). A private key is a secret number, known only to the person that generated it.
 Corresponding type from StratisFullNode's implementation can be found [here](#).

Parameters **value** (*bytes, str, Key*) – data for private key.

Raises **ValueError** – Attempt to create Key with unsupported *value* type.

get_bytes() → bytes
 Get private key bytes

Returns raw private key data

Return type *bytes*

generate_wif_key() → str
 Convert current key to [Wallet import format](#)

Returns WIF compliant key.

Return type *str*

1.3.13 MultisigSecret

class MultisigSecret(**, mnemonic: pydantic.types.SecretStr, passphrase: pydantic.types.SecretStr*)

A MultisigSecret.

mnemonic: `pydantic.types.SecretStr`

passphrase: `pydantic.types.SecretStr`

1.3.14 Outpoint

class Outpoint(**, transactionId: uint256 = None, index: pystratis.core.outpoint.ConstrainedIntValue*)

A pydantic model representing an outpoint.

transaction_id: `Optional[uint256]`

The transaction hash of the unspent output.

index: `pystratis.core.outpoint.ConstrainedIntValue`

The index of the outpoint. Must be ≥ 0 .

1.3.15 PubKey

class PubKey(*value: str*)

Type representing public key. A public key is the number that corresponds to a private key, but does not need to be kept secret. A public key can be calculated from a private key, but not vice versa.

A public key can be presented in compressed or uncompressed format.

Note: Read more about [public key formats](#).

x

y

uncompressed() \rightarrow str

Retrieves a uncompressed pubkey.

compressed() \rightarrow str

Retreives a compressed pubkey.

1.3.16 Recipient

```
class Recipient(*, destinationAddress: Address = None, destinationScript: Address = None,
               subtractFeeFromAmount: bool = True, amount: Money)
```

A pydantic model for a recipient.

destination_address: `Optional[Address]`

The destination address, if applicable.

destination_script: `Optional[Address]`

The destination script, if applicable.

subtraction_fee_from_amount: `Optional[bool]`

If true, subtract fee from amount.

amount: `Money`

The amount to send to this recipient.

1.3.17 SmartContractParameter

```
class SmartContractParameter(value_type: SmartContractParameterType, value: Any)
```

Type representing smart contract's parameter.

Parameters

- **value_type** (`SmartContractParameterType`) – The type of parameter.
- **value** – The value of the parameter.

Note: Learn more about smart contracts in [Stratis Academy](#).

```
static validate_values(value_type: SmartContractParameterType, value: Any) → bool
```

Validates that type of value matching with value_type.

1.3.18 SmartContractParameterType

```
class SmartContractParameterType(value)
```

Defines (de-)serialization rule for smart contract parameters.

Notes

Learn more about contract's parameters serialization from [Stratis Academy](#).

Boolean = 1

Byte = 2

Char = 3

String = 4

UInt32 = 5

Int32 = 6

UInt64 = 7

Int64 = 8

```
Address = 9
ByteArray = 10
UInt128 = 11
UInt256 = 12
```

1.3.19 TransactionItemType

```
class TransactionItemType(value)
    A TransactionItemType.
    Received = 'received'
    Send = 'send'
    Staked = 'staked'
    Mined = 'mined'
```

1.3.20 WalletSecret

```
class WalletSecret(*, walletName: str, walletPassword: pydantic.types.SecretStr)
    A pydantic model representing credentials of the wallet.
    wallet_name: str
        The name of the wallet.
    wallet_password: pydantic.types.SecretStr
        The wallet password.
```

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pystratis.api.addressbook.responsemodels.addressbookentrymodel, 23
2
pystratis.api.addressbook.responsemodels.addressbookentrymodel, 23
pystratis.api.apierror, 102
pystratis.api.blockstore.responsemodels.addressbalancemodel, 23
4
pystratis.api.blockstore.responsemodels.addressindexertipmodel, 25
5
pystratis.api.blockstore.responsemodels.balancechangesmodel, 26
5
pystratis.api.blockstore.responsemodels.getaddressesbalancesmodel, 28
8
pystratis.api.blockstore.responsemodels.getlastbalanceupdatetransactionmodel, 29
8
pystratis.api.blockstore.responsemodels.getutxosforaddressmodel, 30
8
pystratis.api.blockstore.responsemodels.getverboseaddressesbalancesmodel, 30
9
pystratis.api.blockstore.responsemodels.utxomodel, 31
9
pystratis.api.blockstore.responsemodels.verboseaddressbalancemodel, 31
9
pystratis.api.coldstaking.responsemodels.accountmodel, 31
15
pystratis.api.coldstaking.responsemodels.addressmodel, 34
15
pystratis.api.coldstaking.responsemodels.infomodel, 89
16
pystratis.api.coldstaking.responsemodels.setupmodel, 90
16
pystratis.api.coldstaking.responsemodels.withdrawalmodel, 90
16
pystratis.api.collateral.responsemodels.joinfederationresponsemodel, 91
17
pystratis.api.connectionmanager.responsemodels.peerinfomodel, 91
18
pystratis.api.consensus.responsemodels.deploymentflagmodel, 91
20
pystratis.api.diagnostic.responsemodels.connectedpeerinfomodel, 93
22
pystratis.api.diagnostic.responsemodels.getconnectedpeersinfomodel, 94
23
pystratis.api.diagnostic.responsemodels.getstatusmodel, 23
pystratis.api.diagnostic.responsemodels.peerstatisticsmodel, 23
pystratis.api.featureinitializationstate, 101
pystratis.api.federation.responsemodels.federationmembermodel, 25
pystratis.api.federation.responsemodels.federationmembermodel, 26
pystratis.api.federationgateway.responsemodels.crosschaintransactionmodel, 28
pystratis.api.federationgateway.responsemodels.federationmembermodel, 29
pystratis.api.federationgateway.responsemodels.federationmembermodel, 30
pystratis.api.federationgateway.responsemodels.maturedblockmodel, 31
pystratis.api.federationgateway.responsemodels.serializableblockmodel, 31
pystratis.api.federationgateway.responsemodels.validatetransactionmodel, 31
pystratis.api.federationwallet.responsemodels.withdrawalmodel, 34
pystratis.api.fullnodestate, 101
pystratis.api.global_responsemodels.accountbalancemodel, 89
pystratis.api.global_responsemodels.addressbalancemodel, 90
pystratis.api.global_responsemodels.addressdescriptor, 90
pystratis.api.global_responsemodels.addressesmodel, 91
pystratis.api.global_responsemodels.addressmodel, 91
pystratis.api.global_responsemodels.blockmodel, 91
pystratis.api.global_responsemodels.blocktransactiondetailmodel, 93
pystratis.api.global_responsemodels.buildcontracttransactionmodel, 94

pystratis.api.global_responsemodels.buildcreatecontracttransactionmodel, 50
 94 pystratis.api.rpc.responsemodels.rpccommandresponsemodel,
 pystratis.api.global_responsemodels.buildofflinesignaturemodel, 50
 95 pystratis.api.signalr.responsemodels.getconnectioninfomodel,
 pystratis.api.global_responsemodels.buildtransactionmodel, 50
 95 pystratis.api.smartcontracts.responsemodels.addressbalance
 pystratis.api.global_responsemodels.maturedblockinfomodel, 57
 96 pystratis.api.smartcontracts.responsemodels.getcodemodel,
 pystratis.api.global_responsemodels.pollviewmodel, 58
 96 pystratis.api.smartcontracts.responsemodels.localexecution
 pystratis.api.global_responsemodels.removedtransactionmodel, 58
 97 pystratis.api.smartcontracts.responsemodels.logmodel,
 pystratis.api.global_responsemodels.scriptpubkey, 59
 97 pystratis.api.smartcontracts.responsemodels.receiptmodel,
 pystratis.api.global_responsemodels.scriptsig, 59
 97 pystratis.api.smartcontracts.responsemodels.transferinfomodel,
 pystratis.api.global_responsemodels.transactionmodel, 60
 98 pystratis.api.smartcontractwallet.responsemodels.contractt
 pystratis.api.global_responsemodels.transactionoutputmodel, 60
 98 pystratis.api.staking.responsemodels.getstakinginfomodel,
 pystratis.api.global_responsemodels.utxodescriptor, 65
 99 pystratis.api.voting.responsemodels.votingdatamodel,
 pystratis.api.global_responsemodels.vin, 99 68
 pystratis.api.global_responsemodels.vout, 99 pystratis.api.voting.responsemodels.whitelistedhashesmodel
 pystratis.api.global_responsemodels.walletbalancemodel, 69
 100 pystratis.api.wallet.responsemodels.accounthistorymodel,
 pystratis.api.global_responsemodels.walletgeneralinfomodel, 69
 100 pystratis.api.wallet.responsemodels.distributeutxomodel,
 pystratis.api.global_responsemodels.walletpendtransactionmodel, 69
 100 pystratis.api.wallet.responsemodels.maxspendableamountmodel,
 pystratis.api.interop.responsemodels.conversionrequestmodel, 69
 38 pystratis.api.wallet.responsemodels.paymentdetailmodel,
 pystratis.api.interop.responsemodels.transactionresponsemodel, 69
 39 pystratis.api.wallet.responsemodels.spendabletransactionmo
 pystratis.api.logrule, 101 85
 pystratis.api.mining.responsemodels.generateblockmodel, 85
 40 pystratis.api.wallet.responsemodels.spendabletransactions
 pystratis.api.network.responsemodels.bannedpeersmodel, 86
 42 pystratis.api.wallet.responsemodels.transactionitemmodel,
 pystratis.api.node.responsemodels.asyncloopsmodel, 86
 45 pystratis.api.wallet.responsemodels.utxoamountmodel,
 pystratis.api.node.responsemodels.blockheadermodel, 87
 45 pystratis.api.wallet.responsemodels.utxoperblockmodel,
 pystratis.api.node.responsemodels.connectedpeersmodel, 87
 46 pystratis.api.wallet.responsemodels.utxopertransactionmo
 pystratis.api.node.responsemodels.featuresdatamodel, 87
 46 pystratis.api.wallet.responsemodels.wallethistorymodel,
 pystratis.api.node.responsemodels.gettxoutmodel, 88
 46 pystratis.api.wallet.responsemodels.walletstatsmodel,
 pystratis.api.node.responsemodels.statusmodel, pystratis.core.cointype, 123
 46 pystratis.core.contracttransactionitemtype,
 pystratis.api.node.responsemodels.validateaddressmodel, 123
 48 pystratis.core.conversionrequeststatus, 124
 pystratis.api.rpc.responsemodels.rpccommandlistmodel, pystratis.core.conversionrequesttype, 124

pystratis.core.crosschaintransferstatus, 124
pystratis.core.deposit, 124
pystratis.core.depositretrievaltype, 125
pystratis.core.destinationchain, 126
pystratis.core.extkey, 126
pystratis.core.extpubkey, 127
pystratis.core.key, 127
pystratis.core.multisigsecret, 128
pystratis.core.outpoint, 128
pystratis.core.pubkey, 128
pystratis.core.recipient, 129
pystratis.core.smartcontractparameter, 129
pystratis.core.smartcontractparametertype,
129
pystratis.core.transactionitemtype, 130
pystratis.core.types.address, 120
pystratis.core.types.hexstr, 120
pystratis.core.types.int32, 120
pystratis.core.types.int64, 121
pystratis.core.types.money, 121
pystratis.core.types.uint128, 122
pystratis.core.types.uint160, 122
pystratis.core.types.uint256, 122
pystratis.core.types.uint32, 122
pystratis.core.types.uint64, 123
pystratis.core.walletsecret, 130

Symbols

`__call__()` (*AddressBook* method), 1

A

`account()` (*ColdStaking* method), 10

`account()` (*Wallet* method), 76

`account_addresses()` (*SmartContractWallet* method), 60

`account_hd_path` (*AccountBalanceModel* attribute), 89

`account_hd_path` (*AccountHistoryModel* attribute), 82

`account_name` (*AccountBalanceModel* attribute), 89

`account_name` (*AccountHistoryModel* attribute), 82

`account_name` (*AccountModel* attribute), 15

`AccountBalanceModel` (class in *pys-tratis.api.global_responsemodels.accountbalance* model), 89

`AccountHistoryModel` (class in *pys-tratis.api.wallet.responsemodels.accounthistory* model), 82

`AccountModel` (class in *pys-tratis.api.coldstaking.responsemodels.account* model), 15

`accounts()` (*Wallet* method), 76

`active` (*FederationGatewayInfoModel* attribute), 29

`ADA` (*DestinationChain* attribute), 126

`add()` (*AddressBook* method), 1

`add_owner()` (*Interop* method), 35

`addnode` (*PeerInfoModel* attribute), 19

`addnode()` (*ConnectionManager* method), 17

`addr` (*PeerInfoModel* attribute), 18

`address` (*Address* attribute), 120

`address` (*AddressBalanceModel* attribute), 4, 57, 90

`address` (*AddressBookEntryModel* attribute), 2

`address` (*AddressDescriptor* attribute), 90

`address` (*AddressModel* attribute), 15, 91

`Address` (class in *pystratis.core.types.address*), 120

`address` (*LogModel* attribute), 59

`Address` (*SmartContractParameterType* attribute), 130

`address` (*SpendableTransactionModel* attribute), 85

`address` (*TransactionOutputModel* attribute), 98

`address` (*ValidateAddressModel* attribute), 48

`address` (*VerboseAddressBalanceModel* attribute), 9

`address()` (*ColdStaking* method), 10

`address_balance()` (*SmartContractWallet* method), 60

`address_balances()` (*SmartContracts* method), 57

`address_type` (*AddressDescriptor* attribute), 90

`AddressBalanceModel` (class in *pys-tratis.api.blockstore.responsemodels.addressbalance* model), 4

`AddressBalanceModel` (class in *pys-tratis.api.global_responsemodels.addressbalance* model), 90

`AddressBalanceModel` (class in *pys-tratis.api.smartcontracts.responsemodels.addressbalance* model), 57

`addressbook` (*CirrusMasterNode* property), 115

`addressbook` (*CirrusNode* property), 105

`AddressBook` (class in *pystratis.api.addressbook*), 1

`addressbook` (*InterfluxCirrusNode* property), 111

`addressbook` (*InterfluxStraxNode* property), 108

`addressbook` (*StraxMasterNode* property), 113

`addressbook` (*StraxNode* property), 103

`AddressBookEntryModel` (class in *pys-tratis.api.addressbook.responsemodels.addressbookentry* model), 2

`AddressDescriptor` (class in *pys-tratis.api.global_responsemodels.addressdescriptor*), 90

`addresses` (*AccountBalanceModel* attribute), 90

`addresses` (*AddressesModel* attribute), 91

`addresses` (*BuildOfflineSignModel* attribute), 95

`addresses` (*ScriptPubKey* attribute), 97

`addresses()` (*Wallet* method), 77

`AddressesModel` (class in *pys-tratis.api.global_responsemodels.addresses* model), 91

`addressindexer_tip()` (*BlockStore* method), 2

`AddressIndexerTipModel` (class in *pys-tratis.api.blockstore.responsemodels.addressindexertip* model), 5

`AddressModel` (class in *pys-tratis.api.coldstaking.responsemodels.address* model), 15

`AddressModel` (class in *pys-*

- tratis.api.global_responsemodels.addressmodel*), 91
- `addrlocal` (*PeerInfoModel* attribute), 18
- `agent` (*StatusModel* attribute), 46
- `amount` (*ContractTransactionItemModel* attribute), 63
- `amount` (*ConversionRequestModel* attribute), 39
- `amount` (*Deposit* attribute), 125
- `amount` (*PaymentDetailModel* attribute), 85
- `amount` (*Recipient* attribute), 129
- `amount` (*SpendableTransactionModel* attribute), 85
- `amount` (*TransactionItemModel* attribute), 86
- `amount` (*TransactionOutputModel* attribute), 98
- `amount` (*UtxoAmountModel* attribute), 87
- `amount` (*UtxoDescriptor* attribute), 99
- `amount` (*WithdrawalModel* attribute), 34
- `amount_confirmed` (*AccountBalanceModel* attribute), 90
- `amount_confirmed` (*AddressBalanceModel* attribute), 90
- `amount_confirmed` (*AddressModel* attribute), 91
- `amount_unconfirmed` (*AccountBalanceModel* attribute), 90
- `amount_unconfirmed` (*AddressBalanceModel* attribute), 90
- `amount_unconfirmed` (*AddressModel* attribute), 91
- `APIError`, 102
- `args` (*APIError* attribute), 102
- `asm` (*ScriptPubKey* attribute), 97
- `asm` (*ScriptSig* attribute), 97
- `async_loop_state` (*FederationMemberInfoModel* attribute), 30
- `async_loops()` (*Node* method), 44
- `asyncloops_stats()` (*Dashboard* method), 21
- `AsyncLoopsModel` (class in *pys-tratis.api.node.responsemodels.asyncloopsmodel*), 45
- `AVAX` (*DestinationChain* attribute), 126
- ## B
- `balance` (*AddressBalanceModel* attribute), 4
- `balance` (*GetUTXOsForAddressModel* attribute), 8
- `balance()` (*FederationWallet* method), 31
- `balance()` (*Interop* method), 37
- `balance()` (*SmartContracts* method), 51
- `balance()` (*Wallet* method), 72
- `balance_changed_height` (*BalanceChangesModel* attribute), 5
- `balance_changes` (*VerboseAddressBalanceModel* attribute), 9
- `BalanceChangesModel` (class in *pys-tratis.api.blockstore.responsemodels.balancechangesmodel*), 5
- `balances` (*CirrusMasterNode* property), 115
- `balances` (*CirrusNode* property), 104
- `Balances` (class in *pystratis.api.balances*), 2
- `balances` (*GetAddressesBalancesModel* attribute), 8
- `balances` (*InterfluxCirrusNode* property), 109
- `balances` (*StraxMasterNode* property), 113
- `balances` (*WalletBalanceModel* attribute), 100
- `balances_data` (*GetVerboseAddressesBalancesModel* attribute), 9
- `ban_reason` (*BannedPeerModel* attribute), 42
- `ban_until` (*BannedPeerModel* attribute), 42
- `BannedPeerModel` (class in *pys-tratis.api.network.responsemodels.bannedpeermodel*), 42
- `banscore` (*PeerInfoModel* attribute), 19
- `best_block` (*GetTxOutModel* attribute), 46
- `best_peer_height` (*StatusModel* attribute), 47
- `Bitcoin` (*CoinType* attribute), 123
- `bits` (*BlockHeaderModel* attribute), 45
- `bits` (*BlockModel* attribute), 92
- `bits` (*BlockTransactionDetailsModel* attribute), 93
- `block()` (*BlockStore* method), 3
- `block_hash` (*Deposit* attribute), 125
- `block_hash` (*MaturedBlockInfoModel* attribute), 96
- `block_hash` (*ReceiptModel* attribute), 59
- `block_hash` (*WithdrawalModel* attribute), 34
- `block_height` (*ContractTransactionItemModel* attribute), 63
- `block_height` (*ConversionRequestModel* attribute), 38
- `block_height` (*GetLastBalanceUpdateTransactionModel* attribute), 8
- `block_height` (*MaturedBlockInfoModel* attribute), 96
- `block_height` (*WithdrawalModel* attribute), 34
- `block_index` (*TransactionItemModel* attribute), 86
- `block_info` (*MaturedBlockDepositsModel* attribute), 31
- `block_number` (*Deposit* attribute), 125
- `block_number` (*ReceiptModel* attribute), 60
- `block_time` (*MaturedBlockInfoModel* attribute), 96
- `blockchainnetwork` (*CirrusMasterNode* property), 115
- `blockchainnetwork` (*CirrusNode* property), 105
- `blockchainnetwork` (*InterfluxCirrusNode* property), 111
- `blockchainnetwork` (*InterfluxStraxNode* property), 108
- `blockchainnetwork` (*StraxMasterNode* property), 113
- `blockchainnetwork` (*StraxNode* property), 103
- `blockhash` (*TransactionModel* attribute), 98
- `BlockHeaderModel` (class in *pys-tratis.api.node.responsemodels.blockheadermodel*), 45
- `BlockModel` (class in *pys-tratis.api.global_responsemodels.blockmodel*), 91
- `blocks()` (*DeploymentFlagsModel* attribute), 20
- `blocks` (*GenerateBlocksModel* attribute), 40
- `blockstore` (*CirrusMasterNode* property), 116
- `blockstore` (*CirrusNode* property), 105

- BlockStore (class in *pystratis.api.blockstore*), 2
 blockstore (*InterfluxCirrusNode* property), 111
 blockstore (*InterfluxStraxNode* property), 108
 blockstore (*StraxMasterNode* property), 113
 blockstore (*StraxNode* property), 103
 blockstore_height (*StatusModel* attribute), 47
 blocktime (*TransactionModel* attribute), 98
 BlockTransactionDetailsModel (class in *pystratis.api.global_responsemodels.blocktransactiondetailsmodel*), 93
 blocktrust (*BlockModel* attribute), 92
 blocktrust (*BlockTransactionDetailsModel* attribute), 94
 bloom (*ReceiptModel* attribute), 60
 BNB (*DestinationChain* attribute), 126
 Boolean (*SmartContractParameterType* attribute), 129
 build_and_send_call() (*SmartContracts* method), 55
 build_and_send_create() (*SmartContracts* method), 55
 build_call() (*SmartContracts* method), 53
 build_create() (*SmartContracts* method), 52
 build_interflux_transaction() (*Wallet* method), 75
 build_offline_sign_request() (*Wallet* method), 81
 build_transaction() (*Multisig* method), 40
 build_transaction() (*SmartContracts* method), 53
 build_transaction() (*Wallet* method), 74
 BuildContractTransactionModel (class in *pystratis.api.global_responsemodels.buildcontracttransactionmodel*), 94
 BuildCreateContractTransactionModel (class in *pystratis.api.global_responsemodels.buildcreatecontracttransactionmodel*), 94
 BuildOfflineSignModel (class in *pystratis.api.global_responsemodels.buildofflinesignaturemodel*), 95
 BuildTransactionModel (class in *pystratis.api.global_responsemodels.buildtransactionmodel*), 95
 Burn (*ConversionRequestType* attribute), 124
 Byte (*SmartContractParameterType* attribute), 129
 ByteArray (*SmartContractParameterType* attribute), 130
 bytecode (*GetCodeModel* attribute), 58
 bytes_received (*PeerStatisticsModel* attribute), 23
 bytes_sent (*PeerStatisticsModel* attribute), 23
 bytesrecv (*PeerInfoModel* attribute), 18
 bytesrecv_per_msg (*PeerInfoModel* attribute), 19
 bytesent (*PeerInfoModel* attribute), 18
 bytesent_per_msg (*PeerInfoModel* attribute), 19
C
 call() (*SmartContractWallet* method), 62
 call_by_name() (*RPC* method), 49
 ccts_height (*FederationMemberInfoModel* attribute), 30
 ccts_next_deposit_height (*FederationMemberInfoModel* attribute), 30
 ccts_partials (*FederationMemberInfoModel* attribute), 30
 ccts_suspended (*FederationMemberInfoModel* attribute), 30
 chain_code (*ExtPubKey* attribute), 127
 chain_tip (*WalletGeneralInfoModel* attribute), 100
 chaintrust (*BlockModel* attribute), 92
 chaintrust (*BlockTransactionDetailsModel* attribute), 94
 chainwork (*BlockModel* attribute), 92
 chainwork (*BlockTransactionDetailsModel* attribute), 93
 change_requirement() (*Interop* method), 36
 Char (*SmartContractParameterType* attribute), 129
 check_all_endpoints_implemented() (*CirrusMasterNode* method), 116
 check_all_endpoints_implemented() (*CirrusNode* method), 106
 check_all_endpoints_implemented() (*InterfluxCirrusNode* method), 111
 check_all_endpoints_implemented() (*InterfluxStraxNode* method), 108
 check_all_endpoints_implemented() (*StraxMasterNode* method), 113
 check_all_endpoints_implemented() (*StraxNode* method), 103
 checksum (*ExtPubKey* attribute), 127
 Cirrus (*CoinType* attribute), 123
 CirrusMain (class in *pystratis.core.networks.cirrusnetwork*), 119
 CirrusMasterNode (class in *pystratis.nodes.cirrusminernode*), 115
 CirrusNode (class in *pystratis.nodes.cirrusnode*), 104
 CirrusRegTest (class in *pystratis.core.networks.cirrusnetwork*), 119
 CirrusTest (class in *pystratis.core.networks.cirrusnetwork*), 119
 CirrusTest (*CoinType* attribute), 123
 clear_banned() (*Network* method), 41
 code (*APIError* attribute), 102
 code() (*SmartContracts* method), 51
 coin_ticker (*StatusModel* attribute), 47
 coin_type (*AccountBalanceModel* attribute), 89
 coin_type (*AccountHistoryModel* attribute), 82
 coin_type (*AddressBalanceModel* attribute), 90
 coinbase (*GetTxOutModel* attribute), 46
 coinbase (*VIn* attribute), 99
 CoinType (class in *pystratis.core.cointype*), 123
 cold_wallet_account_exists (*InfoModel* attribute),

- 16
- ColdStaking (class in *pystratis.api.coldstaking*), 9
- coldstaking (StraxNode property), 102
- collateral (CirrusMasterNode property), 116
- collateral (CirrusNode property), 104
- Collateral (class in *pystratis.api.collateral*), 16
- collateral (InterfluxCirrusNode property), 109
- collateral (InterfluxStraxNode property), 107
- collateral (StraxMasterNode property), 113
- collateral_amount (FederationMemberDetailedModel attribute), 25
- collateral_amount (FederationMemberModel attribute), 26
- collateral_voting (InterfluxCirrusNode property), 110
- collateral_voting (InterfluxStraxNode property), 107
- CollateralVoting (class in *pys-tratis.api.collateralvoting*), 17
- command (RPCCommandListModel attribute), 50
- compressed() (PubKey method), 128
- confirm_transaction() (Interop method), 36
- confirmation_period (DeploymentFlagsModel attribute), 20
- confirmations (BlockModel attribute), 91
- confirmations (BlockTransactionDetailsModel attribute), 93
- confirmations (GetTxOutModel attribute), 46
- confirmations (SpendableTransactionModel attribute), 85
- confirmations (TransactionModel attribute), 98
- confirmed_in_block (TransactionItemModel attribute), 86
- connected (PeerStatisticsModel attribute), 23
- connected_nodes (WalletGeneralInfoModel attribute), 100
- connected_peers (GetConnectedPeersInfoModel attribute), 23
- connected_peers_not_in_peers_by_peer_id (Get-ConnectedPeersInfoModel attribute), 23
- ConnectedPeerInfoModel (class in *pys-tratis.api.diagnostic.responsemodels.connectedpeerinfo*), 22
- ConnectedPeerModel (class in *pys-tratis.api.node.responsemodels.connectedpeermodel*), 46
- connection_manager (CirrusMasterNode property), 116
- connection_manager (CirrusNode property), 106
- connection_manager (InterfluxCirrusNode property), 111
- connection_manager (InterfluxStraxNode property), 108
- connection_manager (StraxMasterNode property), 113
- connection_manager (StraxNode property), 103
- ConnectionManager (class in *pys-tratis.api.connectionmanager*), 17
- conntime (PeerInfoModel attribute), 18
- consensus (CirrusMasterNode property), 116
- consensus (CirrusNode property), 106
- Consensus (class in *pystratis.api.consensus*), 19
- consensus (InterfluxCirrusNode property), 111
- consensus (InterfluxStraxNode property), 108
- consensus (StraxMasterNode property), 113
- consensus (StraxNode property), 103
- consensus_height (FederationMemberInfoModel attribute), 30
- consensus_height (StatusModel attribute), 47
- consensus_tip_height (GetVerboseAddressesBal-ancesModel attribute), 9
- consolidate() (Wallet method), 82
- contract_swagger (CirrusMasterNode property), 116
- contract_swagger (CirrusNode property), 104
- contract_swagger (InterfluxCirrusNode property), 110
- contract_swagger (StraxMasterNode property), 113
- ContractCall (ContractTransactionItemType attribute), 123
- ContractCreate (ContractTransactionItemType attribute), 123
- ContractTransactionItemModel (class in *pys-tratis.api.smartcontractwallet.responsemodels.contracttransactionitem*), 63
- ContractTransactionItemType (class in *pys-tratis.core.contracttransactionitemtype*), 123
- ConversionLarge (DepositRetrievalType attribute), 125
- ConversionNormal (DepositRetrievalType attribute), 125
- ConversionRequestModel (class in *pys-tratis.api.interop.responsemodels.conversionrequestmodel*), 38
- ConversionRequestStatus (class in *pys-tratis.core.conversionrequeststatus*), 124
- ConversionRequestType (class in *pys-tratis.core.conversionrequesttype*), 124
- ConversionSmall (DepositRetrievalType attribute), 125
- confirm (ContractAmountModel attribute), 87
- count (UtxoPerBlockModel attribute), 87
- count (UtxoPerTransactionModel attribute), 87
- create() (SmartContractWallet method), 61
- create() (Wallet method), 69
- Created (FullNodeState attribute), 101
- creation_time (RemovedTransactionModel attribute), 97
- creation_time (SpendableTransactionModel attribute), 85
- creation_time (WalletGeneralInfoModel attribute), 100
- CrossChainTransferModel (class in *pys-tratis.api.federationgateway.responsemodels.crosschaintransfermodel*), 103

- 28
- CrossChainTransferStatus (class in *pys-tratis.core.crosschaintransferstatus*), 124
- csharp (*GetCodeModel* attribute), 58
- current_block_tx (*GetStakingInfoModel* attribute), 65
- current_blocksize (*GetStakingInfoModel* attribute), 65
- ## D
- dashboard (*CirrusMasterNode* property), 116
- dashboard (*CirrusNode* property), 106
- Dashboard (class in *pystratis.api.dashboard*), 21
- dashboard (*InterfluxCirrusNode* property), 111
- dashboard (*InterfluxStraxNode* property), 108
- dashboard (*StraxMasterNode* property), 114
- dashboard (*StraxNode* property), 103
- data (*LogModel* attribute), 59
- data (*TransactionResponseModel* attribute), 39
- data_directory_path (*StatusModel* attribute), 47
- decode_extpubkey() (*ExtPubKey* method), 127
- decode_raw_transaction() (*Node* method), 43
- delete_datafolder_chain() (*Node* method), 45
- deployment_flags() (*Consensus* method), 19
- deployment_index (*DeploymentFlagsModel* attribute), 20
- deployment_name (*DeploymentFlagsModel* attribute), 20
- DeploymentFlagsModel (class in *pys-tratis.api.consensus.responsemodels.deploymentflagsmodel*), 20
- Deposit (class in *pystratis.core.deposit*), 124
- deposit_amount (*CrossChainTransferModel* attribute), 28
- deposit_height (*CrossChainTransferModel* attribute), 28
- deposit_id (*CrossChainTransferModel* attribute), 28
- deposit_id (*Deposit* attribute), 124
- deposit_id (*WithdrawalModel* attribute), 34
- deposited (*BalanceChangesModel* attribute), 5
- DepositRetrievalType (class in *pys-tratis.core.depositretrievaltype*), 125
- deposits (*MaturedBlockDepositsModel* attribute), 31
- deposits() (*FederationGateway* method), 26
- depth (*ExtPubKey* attribute), 127
- description (*RPCCommandListModel* attribute), 50
- destination (*TransactionResponseModel* attribute), 39
- destination_address (*ConversionRequestModel* attribute), 38
- destination_address (*PaymentDetailModel* attribute), 85
- destination_address (*Recipient* attribute), 129
- destination_chain (*ConversionRequestModel* attribute), 38
- destination_script (*Recipient* attribute), 129
- DestinationChain (class in *pys-tratis.core.destinationchain*), 126
- diagnostic (*CirrusNode* property), 105
- Diagnostic (class in *pystratis.api.diagnostic*), 21
- diagnostic (*StraxNode* property), 102
- difficulty (*BlockModel* attribute), 92
- difficulty (*BlockTransactionDetailsModel* attribute), 93
- difficulty (*GetStakingInfoModel* attribute), 65
- difficulty (*StatusModel* attribute), 47
- disconnect() (*Network* method), 41
- disconnect_reason (*ConnectedPeerInfoModel* attribute), 22
- Disposed (*FeatureInitializationState* attribute), 101
- Disposed (*FullNodeState* attribute), 101
- Disposing (*FeatureInitializationState* attribute), 101
- Disposing (*FullNodeState* attribute), 101
- distribute_utxos() (*Wallet* method), 80
- DistributeUtxoModel (class in *pys-tratis.api.wallet.responsemodels.distributeutxomodel*), 84
- Distribution (*DepositRetrievalType* attribute), 125
- dry_run (*DistributeUtxoModel* attribute), 84
- dynamic_contract (*CirrusMasterNode* property), 116
- dynamic_contract (*CirrusNode* property), 105
- dynamic_contract (*InterfluxCirrusNode* property), 110
- dynamic_contract (*StraxMasterNode* property), 114
- ## E
- enable_federation() (*FederationWallet* method), 32
- enabled (*GetStakingInfoModel* attribute), 65
- endpoint (*BannedPeerModel* attribute), 42
- endpoint (*ConnectedPeerInfoModel* attribute), 22
- error (*ReceiptModel* attribute), 60
- error_message (*LocalExecutionResultModel* attribute), 59
- errors (*GetStakingInfoModel* attribute), 65
- errors (*ValidateTransactionResultModel* attribute), 31
- estimate_fee() (*SmartContracts* method), 54
- estimate_offline_setup_tx_fee() (*ColdStaking* method), 12
- estimate_offline_withdrawal_tx_fee() (*ColdStaking* method), 13
- estimate_setup_tx_fee() (*ColdStaking* method), 11
- estimate_txfee() (*Wallet* method), 73
- estimate_withdrawal_tx_fee() (*ColdStaking* method), 14
- ETC (*DestinationChain* attribute), 126
- ETH (*DestinationChain* attribute), 126
- Ethereum (class in *pystratis.core.networks.ethereum*), 120
- executed (*TransactionResponseModel* attribute), 39
- executed_polls() (*Voting* method), 66
- expected_time (*GetStakingInfoModel* attribute), 65

- external_address (*StatusModel* attribute), 47
externalapi (*InterfluxCirrusNode* property), 110
externalapi (*StraxNode* property), 102
ExtKey (*class in pystratis.core.extkey*), 126
ExtPubKey (*class in pystratis.core.extpubkey*), 127
extpubkey() (*Wallet* method), 78
- ## F
- Failed (*FullNodeState* attribute), 101
FeatureInitializationState (*class in pys-
tratis.api.featureinitializationstate*), 101
features_data (*StatusModel* attribute), 47
FeaturesDataModel (*class in pys-
tratis.api.node.responsemodels.featuresdatamodel*),
46
federation (*CirrusMasterNode* property), 116
federation (*CirrusNode* property), 105
Federation (*class in pystratis.api.federation*), 24
federation (*InterfluxCirrusNode* property), 110
federation (*StraxMasterNode* property), 114
federation_at_height() (*Federation* method), 24
federation_connection_state (*FederationMember-
InfoModel* attribute), 30
federation_gateway (*InterfluxCirrusNode* property),
110
federation_gateway (*InterfluxStraxNode* property),
107
federation_member_connections (*FederationMem-
berInfoModel* attribute), 30
federation_member_ip (*FederationMemberConnec-
tionInfoModel* attribute), 30
federation_mining_pubkeys (*FederationGatewayIn-
foModel* attribute), 29
federation_multisig_pubkeys (*FederationGateway-
InfoModel* attribute), 29
federation_size (*FederationMemberDetailedModel*
attribute), 25
federation_wallet (*InterfluxCirrusNode* property),
110
federation_wallet (*InterfluxStraxNode* property), 107
federation_wallet_active (*FederationMemberInfo-
Model* attribute), 30
federation_wallet_height (*FederationMemberInfo-
Model* attribute), 30
FederationGateway (*class in pys-
tratis.api.federationgateway*), 26
FederationGatewayInfoModel (*class in pys-
tratis.api.federationgateway.responsemodels.federationgatewayinfomodel*),
29
FederationMemberConnectionInfoModel
(*class in pys-
tratis.api.federationgateway.responsemodels.federationmemberconnectioninfomodel*),
30
FederationMemberDetailedModel (*class in pys-
tratis.api.federation.responsemodels.federationmemberdetailedmodel*),
25
FederationMemberInfoModel (*class in pys-
tratis.api.federationgateway.responsemodels.federationmemberinfo-
model*),
30
FederationMemberModel (*class in pys-
tratis.api.federation.responsemodels.federationmembermodel*),
26
FederationWallet (*class in pys-
tratis.api.federationwallet*), 31
fee (*BuildContractTransactionModel* attribute), 94
fee (*BuildCreateContractTransactionModel* attribute),
94
fee (*BuildOfflineSignModel* attribute), 95
fee (*BuildTransactionModel* attribute), 95
fee (*MaxSpendableAmountModel* attribute), 84
fee (*TransactionItemModel* attribute), 86
filename (*LogRule* attribute), 101
finalized_transactions (*WalletStatsModel* at-
tribute), 88
finished_polls() (*Voting* method), 66
flags (*BlockModel* attribute), 92
flags (*BlockTransactionDetailsModel* attribute), 94
from_address (*ReceiptModel* attribute), 59
from_address (*TransferInfoModel* attribute), 60
from_satoshi_units() (*Money* class method), 121
FullNodeState (*class in pystratis.api.fullnodestate*),
101
FullySigned (*CrossChainTransferStatus* attribute), 124
fullysigned_transfer() (*FederationGateway*
method), 26
- ## G
- gas_consumed (*LocalExecutionResultModel* attribute),
58
gas_fee (*ContractTransactionItemModel* attribute), 63
gas_used (*ReceiptModel* attribute), 59
GasRefund (*ContractTransactionItemType* attribute), 123
general_info() (*FederationWallet* method), 31
general_info() (*Wallet* method), 71
generate() (*Mining* method), 39
generate_chain_code_bytes() (*ExtKey* method), 126
generate_private_key() (*ExtKey* method), 126
generate_private_key_base58() (*ExtKey* method),
126
generate_private_key_bytes() (*ExtKey* method),
126
generate_wif_key() (*ExtKey* method), 126
generate_wif_key() (*Key* method), 127
GenerateBlocksModel (*class in pys-
tratis.api.federationgateway.responsemodels.generateblocksmodel*),
40
get_addresses_balances() (*BlockStore* method), 3

- get_bans() (*Network method*), 41
 get_best_blockhash() (*Consensus method*), 19
 get_block_count() (*BlockStore method*), 3
 get_blockhash() (*Consensus method*), 20
 get_blockheader() (*Node method*), 42
 get_bytes() (*ExtKey method*), 126
 get_bytes() (*Key method*), 127
 get_connectedpeers_info() (*Diagnostic method*), 21
 get_connection_info() (*SignalR method*), 50
 get_last_balance_update_transaction() (*BlockStore method*), 4
 get_payload() (*ExtPubKey method*), 127
 get_peer_statistics() (*Diagnostic method*), 22
 get_raw_mempool() (*Mempool method*), 39
 get_raw_transaction() (*Node method*), 42
 get_staking_info() (*Staking method*), 64
 get_status() (*Diagnostic method*), 21
 get_txout() (*Node method*), 43
 get_txout_proof() (*Node method*), 44
 get_utxo_set() (*BlockStore method*), 4
 get_utxoset_for_address() (*BlockStore method*), 4
 get_verbose_addresses_balances() (*BlockStore method*), 3
 GetAddressesBalancesModel (class in `pys-tratis.api.blockstore.responsemodels.getaddressesbalancesmodel`), 8
 GetCodeModel (class in `pys-tratis.api.smartcontracts.responsemodels.getcodemodel`), 58
 GetConnectedPeersInfoModel (class in `pys-tratis.api.diagnostic.responsemodels.getconnectedpeersinfo`), 23
 GetConnectionInfoModel (class in `pys-tratis.api.signalr.responsemodels.getconnectioninfo`), 50
 GetLastBalanceUpdateTransactionModel (class in `pys-tratis.api.blockstore.responsemodels.getlastbalanceupdatetransactionmodel`), 8
 getpeerinfo() (*ConnectionManager method*), 18
 GetStakingInfoModel (class in `pys-tratis.api.staking.responsemodels.getstakinginfomodel`), 65
 GetStatusModel (class in `pys-tratis.api.diagnostic.responsemodels.getstatusmodel`), 23
 GetTxOutModel (class in `pys-tratis.api.node.responsemodels.gettxoutmodel`), 46
 GetUTXOsForAddressModel (class in `pys-tratis.api.blockstore.responsemodels.getutxosforaddressmodel`), 8
 GetVerboseAddressesBalancesModel (class in `pys-tratis.api.blockstore.responsemodels.getverboseaddressesbalancesmodel`),
- 9
H
 hash (*BlockModel attribute*), 91
 hash (*BlockTransactionDetailsModel attribute*), 93
 hash (*ContractTransactionItemModel attribute*), 63
 hash (*TransactionModel attribute*), 98
 hash (*VotingDataModel attribute*), 68
 hash (*WhitelistedHashesModel attribute*), 69
 hashproof (*BlockModel attribute*), 92
 hashproof (*BlockTransactionDetailsModel attribute*), 94
 header_height (*StatusModel attribute*), 47
 height (*BlockModel attribute*), 91
 height (*BlockTransactionDetailsModel attribute*), 93
 height (*DeploymentFlagsModel attribute*), 20
 hex (*BuildContractTransactionModel attribute*), 94
 hex (*BuildCreateContractTransactionModel attribute*), 94
 hex (*BuildTransactionModel attribute*), 95
 hex (*ScriptPubKey attribute*), 97
 hex (*ScriptSig attribute*), 97
 hex (*TransactionModel attribute*), 98
 hex_to_int() (*int32 class method*), 120
 hex_to_int() (*int64 class method*), 121
 hex_to_int() (*uint128 class method*), 122
 hex_to_int() (*uint160 class method*), 122
 hex_to_int() (*uint256 class method*), 122
 hex_to_int() (*uint32 class method*), 122
 hex_to_int() (*uint64 class method*), 123
 hexstr (class in `pystratis.core.types.hexstr`), 120
 history (*WalletHistoryModel attribute*), 88
 history() (*FederationWallet method*), 32
 history() (*SmartContractWallet method*), 61
 history() (*Wallet method*), 71
 hot_wallet_account_exists (*InfoModel attribute*), 16
 immature (*GetStakingInfoModel attribute*), 65
 in_ibd (*StatusModel attribute*), 47
 inbound (*PeerInfoModel attribute*), 19
 inbound (*PeerStatisticsModel attribute*), 23
 inbound_peers (*StatusModel attribute*), 47
 index (*ExtPubKey attribute*), 127
 index (*Outpoint attribute*), 128
 index (*SpendableTransactionModel attribute*), 85
 index (*UtxoDescriptor attribute*), 99
 index (*UTXOModel attribute*), 9
 inflight (*PeerInfoModel attribute*), 19
 info() (*InfoModel method*), 9
 info() (*FederationGateway method*), 27

- InfoModel (class in `pys-tratis.api.coldstaking.responsemodels.infomodel`), 16
- Initialized (*FeatureInitializationState* attribute), 101
- Initialized (*FullNodeState* attribute), 101
- Initializing (*FeatureInitializationState* attribute), 101
- Initializing (*FullNodeState* attribute), 101
- int32 (class in `pys-tratis.core.types.int32`), 120
- Int32 (*SmartContractParameterType* attribute), 129
- int64 (class in `pys-tratis.core.types.int64`), 121
- Int64 (*SmartContractParameterType* attribute), 129
- InterfluxCirrusNode (class in `pys-tratis.nodes.interfluxnodes`), 109
- InterfluxStraxNode (class in `pys-tratis.nodes.interfluxnodes`), 107
- internal_transfers (*LocalExecutionResultModel* attribute), 58
- Interop (class in `pys-tratis.api.interop`), 34
- interop (*InterfluxCirrusNode* property), 110
- ip_add() (*FederationGateway* method), 27
- ip_remove() (*FederationGateway* method), 27
- ip_replace() (*FederationGateway* method), 27
- ipaddr (*CirrusMasterNode* property), 117
- ipaddr (*CirrusNode* property), 106
- ipaddr (*InterfluxCirrusNode* property), 112
- ipaddr (*InterfluxStraxNode* property), 108
- ipaddr (*StraxMasterNode* property), 114
- ipaddr (*StraxNode* property), 103
- is_banned (*FederationMemberConnectionInfoModel* attribute), 30
- is_chain_synced (*WalletGeneralInfoModel* attribute), 100
- is_change (*AddressModel* attribute), 91
- is_change (*PaymentDetailModel* attribute), 85
- is_change (*SpendableTransactionModel* attribute), 85
- is_connected (*ConnectedPeerInfoModel* attribute), 22
- is_connected (*FederationMemberConnectionInfoModel* attribute), 30
- is_decrypted (*WalletGeneralInfoModel* attribute), 100
- is_executed (*PollViewModel* attribute), 96
- is_pending (*PollViewModel* attribute), 96
- is_used (*AddressModel* attribute), 91
- is_valid (*ValidateTransactionResultModel* attribute), 31
- isscript (*ValidateAddressModel* attribute), 49
- isvalid (*ValidateAddressModel* attribute), 48
- iswitness (*ValidateAddressModel* attribute), 49
- item_type (*ContractTransactionItemModel* attribute), 63
- ## J
- join_federation() (*Collateral* method), 16
- JoinFederationResponseModel (class in `pys-tratis.api.collateral.responsemodels.joinfederationresponsemodel`), 17
- ## K
- Key (class in `pys-tratis.core.key`), 127
- key (*ExtPubKey* attribute), 127
- key (*VotingDataModel* attribute), 68
- key_path (*AddressDescriptor* attribute), 90
- ## L
- label (*AddressBookEntryModel* attribute), 2
- Large (*DepositRetrievalType* attribute), 125
- last_active_time (*FederationMemberDetailedModel* attribute), 25
- last_active_time (*FederationMemberModel* attribute), 26
- last_block_synced_height (*WalletGeneralInfoModel* attribute), 100
- lastrecv (*PeerInfoModel* attribute), 18
- lastsend (*PeerInfoModel* attribute), 18
- latest_events (*PeerStatisticsModel* attribute), 23
- list_methods() (*RPC* method), 49
- list_wallets() (*Wallet* method), 76
- load() (*Wallet* method), 70
- local_call() (*SmartContracts* method), 56
- LocalExecutionResultModel (class in `pys-tratis.api.smartcontracts.responsemodels.localexecutionresultmodel`), 58
- locktime (*TransactionModel* attribute), 98
- log (*LogModel* attribute), 59
- log_level (*LogRule* attribute), 101
- log_levels() (*Node* method), 44
- log_rules() (*Node* method), 44
- LogModel (class in `pys-tratis.api.smartcontracts.responsemodels.logmodel`), 59
- LogRule (class in `pys-tratis.api.logrule`), 101
- logs (*LocalExecutionResultModel* attribute), 59
- logs (*ReceiptModel* attribute), 60
- loop_name (*AsyncLoopsModel* attribute), 45
- ## M
- mainchain (*FederationGatewayInfoModel* attribute), 29
- MaturedBlockDepositsModel (class in `pys-tratis.api.federationgateway.responsemodels.maturedblockdepositsmodel`), 31
- MaturedBlockInfoModel (class in `pys-tratis.api.global_responsemodels.maturedblockinfomodel`), 96
- max_balance() (*Wallet* method), 73
- max_spendable_amount (*MaxSpendableAmountModel* attribute), 84
- MaxSpendableAmountModel (class in `pys-tratis.api.wallet.responsemodels.maxspendableamountmodel`), 84

- median_time (*BlockModel* attribute), 92
 median_time (*BlockTransactionDetailsModel* attribute), 93
 member_info() (*FederationGateway* method), 27
 member_will_start_earning_rewards_estimate_height (*FederationMemberDetailedModel* attribute), 25
 member_will_start_mining_at_block_height (*FederationMemberDetailedModel* attribute), 25
 members() (*Federation* method), 24
 members_current() (*Federation* method), 24
 mempool (*CirrusMasterNode* property), 117
 mempool (*CirrusNode* property), 106
 Mempool (class in *pystratis.api.mempool*), 39
 mempool (*InterfluxCirrusNode* property), 112
 mempool (*InterfluxStraxNode* property), 109
 mempool (*StraxMasterNode* property), 114
 mempool (*StraxNode* property), 103
 merkleroot (*BlockHeaderModel* attribute), 45
 merkleroot (*BlockModel* attribute), 92
 merkleroot (*BlockTransactionDetailsModel* attribute), 93
 message (*APIError* attribute), 102
 message (*BuildContractTransactionModel* attribute), 94
 message (*BuildCreateContractTransactionModel* attribute), 95
 message (*GetCodeModel* attribute), 58
 message (*SerializableResult* attribute), 31
 min_conf_distribution_deposits (*FederationGatewayInfoModel* attribute), 29
 min_conf_large_deposits (*FederationGatewayInfoModel* attribute), 29
 min_conf_normal_deposits (*FederationGatewayInfoModel* attribute), 29
 min_conf_small_deposits (*FederationGatewayInfoModel* attribute), 29
 min_confirmations (*DistributeUtxoModel* attribute), 84
 Mined (*TransactionItemType* attribute), 130
 miner_at_height() (*Federation* method), 24
 miner_public_key (*JoinFederationResponseModel* attribute), 17
 Mining (class in *pystratis.api.mining*), 39
 mining (*InterfluxStraxNode* property), 107
 mining (*StraxNode* property), 102
 mining_pubkey (*FederationGatewayInfoModel* attribute), 29
 mining_stats (*FederationMemberDetailedModel* attribute), 25
 minping (*PeerInfoModel* attribute), 18
 Mint (*ConversionRequestType* attribute), 124
 mnemonic (*MultisigSecret* attribute), 128
 mnemonic() (*Wallet* method), 69
 modifier_v2 (*BlockModel* attribute), 92
 modifier_v2 (*BlockTransactionDetailsModel* attribute), 94
 module
 pystratis.api.addressbook.responsemodels.addressbookentry, 2
 pystratis.api.apierror, 102
 pystratis.api.blockstore.responsemodels.addressbalance, 4
 pystratis.api.blockstore.responsemodels.addressindexer, 5
 pystratis.api.blockstore.responsemodels.balancechanges, 5
 pystratis.api.blockstore.responsemodels.getaddressesbalance, 8
 pystratis.api.blockstore.responsemodels.getlastbalance, 8
 pystratis.api.blockstore.responsemodels.getutxosforaddress, 8
 pystratis.api.blockstore.responsemodels.getverboseaddress, 9
 pystratis.api.blockstore.responsemodels.utxomodel, 9
 pystratis.api.blockstore.responsemodels.verboseaddress, 9
 pystratis.api.coldstaking.responsemodels.accountmodel, 15
 pystratis.api.coldstaking.responsemodels.addressmodel, 15
 pystratis.api.coldstaking.responsemodels.infomodel, 16
 pystratis.api.coldstaking.responsemodels.setupmodel, 16
 pystratis.api.coldstaking.responsemodels.withdrawalmodel, 16
 pystratis.api.collateral.responsemodels.joinfederation, 17
 pystratis.api.connectionmanager.responsemodels.peerinfo, 18
 pystratis.api.consensus.responsemodels.deploymentflags, 20
 pystratis.api.diagnostic.responsemodels.connectedpeerinfo, 22
 pystratis.api.diagnostic.responsemodels.getconnectedpeerinfo, 23
 pystratis.api.diagnostic.responsemodels.getstatusmodel, 23
 pystratis.api.diagnostic.responsemodels.peerstatistics, 23
 pystratis.api.featureinitializationstate, 101
 pystratis.api.federation.responsemodels.federationmember, 25
 pystratis.api.federation.responsemodels.federationmember, 25

26	pystratis.api.global_responsemodels.utxodescriptor,
pystratis.api.federationgateway.responsemodels.crosschaintransfermodel,	98
28	pystratis.api.global_responsemodels.vin,
pystratis.api.federationgateway.responsemodels.federationgatewayinfomodel,	100
29	pystratis.api.global_responsemodels.vout,
pystratis.api.federationgateway.responsemodels.federationmemberconnectioninfomodel,	100
30	pystratis.api.global_responsemodels.walletbalancemodel,
pystratis.api.federationgateway.responsemodels.federationmemberinfomodel,	100
30	pystratis.api.global_responsemodels.walletgeneralinfomodel,
pystratis.api.federationgateway.responsemodels.maturedblockdepositsmodel,	100
31	pystratis.api.global_responsemodels.walletsendtransactionmodel,
pystratis.api.federationgateway.responsemodels.serializableresult,	101
31	pystratis.api.interop.responsemodels.conversionrequestmodel,
pystratis.api.federationgateway.responsemodels.validdatetransactionresultmodel,	101
31	pystratis.api.interop.responsemodels.transactionresponsemodel,
pystratis.api.federationwallet.responsemodels.withdrawalmodel,	101
34	pystratis.api.logrule, 101
pystratis.api.fullnodestate, 101	pystratis.api.mining.responsemodels.generateblocksmode
pystratis.api.global_responsemodels.accountbalancemodel,	101
89	pystratis.api.network.responsemodels.bannedpeermodel,
pystratis.api.global_responsemodels.addressbalancemodel,	101
90	pystratis.api.node.responsemodels.asyncloopsmodel,
pystratis.api.global_responsemodels.addressdescriptor,	101
90	pystratis.api.node.responsemodels.blockheadermodel,
pystratis.api.global_responsemodels.addressesmodel,	101
91	pystratis.api.node.responsemodels.connectedpeermodel,
pystratis.api.global_responsemodels.addressmodel, 46	pystratis.api.node.responsemodels.featuresdatamodel,
91	pystratis.api.node.responsemodels.gettxoutmodel,
pystratis.api.global_responsemodels.blockmodel, 46	pystratis.api.node.responsemodels.statusmodel,
91	pystratis.api.node.responsemodels.validateaddressmodel,
pystratis.api.global_responsemodels.blocktransactiondetailsmodel,	101
93	pystratis.api.node.responsemodels.transactionmodel,
pystratis.api.global_responsemodels.buildcontracttransactionmodel,	101
94	pystratis.api.node.responsemodels.validateaddressmodel,
pystratis.api.global_responsemodels.buildcreatecontracttransactionmodel,	101
94	pystratis.api.rpc.responsemodels.rpccommandlistmodel,
pystratis.api.global_responsemodels.buildofflineignmodel,	101
95	pystratis.api.rpc.responsemodels.rpccommandresponsemodel,
pystratis.api.global_responsemodels.buildtransactionmodel,	101
95	pystratis.api.signalr.responsemodels.getconnectioninfo
pystratis.api.global_responsemodels.maturedblockinfomodel,	101
96	pystratis.api.smartcontracts.responsemodels.addressbal
pystratis.api.global_responsemodels.pollviewmodel, 57	pystratis.api.smartcontracts.responsemodels.getcodemod
96	pystratis.api.smartcontracts.responsemodels.getcodemod
pystratis.api.global_responsemodels.removedtransactionmodel,	101
97	pystratis.api.smartcontracts.responsemodels.localexecu
pystratis.api.global_responsemodels.scriptpubkey, 58	pystratis.api.smartcontracts.responsemodels.logmodel,
97	pystratis.api.smartcontracts.responsemodels.receiptmod
pystratis.api.global_responsemodels.scriptsig, 59	pystratis.api.smartcontracts.responsemodels.transferin
97	pystratis.api.smartcontracts.responsemodels.transferin
pystratis.api.global_responsemodels.transactionmodel,	101
98	pystratis.api.smartcontractwallet.responsemodels.contr
pystratis.api.global_responsemodels.transactionoutputmodel,	101
98	pystratis.api.smartcontractwallet.responsemodels.contr

63
 pystratis.api.staking.responsemodels.getstakinginfo, 120
 65
 pystratis.api.voting.responsemodels.votingdata, 121
 68
 pystratis.api.voting.responsemodels.whitelisted, 122
 69
 pystratis.api.wallet.responsemodels.accounthistory, 122
 82
 pystratis.api.wallet.responsemodels.distributed, 123
 84
 pystratis.api.wallet.responsemodels.maxspendable, 121
 84
 pystratis.api.wallet.responsemodels.paymentdetails, 110
 85
 pystratis.api.wallet.responsemodels.spendable, 121
 85
 pystratis.api.wallet.responsemodels.spendable, 121
 86
 pystratis.api.wallet.responsemodels.transactionitem, 129
 86
 pystratis.api.wallet.responsemodels.utxoamount, 129
 87
 pystratis.api.wallet.responsemodels.utxoperblock, 129
 87
 pystratis.api.wallet.responsemodels.utxoperblock, 128
 87
 pystratis.api.wallet.responsemodels.wallethistory, 128
 88
 pystratis.api.wallet.responsemodels.walletstatus, 99
 88
 pystratis.core.cointype, 123
 pystratis.core.contracttransactionitem, 117
 123
 pystratis.core.conversionrequeststatus, 112
 124
 pystratis.core.conversionrequesttype, 114
 124
 pystratis.core.crosschaintransferstatus, 104
 124
 pystratis.core.deposit, 124
 pystratis.core.depositretrievaltype, 125
 pystratis.core.destinationchain, 126
 pystratis.core.extkey, 126
 pystratis.core.extpubkey, 127
 pystratis.core.key, 127
 pystratis.core.multisigsecret, 128
 pystratis.core.outpoint, 128
 pystratis.core.pubkey, 128
 pystratis.core.recipient, 129
 pystratis.core.smartcontractparameter, 100
 129
 pystratis.core.smartcontractparameter, 94
 129
 pystratis.core.transactionitem, 130
 pystratis.core.types.address, 120
 pystratis.core.types.hexstr, 120
 pystratis.core.types.int32, 120
 pystratis.core.types.int64, 121
 pystratis.core.types.money, 121
 pystratis.core.types.uint128, 122
 pystratis.core.types.uint160, 122
 pystratis.core.types.uint256, 122
 pystratis.core.types.uint32, 122
 pystratis.core.types.uint64, 123
 pystratis.core.walletsecret, 130
 pystratis.core.walletsecret, 121
 Multisig (class in pystratis.api.multisig), 40
 multisig (InterfluxCirrusNode property), 110
 multisig (InterfluxStraxNode property), 107
 multisig (FederationGatewayInfoModel attribute), 29
 multisig_publickey (FederationGatewayInfoModel attribute), 29
 multisig_redeem_script (FederationGatewayInfoModel attribute), 29
 multisig_redeem_script_payment_script (FederationGatewayInfoModel attribute), 29
 multisig_transaction() (Interop method), 36
 multisigsecret, 128
 N
 n_tx (BlockModel attribute), 92
 n_tx (BlockTransactionDetailsModel attribute), 93
 name (CirrusMasterNode property), 117
 name (CirrusNode property), 106
 name (InterfluxCirrusNode property), 112
 name (InterfluxStraxNode property), 109
 name (StraxMasterNode property), 114
 name (StraxNode property), 104
 namespace (FeaturesDataModel attribute), 46
 net_stake_weight (GetStakingInfoModel attribute), 65
 network (Address attribute), 120
 network (CirrusMasterNode property), 117
 network (CirrusNode property), 106
 Network (class in pystratis.api.network), 41
 network (InterfluxCirrusNode property), 112
 network (InterfluxStraxNode property), 109
 network (StatusModel attribute), 47
 network (StraxMasterNode property), 114
 network (StraxNode property), 104
 network (WalletGeneralInfoModel attribute), 100
 new_addresses() (Wallet method), 77
 new_contract_address (BuildCreateContractTransactionModel attribute), 94
 new_contract_address (ReceiptModel attribute), 59

- next_blockhash (*BlockModel* attribute), 92
- next_blockhash (*BlockTransactionDetailsModel* attribute), 94
- node (*CirrusMasterNode* property), 117
- node (*CirrusNode* property), 106
- Node (class in *pystratis.api.node*), 42
- node (*InterfluxCirrusNode* property), 112
- node (*InterfluxStraxNode* property), 109
- node (*StraxMasterNode* property), 114
- node (*StraxNode* property), 104
- node_version (*FederationMemberInfoModel* attribute), 30
- nonce (*BlockHeaderModel* attribute), 45
- nonce (*BlockModel* attribute), 92
- nonce (*BlockTransactionDetailsModel* attribute), 93
- Normal (*DepositRetrievalType* attribute), 125
- notifications (*CirrusMasterNode* property), 117
- Notifications (class in *pystratis.api.notifications*), 49
- notifications (*InterfluxCirrusNode* property), 111
- notifications (*InterfluxStraxNode* property), 107
- notifications (*StraxMasterNode* property), 114
- NotOriginator (*ConversionRequestStatus* attribute), 124
- ## O
- offline_sign_request() (*Wallet* method), 81
- offline_withdrawal() (*ColdStaking* method), 13
- op_return_data (*TransactionOutputModel* attribute), 99
- OriginatorNotSubmitted (*ConversionRequestStatus* attribute), 124
- OriginatorSubmitted (*ConversionRequestStatus* attribute), 124
- outbound_peers (*StatusModel* attribute), 47
- Outpoint (class in *pystratis.core.outpoint*), 128
- outputs (*WalletSendTransactionModel* attribute), 100
- over_amount_at_height() (*Balances* method), 2
- owners() (*Interop* method), 35
- ## P
- parent_fingerprint (*ExtPubKey* attribute), 127
- Partial (*CrossChainTransferStatus* attribute), 124
- passphrase (*MultisigSecret* attribute), 128
- paying_to (*WithdrawalModel* attribute), 34
- PaymentDetailModel (class in *pystratis.api.wallet.responsemodels.paymentdetailmodel*), 85
- payments (*TransactionItemModel* attribute), 86
- peer_endpoint (*PeerStatisticsModel* attribute), 23
- peer_id (*PeerInfoModel* attribute), 18
- peer_statistics (*GetStatusModel* attribute), 23
- PeerInfoModel (class in *pystratis.api.connectionmanager.responsemodels.peerinfomodel*), 18
- peers_by_peer_id (*GetConnectedPeersInfoModel* attribute), 23
- PeerStatisticsModel (class in *pystratis.api.diagnostic.responsemodels.peerstatisticsmodel*), 23
- pending_polls() (*Voting* method), 66
- pending_transfer() (*FederationGateway* method), 26
- period_end_height (*DeploymentFlagsModel* attribute), 20
- period_of_inactivity (*FederationMemberDetailedModel* attribute), 25
- period_of_inactivity (*FederationMemberModel* attribute), 26
- period_start_height (*DeploymentFlagsModel* attribute), 20
- pingtime (*PeerInfoModel* attribute), 18
- pingwait (*PeerInfoModel* attribute), 18
- poll_executed_block_height (*FederationMemberDetailedModel* attribute), 25
- poll_executed_blockdata_hash (*PollViewModel* attribute), 96
- poll_executed_blockdata_height (*PollViewModel* attribute), 96
- poll_finished_block_height (*FederationMemberDetailedModel* attribute), 25
- poll_id (*PollViewModel* attribute), 96
- poll_number_of_votes_acquired (*FederationMemberDetailedModel* attribute), 25
- poll_start_block_height (*FederationMemberDetailedModel* attribute), 25
- poll_start_favor_blockdata_hash (*PollViewModel* attribute), 96
- poll_start_favor_blockdata_height (*PollViewModel* attribute), 96
- poll_type (*FederationMemberDetailedModel* attribute), 25
- poll_voted_in_favor_blockdata_hash (*PollViewModel* attribute), 96
- poll_voted_in_favor_blockdata_height (*PollViewModel* attribute), 96
- poll_will_finish_in_blocks (*FederationMemberDetailedModel* attribute), 25
- polls_tip() (*Voting* method), 67
- PollViewModel (class in *pystratis.api.global_responsemodels.pollviewmodel*), 96
- pooled_tx (*GetStakingInfoModel* attribute), 65
- post_state (*ReceiptModel* attribute), 59
- previous_blockhash (*BlockHeaderModel* attribute), 45
- previous_blockhash (*BlockModel* attribute), 92
- previous_blockhash (*BlockTransactionDetailsModel* attribute), 93
- private_key() (*Wallet* method), 78

process_id (*StatusModel* attribute), 47
 processed (*ConversionRequestModel* attribute), 39
 Processed (*ConversionRequestStatus* attribute), 124
 protocol_version (*StatusModel* attribute), 47
 PubKey (*class* in *pystratis.core.pubkey*), 128
 pubkey (*FederationMemberDetailedModel* attribute), 25
 pubkey (*FederationMemberInfoModel* attribute), 30
 pubkey (*FederationMemberModel* attribute), 26
 pubkey() (*Wallet* method), 70
 pubkeys_hex_voted_in_favor (*PollViewModel* attribute), 96
 pystratis.api.addressbook.responsemodels.addressbookentrymodel module, 2
 pystratis.api.apierror module, 102
 pystratis.api.blockstore.responsemodels.addressbookentrymodel module, 4
 pystratis.api.blockstore.responsemodels.addressbookentrymodel module, 5
 pystratis.api.blockstore.responsemodels.balancechange model, 5
 pystratis.api.blockstore.responsemodels.getaddressbalance model, 8
 pystratis.api.blockstore.responsemodels.getlastupdateofaddressbalance model, 8
 pystratis.api.blockstore.responsemodels.getutxoaddress model, 8
 pystratis.api.blockstore.responsemodels.getverboaddress model, 9
 pystratis.api.blockstore.responsemodels.utxomodel module, 9
 pystratis.api.blockstore.responsemodels.verboseaddress model, 9
 pystratis.api.coldstaking.responsemodels.account model, 15
 pystratis.api.coldstaking.responsemodels.address model, 15
 pystratis.api.coldstaking.responsemodels.infomodel module, 16
 pystratis.api.coldstaking.responsemodels.setup model, 16
 pystratis.api.coldstaking.responsemodels.withdrawal model, 16
 pystratis.api.collateral.responsemodels.joinfees model, 17
 pystratis.api.connectionmanager.responsemodels.pystratisinfo model, 18
 pystratis.api.consensus.responsemodels.deploy model, 20
 pystratis.api.diagnostic.responsemodels.connection model, 22
 pystratis.api.diagnostic.responsemodels.getcompleterapi model, 23
 pystratis.api.diagnostic.responsemodels.getstatus model, 23
 pystratis.api.diagnostic.responsemodels.peerstatisticsmodel module, 23
 pystratis.api.featureinitializationstate module, 101
 pystratis.api.federation.responsemodels.federationmember model, 25
 pystratis.api.federation.responsemodels.federationmember model, 26
 pystratis.api.federationgateway.responsemodels.crosschain model, 28
 pystratis.api.federationgateway.responsemodels.federation model, 29
 pystratis.api.federationgateway.responsemodels.federation model, 30
 pystratis.api.federationgateway.responsemodels.federation model, 30
 pystratis.api.federationgateway.responsemodels.maturedblock model, 31
 pystratis.api.federationgateway.responsemodels.serializable model, 31
 pystratis.api.federationgateway.responsemodels.validate transaction model, 31
 pystratis.api.federationgateway.responsemodels.withdrawal model, 34
 pystratis.api.global_responsemodels.accountbalancemodel module, 89
 pystratis.api.global_responsemodels.addressbalancemodel module, 90
 pystratis.api.global_responsemodels.addressdescriptor module, 90
 pystratis.api.global_responsemodels.addressesmodel module, 91
 pystratis.api.global_responsemodels.addressmodel module, 91
 pystratis.api.global_responsemodels.blockmodel module, 91
 pystratis.api.global_responsemodels.blocktransactiondetail model, 93
 pystratis.api.global_responsemodels.buildcontracttransaction model, 94
 pystratis.api.global_responsemodels.buildcreatecontracttransaction model, 94
 pystratis.api.global_responsemodels.buildofflinesignmodel module, 95
 pystratis.api.global_responsemodels.buildtransactionmodel module, 95
 pystratis.api.global_responsemodels.maturedblockinfomodel module, 96
 pystratis.api.global_responsemodels.pollviewmodel module, 96
 pystratis.api.global_responsemodels.removedtransactionmodel module, 96

module, 97
 pystratis.api.global_responsemodels.scriptpublickey module, 97
 pystratis.api.global_responsemodels.scriptsig module, 97
 pystratis.api.global_responsemodels.transactionmodel module, 98
 pystratis.api.global_responsemodels.transactionpostinfo module, 98
 pystratis.api.global_responsemodels.utxodescriptor module, 99
 pystratis.api.global_responsemodels.vin module, 99
 pystratis.api.global_responsemodels.vout module, 99
 pystratis.api.global_responsemodels.walletbalances module, 100
 pystratis.api.global_responsemodels.walletgenerations module, 100
 pystratis.api.global_responsemodels.walletsendtransaction module, 100
 pystratis.api.interop.responsemodels.conversion module, 38
 pystratis.api.interop.responsemodels.transaction module, 39
 pystratis.api.logrule module, 101
 pystratis.api.mining.responsemodels.generateblock module, 40
 pystratis.api.network.responsemodels.bannedpeer module, 42
 pystratis.api.node.responsemodels.asyncloops module, 45
 pystratis.api.node.responsemodels.blockheader module, 45
 pystratis.api.node.responsemodels.connectedpeer module, 46
 pystratis.api.node.responsemodels.featuresdata module, 46
 pystratis.api.node.responsemodels.gettxout module, 46
 pystratis.api.node.responsemodels.status module, 46
 pystratis.api.node.responsemodels.validateaddress module, 48
 pystratis.api.rpc.responsemodels.rpccommandlist module, 50
 pystratis.api.rpc.responsemodels.rpccommandresponse module, 50
 pystratis.api.signalr.responsemodels.getconnection module, 50
 pystratis.api.smartcontracts.responsemodels.address module, 57
 pystratis.api.smartcontracts.responsemodels.gettransaction module, 58
 pystratis.api.smartcontracts.responsemodels.localexecution module, 58
 pystratis.api.smartcontracts.responsemodels.logmodel module, 59
 pystratis.api.smartcontracts.responsemodels.receiptmodel module, 59
 pystratis.api.smartcontracts.responsemodels.transferinfo module, 60
 pystratis.api.smartcontractwallet.responsemodels.contract module, 63
 pystratis.api.staking.responsemodels.getstakinginfo module, 65
 pystratis.api.voting.responsemodels.votingdata module, 68
 pystratis.api.voting.responsemodels.whitelistedhashes module, 69
 pystratis.api.wallet.responsemodels.accounthistory module, 82
 pystratis.api.wallet.responsemodels.distributeutxo module, 84
 pystratis.api.wallet.responsemodels.maxspendableamount module, 84
 pystratis.api.wallet.responsemodels.paymentdetail module, 85
 pystratis.api.wallet.responsemodels.spendabletransaction module, 85
 pystratis.api.wallet.responsemodels.spendabletransactions module, 86
 pystratis.api.wallet.responsemodels.transactionitem module, 86
 pystratis.api.wallet.responsemodels.utxoamount module, 87
 pystratis.api.wallet.responsemodels.utxoperblock module, 87
 pystratis.api.wallet.responsemodels.utxopertransaction module, 87
 pystratis.api.wallet.responsemodels.wallethistory module, 88
 pystratis.api.wallet.responsemodels.walletstats module, 88
 pystratis.core.cointype module, 123
 pystratis.core.contracttransactionitemtype module, 123
 pystratis.core.conversionrequeststatus module, 124
 pystratis.core.conversionrequesttype module, 124
 pystratis.core.crosschaintransferstatus module, 124
 pystratis.core.deposit module, 124
 pystratis.core.depositretrievaltype module, 124

- module, 125
 - pystratis.core.destinationchain
 - module, 126
 - pystratis.core.extkey
 - module, 126
 - pystratis.core.extpubkey
 - module, 127
 - pystratis.core.key
 - module, 127
 - pystratis.core.multisigsecret
 - module, 128
 - pystratis.core.outpoint
 - module, 128
 - pystratis.core.pubkey
 - module, 128
 - pystratis.core.recipient
 - module, 129
 - pystratis.core.smartcontractparameter
 - module, 129
 - pystratis.core.smartcontractparametertype
 - module, 129
 - pystratis.core.transactionitemtype
 - module, 130
 - pystratis.core.types.address
 - module, 120
 - pystratis.core.types.hexstr
 - module, 120
 - pystratis.core.types.int32
 - module, 120
 - pystratis.core.types.int64
 - module, 121
 - pystratis.core.types.money
 - module, 121
 - pystratis.core.types.uint128
 - module, 122
 - pystratis.core.types.uint160
 - module, 122
 - pystratis.core.types.uint256
 - module, 122
 - pystratis.core.types.uint32
 - module, 122
 - pystratis.core.types.uint64
 - module, 123
 - pystratis.core.walletsecret
 - module, 130
- R**
- reason (*GetAddressesBalancesModel* attribute), 8
 - reason (*GetVerboseAddressesBalancesModel* attribute), 9
 - receipt() (*SmartContracts* method), 51
 - receipt_search() (*SmartContracts* method), 52
 - ReceiptModel (class in *pys-tratis.api.smartcontracts.responsemodels.receiptmodel*), 59
 - Received (*ContractTransactionItemType* attribute), 123
 - Received (*TransactionItemType* attribute), 130
 - received_by_address() (*Wallet* method), 72
 - received_messages (*PeerStatisticsModel* attribute), 23
 - Recipient (class in *pystratis.core.recipient*), 129
 - reconstruct() (*Federation* method), 24
 - recover() (*Wallet* method), 70
 - recover_via_extpubkey() (*Wallet* method), 71
 - Rejected (*CrossChainTransferStatus* attribute), 124
 - relay_fee (*StatusModel* attribute), 47
 - relaytxes (*PeerInfoModel* attribute), 18
 - remote_socket_endpoint (*ConnectedPeerModel* attribute), 46
 - remove() (*AddressBook* method), 1
 - remove_owner() (*Interop* method), 35
 - remove_transactions() (*FederationWallet* method), 32
 - remove_transactions() (*Wallet* method), 77
 - remove_wallet() (*Wallet* method), 78
 - RemovedTransactionModel (class in *pys-tratis.api.global_responsemodels.removedtransactionmodel*), 97
 - req_sigs (*ScriptPubKey* attribute), 97
 - request_id (*ConversionRequestModel* attribute), 38
 - request_status (*ConversionRequestModel* attribute), 38
 - request_type (*ConversionRequestModel* attribute), 38
 - requests_delete() (*Interop* method), 37
 - requests_pushvote() (*Interop* method), 38
 - requests_reprocess_burn() (*Interop* method), 38
 - requests_setnotoriginator() (*Interop* method), 37
 - requests_setoriginator() (*Interop* method), 37
 - retrieval_type (*Deposit* attribute), 125
 - retrieve_filtered_utxos() (*ColdStaking* method), 14
 - return_obj (*LocalExecutionResultModel* attribute), 59
 - return_value (*ReceiptModel* attribute), 60
 - revert (*LocalExecutionResultModel* attribute), 58
 - reward_estimate_per_block (*FederationMemberDetailedModel* attribute), 25
 - rewind() (*Node* method), 45
 - rpc (*CirrusMasterNode* property), 117
 - rpc (*CirrusNode* property), 106
 - RPC (class in *pystratis.api.rpc*), 49
 - rpc (*InterfluxCirrusNode* property), 112
 - rpc (*InterfluxStraxNode* property), 109
 - rpc (*StraxMasterNode* property), 114
 - rpc (*StraxNode* property), 104
 - RPCCommandListModel (class in *pys-tratis.api.rpc.responsemodels.rpccommandlistmodel*), 50
 - RPCCommandResponseModel (class in *pys-tratis.api.rpc.responsemodels.rpccommandresponsemodel*), 50

- 50
- rule_name (*LogRule* attribute), 101
- running_time (*StatusModel* attribute), 47
- ## S
- satoshi (*BalanceChangesModel* attribute), 5
- scheduled_votes() (*Voting* method), 67
- schedulevote_kickfedmember() (*CollateralVoting* method), 17
- schedulevote_kickmember() (*Voting* method), 67
- schedulevote_removehash() (*Voting* method), 67
- schedulevote_whitelistash() (*Voting* method), 67
- script_pubkey (*GetTxOutModel* attribute), 46
- script_pubkey (*UtxoDescriptor* attribute), 99
- script_pubkey (*UTXOModel* attribute), 9
- script_pubkey (*VOut* attribute), 100
- script_sig (*VIn* attribute), 99
- ScriptPubKey (class in *pys-tratis.api.global_responsemodels.scriptpubkey*), 97
- scriptPubKey (*ValidateAddressModel* attribute), 48
- ScriptSig (class in *pys-tratis.api.global_responsemodels.scriptsig*), 97
- search_interval (*GetStakingInfoModel* attribute), 65
- SeenInBlock (*CrossChainTransferStatus* attribute), 124
- Send (*ContractTransactionItemType* attribute), 123
- Send (*TransactionItemType* attribute), 130
- send_messages (*PeerStatisticsModel* attribute), 23
- send_transaction() (*SmartContractWallet* method), 62
- send_transaction() (*Wallet* method), 75
- sequence (*VIn* attribute), 99
- SerializableResult (class in *pys-tratis.api.federationgateway.responsemodels.serializableresult*), 31
- services (*PeerInfoModel* attribute), 18
- set_ban() (*Network* method), 41
- setup() (*ColdStaking* method), 10
- setup_offline() (*ColdStaking* method), 11
- SetupModel (class in *pys-tratis.api.coldstaking.responsemodels.setupmodel*), 16
- shutdown() (*Node* method), 44
- sign_message() (*Wallet* method), 69
- signalr (*CirrusMasterNode* property), 117
- signalr (*CirrusNode* property), 105
- SignalR (class in *pystratis.api.signalr*), 50
- signalr (*StraxMasterNode* property), 115
- signalr (*StraxNode* property), 102
- signalr_port (*GetConnectionInfoModel* attribute), 50
- signalr_uri (*GetConnectionInfoModel* attribute), 50
- signature (*BlockModel* attribute), 92
- signature (*BlockTransactionDetailsModel* attribute), 94
- signature_count (*WithdrawalModel* attribute), 34
- since_height (*DeploymentFlagsModel* attribute), 20
- size (*BlockModel* attribute), 91
- size (*BlockTransactionDetailsModel* attribute), 93
- size (*TransactionModel* attribute), 98
- Small (*DepositRetrievalType* attribute), 125
- smart_contract_wallet (*CirrusMasterNode* property), 117
- smart_contract_wallet (*CirrusNode* property), 105
- smart_contract_wallet (*InterfluxCirrusNode* property), 111
- smart_contract_wallet (*StraxMasterNode* property), 115
- smart_contracts (*CirrusMasterNode* property), 118
- smart_contracts (*CirrusNode* property), 105
- smart_contracts (*InterfluxCirrusNode* property), 110
- smart_contracts (*StraxMasterNode* property), 115
- SmartContractParameter (class in *pys-tratis.core.smartcontractparameter*), 129
- SmartContractParameterType (class in *pys-tratis.core.smartcontractparametertype*), 129
- SmartContracts (class in *pystratis.api.smartcontracts*), 51
- SmartContractWallet (class in *pys-tratis.api.smartcontractwallet*), 60
- spendable_amount (*AccountBalanceModel* attribute), 90
- spendable_amount (*AddressBalanceModel* attribute), 90
- spendable_transactions() (*Wallet* method), 73
- SpendableTransactionModel (class in *pys-tratis.api.wallet.responsemodels.spendabletransactionmodel*), 85
- SpendableTransactionsModel (class in *pys-tratis.api.wallet.responsemodels.spendabletransactionsmodel*), 86
- spending_output_details (*WithdrawalModel* attribute), 34
- split_coins() (*Wallet* method), 79
- Staked (*ContractTransactionItemType* attribute), 123
- Staked (*TransactionItemType* attribute), 130
- Staking (class in *pystratis.api.staking*), 64
- staking (*GetStakingInfoModel* attribute), 65
- staking (*InterfluxStraxNode* property), 108
- staking (*StraxNode* property), 102
- start_collecting_peerstatistics() (*Diagnostic* method), 22
- start_multistaking() (*Staking* method), 64
- start_staking() (*Staking* method), 64
- Started (*FullNodeState* attribute), 101
- Starting (*FullNodeState* attribute), 101

- starting_height (*PeerInfoModel* attribute), 19
- state (*ConnectedPeerInfoModel* attribute), 22
- state (*FeaturesDataModel* attribute), 46
- state (*StatusModel* attribute), 47
- state_value (*DeploymentFlagsModel* attribute), 20
- stats() (*Dashboard* method), 21
- status (*AsyncLoopsModel* attribute), 45
- status() (*Node* method), 42
- status_burns() (*Interop* method), 34
- status_mints() (*Interop* method), 34
- status_votes() (*Interop* method), 34
- StatusModel (class in *pys-tratis.api.node.responsemodels.statusmodel*), 46
- stop() (*Node* method), 44
- stop_collecting_peerstatistics() (*Diagnostic* method), 22
- stop_mining() (*Mining* method), 40
- stop_node() (*CirrusMasterNode* method), 118
- stop_node() (*CirrusNode* method), 107
- stop_node() (*InterfluxCirrusNode* method), 112
- stop_node() (*InterfluxStraxNode* method), 109
- stop_node() (*StraxMasterNode* method), 115
- stop_node() (*StraxNode* method), 104
- stop_staking() (*Staking* method), 65
- storage() (*SmartContracts* method), 51
- Strax (*CoinType* attribute), 123
- STRAX (*DestinationChain* attribute), 126
- StraxMain (class in *pys-tratis.core.networks.straxnetwork*), 118
- StraxMasterNode (class in *pys-tratis.nodes.cirrusminernode*), 113
- StraxNode (class in *pystratis.nodes.straxnode*), 102
- StraxRegTest (class in *pys-tratis.core.networks.straxnetwork*), 119
- StraxTest (class in *pys-tratis.core.networks.straxnetwork*), 118
- String (*SmartContractParameterType* attribute), 129
- Submitted (*ConversionRequestStatus* attribute), 124
- subtraction_fee_from_amount (*Recipient* attribute), 129
- subver (*PeerInfoModel* attribute), 19
- success (*BuildContractTransactionModel* attribute), 94
- success (*BuildCreateContractTransactionModel* attribute), 95
- success (*ReceiptModel* attribute), 60
- sum (*AddressBalanceModel* attribute), 57
- Suspended (*CrossChainTransferStatus* attribute), 124
- sweep() (*Wallet* method), 80
- sync() (*FederationWallet* method), 32
- sync() (*Notifications* method), 49
- sync() (*Wallet* method), 79
- sync_from_date() (*Wallet* method), 79
- synced_blocks (*PeerInfoModel* attribute), 19
- synced_headers (*PeerInfoModel* attribute), 19
- ## T
- target_address (*Deposit* attribute), 125
- target_chain (*Deposit* attribute), 125
- Testnet (*CoinType* attribute), 123
- testnet (*StatusModel* attribute), 47
- threshold (*DeploymentFlagsModel* attribute), 20
- threshold_state (*DeploymentFlagsModel* attribute), 20
- time (*BlockHeaderModel* attribute), 45
- time (*BlockModel* attribute), 92
- time (*BlockTransactionDetailsModel* attribute), 93
- time (*TransactionModel* attribute), 98
- time_start (*DeploymentFlagsModel* attribute), 21
- time_time_out (*DeploymentFlagsModel* attribute), 21
- timeoffset (*PeerInfoModel* attribute), 18
- timestamp (*TransactionItemModel* attribute), 86
- timestamp_difference_between_transactions (*DistributeUtxoModel* attribute), 84
- tip_hash (*AddressIndexerTipModel* attribute), 5
- tip_height (*AddressIndexerTipModel* attribute), 5
- tip_height (*ConnectedPeerModel* attribute), 46
- to_address (*ContractTransactionItemModel* attribute), 63
- to_address (*ReceiptModel* attribute), 59
- to_address (*TransactionItemModel* attribute), 86
- to_address (*TransferInfoModel* attribute), 60
- to_bytes() (*hexstr* method), 120
- to_coin_unit() (*Money* method), 121
- to_hex() (*int32* method), 120
- to_hex() (*int64* method), 121
- to_hex() (*uint128* method), 122
- to_hex() (*uint160* method), 122
- to_hex() (*uint256* method), 122
- to_hex() (*uint32* method), 122
- to_hex() (*uint64* method), 123
- topics (*LogModel* attribute), 59
- total_utxo_count (*WalletStatsModel* attribute), 88
- transaction (*GetLastBalanceUpdateTransactionModel* attribute), 8
- transaction_count() (*Wallet* method), 71
- transaction_fee (*ContractTransactionItemModel* attribute), 63
- transaction_hash (*ReceiptModel* attribute), 59
- transaction_hex (*SetupModel* attribute), 16
- transaction_hex (*WithdrawalModel* attribute), 16
- transaction_id (*BuildContractTransactionModel* attribute), 94
- transaction_id (*BuildCreateContractTransactionModel* attribute), 95
- transaction_id (*BuildTransactionModel* attribute), 95
- transaction_id (*Outpoint* attribute), 128

- transaction_id (*RemovedTransactionModel* attribute), 97
- transaction_id (*SpendableTransactionModel* attribute), 85
- transaction_id (*TransactionItemModel* attribute), 86
- transaction_id (*UtxoDescriptor* attribute), 99
- transaction_id (*UTXOModel* attribute), 9
- transaction_id (*WalletSendTransactionModel* attribute), 100
- transaction_type (*TransactionItemModel* attribute), 86
- TransactionItemModel* (class in *pys-tratis.api.wallet.responsemodels.transactionitemmodel*), 86
- TransactionItemType* (class in *pys-tratis.core.transactionitemtype*), 130
- TransactionModel* (class in *pys-tratis.api.global_responsemodels.transactionmodel*), 98
- TransactionOutputModel* (class in *pys-tratis.api.global_responsemodels.transactionoutputmodel*), 98
- TransactionResponseModel* (class in *pys-tratis.api.interop.responsemodels.transactionresponsemodel*), 39
- transactions (*BlockTransactionDetailsModel* attribute), 93
- transactions (*SpendableTransactionsModel* attribute), 86
- transactions_history (*AccountHistoryModel* attribute), 82
- transfer() (*FederationGateway* method), 28
- transfer_status (*CrossChainTransferModel* attribute), 29
- transfer_status (*WithdrawalModel* attribute), 34
- TransferInfoModel* (class in *pys-tratis.api.smartcontracts.responsemodels.transferinfomodel*), 60
- transfers_delete_suspended() (*FederationGateway* method), 28
- trx_id (*WithdrawalModel* attribute), 34
- tx (*BlockModel* attribute), 92
- tx (*BlockTransactionDetailsModel* attribute), 93
- tx (*CrossChainTransferModel* attribute), 29
- tx_output_index (*TransactionItemModel* attribute), 86
- tx_output_time (*TransactionItemModel* attribute), 86
- txid (*TransactionModel* attribute), 98
- txid (*VIn* attribute), 99
- type (*GetCodeModel* attribute), 58
- type (*ScriptPubKey* attribute), 97
- U**
- uint128 (class in *pystratis.core.types.uint128*), 122
- UInt128 (*SmartContractParameterType* attribute), 130
- uint160 (class in *pystratis.core.types.uint160*), 122
- uint256 (class in *pystratis.core.types.uint256*), 122
- UInt256 (*SmartContractParameterType* attribute), 130
- uint32 (class in *pystratis.core.types.uint32*), 122
- UInt32 (*SmartContractParameterType* attribute), 129
- uint64 (class in *pystratis.core.types.uint64*), 123
- UInt64 (*SmartContractParameterType* attribute), 129
- uncompressed() (*PubKey* method), 128
- Uninitialized (*FeatureInitializationState* attribute), 101
- unique_block_count (*WalletStatsModel* attribute), 88
- unique_transaction_count (*WalletStatsModel* attribute), 88
- Unprocessed (*ConversionRequestStatus* attribute), 124
- unsigned_transaction (*BuildOfflineSignModel* attribute), 95
- unused_address() (*Wallet* method), 76
- unused_addresses() (*Wallet* method), 77
- use_unique_address_per_utxo (*DistributeUtxoModel* attribute), 84
- utxo_amounts (*WalletStatsModel* attribute), 88
- utxo_per_block (*UtxoPerBlockModel* attribute), 87
- utxo_per_block (*WalletStatsModel* attribute), 88
- utxo_per_block_transaction (*DistributeUtxoModel* attribute), 84
- utxo_per_transaction (*UtxoPerTransactionModel* attribute), 87
- utxo_per_transaction (*WalletStatsModel* attribute), 88
- UtxoAmountModel* (class in *pys-tratis.api.wallet.responsemodels.utxoamountmodel*), 87
- UtxoDescriptor* (class in *pys-tratis.api.global_responsemodels.utxodescriptor*), 99
- UTXOModel* (class in *pys-tratis.api.blockstore.responsemodels.utxomodel*), 9
- UtxoPerBlockModel* (class in *pys-tratis.api.wallet.responsemodels.utxoperblockmodel*), 87
- UtxoPerTransactionModel* (class in *pys-tratis.api.wallet.responsemodels.utxopertransactionmodel*), 87
- utxos (*BuildOfflineSignModel* attribute), 95
- utxos (*GetUTXOsForAddressModel* attribute), 8
- utxos_count (*DistributeUtxoModel* attribute), 84
- V**
- validate_address() (*CirrusMain* method), 119
- validate_address() (*CirrusRegTest* method), 120
- validate_address() (*CirrusTest* method), 119
- validate_address() (*Node* method), 43
- validate_address() (*StraxMain* method), 118

- validate_address() (*StraxRegTest* method), 119
 validate_address() (*StraxTest* method), 119
 validate_value() (*int32* method), 120
 validate_value() (*int64* method), 121
 validate_value() (*uint128* method), 122
 validate_value() (*uint160* method), 122
 validate_value() (*uint256* method), 122
 validate_value() (*uint32* method), 122
 validate_value() (*uint64* method), 123
 validate_values() (*Address* static method), 120
 validate_values() (*SmartContractParameter* static method), 129
 ValidateAddressModel (class in *pys-tratis.api.node.responsemodels.validateaddressmodel*), 48
 ValidateTransactionResultModel (class in *pys-tratis.api.federationgateway.responsemodels.validatetransactionresultmodel*), 31
 value (*GetTxOutModel* attribute), 46
 value (*int32* property), 120
 value (*int64* property), 121
 value (*Money* property), 121
 value (*RPCCommandResponseModel* attribute), 50
 value (*SerializableResult* attribute), 31
 value (*TransactionResponseModel* attribute), 39
 value (*TransferInfoModel* attribute), 60
 value (*uint128* property), 122
 value (*uint160* property), 122
 value (*uint256* property), 122
 value (*uint32* property), 122
 value (*uint64* property), 123
 value (*UTXOModel* attribute), 9
 value (*VOut* attribute), 99
 VerboseAddressBalanceModel (class in *pys-tratis.api.blockstore.responsemodels.verboseaddressbalancemodel*), 9
 verify_message() (*Wallet* method), 70
 verify_transfer() (*FederationGateway* method), 28
 version (*BlockHeaderModel* attribute), 45
 version (*BlockModel* attribute), 91
 version (*BlockTransactionDetailsModel* attribute), 93
 version (*ConnectedPeerModel* attribute), 46
 version (*ExtPubKey* attribute), 127
 version (*PeerInfoModel* attribute), 19
 version (*StatusModel* attribute), 47
 version (*TransactionModel* attribute), 98
 version_hex (*BlockModel* attribute), 91
 version_hex (*BlockTransactionDetailsModel* attribute), 93
 versions (*DeploymentFlagsModel* attribute), 20
 VIn (class in *pystratis.api.global_responsemodels.vin*), 99
 vin (*TransactionModel* attribute), 98
 VoteFinalised (*ConversionRequestStatus* attribute), 124
 votes (*DeploymentFlagsModel* attribute), 20
 voting (*CirrusMasterNode* property), 118
 voting (*CirrusNode* property), 105
 Voting (class in *pystratis.api.voting*), 66
 voting (*InterfluxCirrusNode* property), 111
 voting (*StraxMasterNode* property), 115
 voting_data_string (*PollViewModel* attribute), 96
 VotingDataModel (class in *pys-tratis.api.voting.responsemodels.votingdatamodel*), 68
 VOut (class in *pystratis.api.global_responsemodels.vout*), 99
 vout (*TransactionModel* attribute), 98
 vout (*VIn* attribute), 99
 vsize (*TransactionModel* attribute), 98
W
 wallet (*CirrusMasterNode* property), 118
 wallet (*CirrusNode* property), 107
 Wallet (class in *pystratis.api.wallet*), 69
 wallet (*InterfluxCirrusNode* property), 112
 wallet (*InterfluxStraxNode* property), 109
 wallet (*StraxMasterNode* property), 115
 wallet (*StraxNode* property), 104
 wallet_account (*BuildOfflineSignModel* attribute), 95
 wallet_name (*BuildOfflineSignModel* attribute), 95
 wallet_name (*DistributeUtxoModel* attribute), 84
 wallet_name (*WalletGeneralInfoModel* attribute), 100
 wallet_name (*WalletSecret* attribute), 130
 wallet_name (*WalletStatsModel* attribute), 88
 wallet_password (*WalletSecret* attribute), 130
 wallet_send_transaction (*DistributeUtxoModel* attribute), 84
 wallet_stats() (*Wallet* method), 79
 WalletBalanceModel (class in *pys-tratis.api.global_responsemodels.walletbalancemodel*), 100
 WalletGeneralInfoModel (class in *pys-tratis.api.global_responsemodels.walletgeneralinfomodel*), 100
 WalletHistoryModel (class in *pys-tratis.api.wallet.responsemodels.wallethistormodel*), 88
 WalletSecret (class in *pystratis.core.walletsecret*), 130
 WalletSendTransactionModel (class in *pys-tratis.api.global_responsemodels.walletsendtransactionmodel*), 100
 WalletStatsModel (class in *pys-tratis.api.wallet.responsemodels.walletstatsmodel*), 88
 weight (*BlockModel* attribute), 91
 weight (*BlockTransactionDetailsModel* attribute), 93
 weight (*GetStakingInfoModel* attribute), 65
 weight (*TransactionModel* attribute), 98

`whitelisted` (*PeerInfoModel* attribute), 19
`whitelisted_hashes()` (*Voting* method), 66
`WhitelistedHashesModel` (class in *pys-*
tratis.api.voting.responsemodels.whitelistedhashesmodel),
69
`with_traceback()` (*APIError* method), 102
`withdrawal()` (*ColdStaking* method), 12
`WithdrawalModel` (class in *pys-*
tratis.api.coldstaking.responsemodels.withdrawalmodel),
16
`WithdrawalModel` (class in *pys-*
tratis.api.federationwallet.responsemodels.withdrawalmodel),
34

X

`x` (*PubKey* attribute), 128

Y

`y` (*PubKey* attribute), 128