Safe-eth-py

Uxio, Moisés

Mar 05, 2024

INTRO

1 Table of Contents **3**

1.1 Quick start . . . 3
1.2 Ethereum utils . . . 3
1.3 Ethereum django (REST) utils . . . 5
1.4 Gnosis Products . . . 6
1.5 gnosis . . . 7

2 Indices and tables **129**

Python Module Index **131**

Index **133**

Safe-eth-py includes a set of libraries to work with Ethereum and Gnosis projects:

- *EthereumClient*, a wrapper over Web3.py *Web3* client including utilities to deal with ERC20/721 tokens and tracing.
- Gnosis Safe classes and utilities.
- Price oracles for *Uniswap*, *Kyber*…
- Django serializers, models and utils.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Quick start

Just run `pip install safe-eth-py` or add it to your **requirements.txt**

If you want django ethereum utils (models, serializers, filters...) you need to run `pip install safe-eth-py[django]`

If you have issues building **coincurve** maybe you are missing some libraries

1.2 Ethereum utils

1.2.1 gnosis.eth

- `class EthereumClient (ethereum_node_url: str)`: Class to connect and do operations with an ethereum node. Uses web3 and raw rpc calls for things not supported in web3. Only `http/https` urls are supported for the node url.

`EthereumClient` has some utils that improve a lot performance using Ethereum nodes, like the possibility of doing `batch_calls` (a single request making read-only calls to multiple contracts):

```
from gnosis.eth import EthereumClient
from gnosis.eth.contracts import get_erc721_contract
ethereum_client = EthereumClient(ETHEREUM_NODE_URL)
erc721_contract = get_erc721_contract(ethereum_client.w3, token_address)
name, symbol = ethereum_client.batch_call([
                    erc721_contract.functions.name(),
                    erc721_contract.functions.symbol(),
                ])
```

More optimal in case you want to call the same function in multiple contracts

```
from gnosis.eth import EthereumClient
from gnosis.eth.contracts import get_erc20_contract
ethereum_client = EthereumClient(ETHEREUM_NODE_URL)
erc20_contract = get_erc20_contract(ethereum_client.w3, token_address)
my_account = '0xD0E03B027A367fED4fd0E7834a82CD8A73E76B45'
name, symbol = ethereum_client.batch_call_same_function(
                    erc20_contract.functions.balanceOf(my_account),
                    ['0x6810e776880C02933D47DB1b9fc05908e5386b96',
```

(continues on next page)

(continued from previous page)

```
↪'0x6B175474E89094C44Da98b954EedeAC495271d0F']
                )
```

If you want to use the underlying web3.py library:

```
from gnosis.eth import EthereumClient
ethereum_client = EthereumClient(ETHEREUM_NODE_URL)
ethereum_client.w3.eth.get_block(57)
```

`EthereumClient` supports EIP1559 fees:

```
from gnosis.eth import TxSpeed
base_fee, priority_fee = ethereum_client.estimate_fee_eip1559(tx_speed=TxSpeed.NORMAL)
# If you want to convert a legacy tx to a EIP1559 one
eip1559_tx = ethereum_client.set_eip1559_fees(legacy_tx, tx_speed=TxSpeed.NORMAL)
```

You can modify timeouts (in seconds) for the RPC endpoints by setting *ETHEREUM_RPC_TIMEOUT* and *ETHEREUM_RPC_SLOW_TIMEOUT* as environment variables.

By default every RPC request will be retried *3* times. You can modify that by setting *ETHEREUM_RPC_RETRY_COUNT*.

1.2.2 gnosis.eth.constants

- `NULL_ADDRESS (0x000...0)`: Solidity `address(0)`.
- `SENTINEL_ADDRESS (0x000...1)`: Used for Gnosis Safe's linked lists (modules, owners...).
- Maximum and minimum values for *R*, *S* and *V* in ethereum signatures.

1.2.3 gnosis.eth.eip712

```
from gnosis.eth.eip712 import eip712_encode_hash

types = {'EIP712Domain': [{'name': 'name', 'type': 'string'},
                          {'name': 'version', 'type': 'string'},
                          {'name': 'chainId', 'type': 'uint256'},
                          {'name': 'verifyingContract', 'type': 'address'}],
         'Mailbox': [{'name': 'owner', 'type': 'address'},
                     {'name': 'messages', 'type': 'Message[]'}],
         'Message': [{'name': 'sender', 'type': 'address'},
                     {'name': 'subject', 'type': 'string'},
                     {'name': 'isSpam', 'type': 'bool'},
                     {'name': 'body', 'type': 'string'}]}

msgs = [{'sender': ADDRESS,
         'subject': 'Hello World',
         'body': 'The sparrow flies at midnight.',
         'isSpam': False},
        {'sender': ADDRESS,
         'subject': 'You may have already Won! :dumb-emoji:',
         'body': 'Click here for sweepstakes!',
```

(continues on next page)

(continued from previous page)

```
         'isSpam': True}]

mailbox = {'owner': ADDRESS,
           'messages': msgs}

payload = {'types': types,
           'primaryType': 'Mailbox',
           'domain': {'name': 'MyDApp',
                      'version': '3.0',
                      'chainId': 41,
                      'verifyingContract': ADDRESS},
           'message': mailbox}

eip712_hash = eip712_encode_hash(payload)
```

1.2.4 gnosis.eth.oracles

Price oracles for Uniswap, UniswapV2, Kyber, SushiSwap, Aave, Balancer, Curve, Mooniswap, Yearn... Example:

```
from gnosis.eth import EthereumClient
from gnosis.eth.oracles import UniswapV2Oracle
ethereum_client = EthereumClient(ETHEREUM_NODE_URL)
uniswap_oracle = UniswapV2Oracle(ethereum_client)
gno_token_mainnet_address = '0x6810e776880C02933D47DB1b9fc05908e5386b96'
weth_token_mainnet_address = '0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2'
price = uniswap_oracle.get_price(gno_token_mainnet_address, uniswap_oracle.weth_address)
```

1.2.5 gnosis.eth.utils

Contains utils for ethereum operations:

- `mk_contract_address_2(from_: Union[str, bytes], salt: Union[str, bytes], init_code: [str, bytes]) -> str`: Calculates the address of a new contract created using the new CREATE2 opcode.

1.3 Ethereum django (REST) utils

Django utils are available under `gnosis.eth.django`. You can find a set of helpers for working with Ethereum using Django and Django Rest framework.

It includes:

- **gnosis.eth.django.filters**: EthereumAddressFilter.
- **gnosis.eth.django.models**: Model fields (Ethereum address, Ethereum big integer field).
- **gnosis.eth.django.serializers**: Serializer fields (Ethereum address field, hexadecimal field).
- **gnosis.eth.django.validators**: Ethereum related validators.
- **gnosis.safe.serializers**: Serializers for Gnosis Safe (signature, transaction...).
- All the tests are written using Django Test suite.

1.4 Gnosis Products

1.4.1 Safe

On `gnosis.safe` there're classes to work with Gnosis Safe

```
from gnosis.eth import EthereumClient
from gnosis.safe import Safe
safe_address = ''  # Fill with checksummed version of a Safe address
ethereum_client = EthereumClient(ETHEREUM_NODE_URL)
safe = Safe(safe_address, ethereum_client)
safe_info = safe.retrieve_all_info()
```

To work with Multisig Transactions:

```
safe_tx = safe.build_multisig_tx(to, value, data, operation, safe_tx_gas, base_gas, gas_
↪price, gas_token,
                                 refund_receiver, signatures, safe_nonce)
safe_tx.sign(owner_1_private_key)
safe_tx.sign(owner_2_private_key)
safe_tx.call()  # Check it works
safe_tx.execute(tx_sender_private_key)
```

1.4.2 CowSwap

On `gnosis.cowswap` there're classes to work with CowSwap

```
import time
from gnosis.eth import EthereumNetwork
from gnosis.cowswap import Order, OrderKind, CowSwapAPI

account_address = ''  # Fill with checksummed version of a CowSwap user address
account_private_key = ''  # Fill with private key of a user address
cow_swap_api = CowSwapAPI(EthereumNetwork.SEPOLIA)
print(cow_swap_api.get_trades(owner=account_address))
buy_amount = cow_swap_api.get_estimated_amount(base_token, quote_token, OrderKind.SELL,␣
↪sell_amount)
valid_to = int(time.time() + (24 * 60 * 60))  # Order valid for 1 day
order = Order(
      sellToken=base_token,
      buyToken=buyToken,
      receiver=receiver,
      sellAmount=sell_amount,
      buyAmount=buy_amount,
      validTo=valid_to,  # timestamp
      appData={},  # Dict with CowSwap AppData schema definition (https://github.com/
↪cowprotocol/app-data)
      fee_amount=0,  # If set to `0` it will be autodetected
      kind='sell',  # `sell` or `buy`
      partiallyFillable=True,  # `True` or `False`
      sellTokenBalance='erc20',  # `erc20`, `external` or `internal`
```

(continues on next page)

(continued from previous page)

```
      buyTokenBalance='erc20',  # `erc20` or `internal`
  )
cow_swap_api.place_order(order, account_private_key)
```

1.5 gnosis

1.5.1 gnosis package

Subpackages

gnosis.eth package

Subpackages

gnosis.eth.clients package

Submodules

gnosis.eth.clients.blockscout_client module

exception gnosis.eth.clients.blockscout_client.**BlockScoutConfigurationProblem**

Bases: *BlockscoutClientException*

class gnosis.eth.clients.blockscout_client.**BlockscoutClient**(*network:* EthereumNetwork)

Bases: object

```
NETWORK_WITH_URL = {EthereumNetwork.ACALA_NETWORK:
'https://blockscout.acala.network/graphiql', EthereumNetwork.ARTHERA_MAINNET:
'https://explorer.arthera.net/graphiql', EthereumNetwork.ARTHERA_TESTNET:
'https://explorer-test.arthera.net/graphiql', EthereumNetwork.ASTAR:
'https://blockscout.com/astar/graphiql', EthereumNetwork.BOBA_NETWORK:
'https://blockexplorer.boba.network/graphiql', EthereumNetwork.CASCADIA_TESTNET:
'https://explorer.cascadia.foundation/graphiql', EthereumNetwork.CRONOS_MAINNET:
'https://cronos.org/explorer/graphiql', EthereumNetwork.CRONOS_TESTNET:
'https://cronos.org/explorer/testnet3/graphiql', EthereumNetwork.CROSSBELL:
'https://scan.crossbell.io/graphiql', EthereumNetwork.ENERGY_WEB_CHAIN:
'https://explorer.energyweb.org/graphiql', EthereumNetwork.ENERGY_WEB_VOLTA_TESTNET:
'https://volta-explorer.energyweb.org/graphiql', EthereumNetwork.ETHEREUM_CLASSIC:
'https://blockscout.com/etc/mainnet/graphiql', EthereumNetwork.EVMOS:
'https://evm.evmos.org/graphiql', EthereumNetwork.EVMOS_TESTNET:
'https://evm.evmos.dev/graphiql', EthereumNetwork.FUSE_MAINNET:
'https://explorer.fuse.io/graphiql', EthereumNetwork.GATHER_DEVNET_NETWORK:
'https://devnet-explorer.gather.network/graphiql',
EthereumNetwork.GATHER_MAINNET_NETWORK: 'https://explorer.gather.network/graphiql',
EthereumNetwork.GATHER_TESTNET_NETWORK:
'https://testnet-explorer.gather.network/graphiql', EthereumNetwork.GNOSIS:
'https://gnosis.blockscout.com/api/v1/graphql', EthereumNetwork.GODWOKEN_MAINNET:
'https://v1.gwscan.com/graphiql', EthereumNetwork.GODWOKEN_TESTNET_V1:
'https://v1.betanet.gwscan.com/graphiql', EthereumNetwork.HAQQ_CHAIN_TESTNET:
'https://explorer.testedge2.haqq.network/graphiql', EthereumNetwork.HAQQ_NETWORK:
'https://explorer.haqq.network/graphiql', EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET:
'https://mainnet.japanopenchain.org/graphiql',
EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org/graphiql',
EthereumNetwork.KARURA_NETWORK_TESTNET:
'https://blockscout.karura.network/graphiql', EthereumNetwork.KCC_MAINNET:
'https://scan.kcc.io/graphiql', EthereumNetwork.KCC_TESTNET:
'https://scan-testnet.kcc.network/graphiql', EthereumNetwork.KLAYTN_MAINNET_CYPRESS:
'https://scope.klaytn.com/graphiql', EthereumNetwork.KLAYTN_TESTNET_BAOBAB:
'https://baobab.scope.klaytn.com/graphiql', EthereumNetwork.KROMA:
'https://blockscout.kroma.network/graphiql', EthereumNetwork.KROMA_SEPOLIA:
'https://blockscout.sepolia.kroma.network/graphiql', EthereumNetwork.LINEA:
'https://explorer.linea.build/graphiql', EthereumNetwork.LINEA_TESTNET:
'https://explorer.goerli.linea.build/graphiql',
EthereumNetwork.LISK_SEPOLIA_TESTNET:
'https://sepolia-blockscout.lisk.com/graphiql',
EthereumNetwork.MANTA_PACIFIC_MAINNET:
'https://pacific-explorer.manta.network/graphiql', EthereumNetwork.MANTLE:
'https://explorer.mantle.xyz/graphiql', EthereumNetwork.MANTLE_SEPOLIA_TESTNET:
'https://explorer.sepolia.mantle.xyz/graphiql', EthereumNetwork.MANTLE_TESTNET:
'https://explorer.testnet.mantle.xyz/graphiql', EthereumNetwork.METER_MAINNET:
'https://scan.meter.io/graphiql', EthereumNetwork.METER_TESTNET:
'https://scan-warringstakes.meter.io/graphiql',
EthereumNetwork.METIS_ANDROMEDA_MAINNET:
'https://andromeda-explorer.metis.io/graphiql',
EthereumNetwork.METIS_GOERLI_TESTNET:
'https://goerli.explorer.metisdevops.link/graphiql', EthereumNetwork.MODE:
'https://explorer.mode.network/graphiql', EthereumNetwork.MODE_TESTNET:
'https://sepolia.explorer.mode.network/graphiql', EthereumNetwork.MORDOR_TESTNET:
'https://blockscout.com/etc/mordor/graphiql', EthereumNetwork.NEON_EVM_DEVNET:
'https://neon-devnet.blockscout.com/graphiql', EthereumNetwork.NEON_EVM_MAINNET:
'https://neon.blockscout.com/graphiql', EthereumNetwork.OASIS_SAPPHIRE:
'https://explorer.sapphire.oasis.io/graphiql',
EthereumNetwork.OASIS_SAPPHIRE_TESTNET:
'https://testnet.explorer.sapphire.oasis.dev/graphiql',
EthereumNetwork.OP_SEPOLIA_TESTNET:
```

build_url(*path: str*)

get_contract_metadata(*address: ChecksumAddress*) → *ContractMetadata* | None

exception gnosis.eth.clients.blockscout_client.**BlockscoutClientException**

Bases: Exception

gnosis.eth.clients.contract_metadata module

class gnosis.eth.clients.contract_metadata.**ContractMetadata**(*name: str | None*, *abi: List[Dict[str, Any]]*, *partial_match: bool*)

Bases: object

abi: List[Dict[str, Any]]

name: str | None

partial_match: bool

gnosis.eth.clients.etherscan_client module

class gnosis.eth.clients.etherscan_client.**EtherscanClient**(*network: EthereumNetwork*, *api_key: str | None = None*, *request_timeout: int = 10*)

Bases: object

HTTP_HEADERS = {'User-Agent': 'curl/7.77.0'}

```
NETWORK_WITH_API_URL = {EthereumNetwork.ARBITRUM_GOERLI:
'https://api-goerli.arbiscan.io', EthereumNetwork.ARBITRUM_NOVA:
'https://api-nova.arbiscan.io', EthereumNetwork.ARBITRUM_ONE:
'https://api.arbiscan.io', EthereumNetwork.ARBITRUM_SEPOLIA:
'https://api-sepolia.arbiscan.io', EthereumNetwork.AVALANCHE_C_CHAIN:
'https://api.snowtrace.io', EthereumNetwork.BASE_GOERLI_TESTNET:
'https://api-goerli.basescan.org', EthereumNetwork.BNB_SMART_CHAIN_MAINNET:
'https://api.bscscan.com', EthereumNetwork.CELO_MAINNET: 'https://api.celoscan.io',
EthereumNetwork.CRONOS_MAINNET: 'https://api.cronoscan.com',
EthereumNetwork.CRONOS_TESTNET: 'https://api-testnet.cronoscan.com',
EthereumNetwork.FANTOM_OPERA: 'https://api.ftmscan.com',
EthereumNetwork.FANTOM_TESTNET: 'https://api-testnet.ftmscan.com',
EthereumNetwork.GNOSIS: 'https://api.gnosisscan.io', EthereumNetwork.GOERLI:
'https://api-goerli.etherscan.io', EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET:
'https://mainnet.japanopenchain.org/api', EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org/api', EthereumNetwork.KROMA:
'https://api.kromascan.com', EthereumNetwork.KROMA_SEPOLIA:
'https://api-sepolia.kromascan.com', EthereumNetwork.LINEA:
'https://api.lineascan.build', EthereumNetwork.LINEA_TESTNET:
'https://api-testnet.lineascan.build', EthereumNetwork.MAINNET:
'https://api.etherscan.io', EthereumNetwork.MANTLE: 'https://explorer.mantle.xyz',
EthereumNetwork.MANTLE_TESTNET: 'https://explorer.testnet.mantle.xyz',
EthereumNetwork.MOONBASE_ALPHA: 'https://api-moonbase.moonscan.io',
EthereumNetwork.MOONBEAM: 'https://api-moonbeam.moonscan.io',
EthereumNetwork.MOONRIVER: 'https://api-moonriver.moonscan.io',
EthereumNetwork.NEON_EVM_DEVNET: 'https://devnet-api.neonscan.org',
EthereumNetwork.NEON_EVM_MAINNET: 'https://api.neonscan.org',
EthereumNetwork.OPTIMISM: 'https://api-optimistic.etherscan.io',
EthereumNetwork.POLYGON: 'https://api.polygonscan.com',
EthereumNetwork.POLYGON_ZKEVM: 'https://api-zkevm.polygonscan.com',
EthereumNetwork.RINKEBY: 'https://api-rinkeby.etherscan.io',
EthereumNetwork.ROPSTEN: 'https://api-ropsten.etherscan.io', EthereumNetwork.SCROLL:
'https://api.scrollscan.com', EthereumNetwork.SCROLL_SEPOLIA_TESTNET:
'https://api-sepolia.scrollscan.dev', EthereumNetwork.SEPOLIA:
'https://api-sepolia.etherscan.io', EthereumNetwork.ZKSYNC_MAINNET:
'https://block-explorer-api.mainnet.zksync.io/'}
```

NETWORK_WITH_URL = {EthereumNetwork.ARBITRUM_GOERLI: 'https://goerli.arbiscan.io',
EthereumNetwork.ARBITRUM_NOVA: 'https://nova.arbiscan.io',
EthereumNetwork.ARBITRUM_ONE: 'https://arbiscan.io',
EthereumNetwork.AVALANCHE_C_CHAIN: 'https://snowtrace.io',
EthereumNetwork.BASE_GOERLI_TESTNET: 'https://goerli.basescan.org',
EthereumNetwork.BNB_SMART_CHAIN_MAINNET: 'https://bscscan.com',
EthereumNetwork.CELO_MAINNET: 'https://celoscan.io', EthereumNetwork.CRONOS_MAINNET:
'https://cronoscan.com', EthereumNetwork.CRONOS_TESTNET:
'https://testnet.cronoscan.com', EthereumNetwork.FANTOM_OPERA:
'https://ftmscan.com', EthereumNetwork.FANTOM_TESTNET:
'https://testnet.ftmscan.com/', EthereumNetwork.GNOSIS: 'https://gnosisscan.io',
EthereumNetwork.GOERLI: 'https://goerli.etherscan.io',
EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET: 'https://mainnet.japanopenchain.org',
EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org', EthereumNetwork.KROMA:
'https://kromascan.com', EthereumNetwork.KROMA_SEPOLIA:
'https://sepolia.kromascan.com', EthereumNetwork.LINEA:
'https://www.lineascan.build', EthereumNetwork.LINEA_TESTNET:
'https://goerli.lineascan.build', EthereumNetwork.MAINNET: 'https://etherscan.io',
EthereumNetwork.MANTLE: 'https://explorer.mantle.xyz',
EthereumNetwork.MANTLE_TESTNET: 'https://explorer.testnet.mantle.xyz',
EthereumNetwork.MOONBASE_ALPHA: 'https://moonbase.moonscan.io',
EthereumNetwork.MOONBEAM: 'https://moonscan.io', EthereumNetwork.MOONRIVER:
'https://moonriver.moonscan.io', EthereumNetwork.NEON_EVM_DEVNET:
'https://devnet.neonscan.org', EthereumNetwork.NEON_EVM_MAINNET:
'https://neonscan.org', EthereumNetwork.OPTIMISM: 'https://optimistic.etherscan.io',
EthereumNetwork.POLYGON: 'https://polygonscan.com', EthereumNetwork.POLYGON_ZKEVM:
'https://zkevm.polygonscan.com', EthereumNetwork.RINKEBY:
'https://rinkeby.etherscan.io', EthereumNetwork.ROPSTEN:
'https://ropsten.etherscan.io', EthereumNetwork.SCROLL: 'https://scrollscan.com',
EthereumNetwork.SCROLL_SEPOLIA_TESTNET: 'https://sepolia.scrollscan.dev',
EthereumNetwork.SEPOLIA: 'https://sepolia.etherscan.io',
EthereumNetwork.ZKSYNC_MAINNET: 'https://explorer.zksync.io/'}

build_url(*path: str*)

get_contract_abi(*contract_address: str*, *retry: bool = True*)

get_contract_metadata(*contract_address: str*, *retry: bool = True*) → *ContractMetadata* | None

get_contract_source_code(*contract_address: str*, *retry: bool = True*)

Get source code for a contract. Source code query also returns:

- ContractName: “”,
- CompilerVersion: “”,
- OptimizationUsed: “”,
- Runs: “”,
- ConstructorArguments: “”
- EVMVersion: “Default”,
- Library: “”,
- LicenseType: “”,

- Proxy: “0”,
- Implementation: “”,
- SwarmSource: “”

Parameters

- **contract_address** –
- **retry** – if `True`, try again if there’s Rate Limit Error

Returns

exception gnosis.eth.clients.etherscan_client.**EtherscanClientConfigurationProblem**

Bases: `Exception`

exception gnosis.eth.clients.etherscan_client.**EtherscanClientException**

Bases: `Exception`

exception gnosis.eth.clients.etherscan_client.**EtherscanRateLimitError**

Bases: *EtherscanClientException*

gnosis.eth.clients.sourcify_client module

class gnosis.eth.clients.sourcify_client.**SourcifyClient**(*network: EthereumNetwork = EthereumNetwork.MAINNET*, *base_url_api: str = 'https://sourcify.dev'*, *base_url_repo: str = 'https://repo.sourcify.dev/'*, *request_timeout: int = 10*)

Bases: `object`

Get contract metadata from Sourcify. Matches can be full or partial:

- Full: Both the source files as well as the meta data files were an exact match between the deployed bytecode and the published files.
- Partial: Source code compiles to the same bytecode and thus the contract behaves in the same way, but the source code can be different: Variables can have misleading names, comments can be different and especially the NatSpec comments could have been modified.

get_chains() → Dict[str, Any]

get_contract_metadata(*contract_address: str*) → *ContractMetadata* | None

is_chain_supported(*chain_id: int*) → bool

exception gnosis.eth.clients.sourcify_client.**SourcifyClientConfigurationProblem**

Bases: `Exception`

exception gnosis.eth.clients.sourcify_client.**SourcifyClientException**

Bases: `Exception`

Module contents

exception gnosis.eth.clients.**BlockScoutConfigurationProblem**

Bases: *BlockscoutClientException*

class gnosis.eth.clients.**BlockscoutClient**(*network:* EthereumNetwork)

Bases: object

```
NETWORK_WITH_URL = {EthereumNetwork.ACALA_NETWORK:
'https://blockscout.acala.network/graphiql', EthereumNetwork.ARTHERA_MAINNET:
'https://explorer.arthera.net/graphiql', EthereumNetwork.ARTHERA_TESTNET:
'https://explorer-test.arthera.net/graphiql', EthereumNetwork.ASTAR:
'https://blockscout.com/astar/graphiql', EthereumNetwork.BOBA_NETWORK:
'https://blockexplorer.boba.network/graphiql', EthereumNetwork.CASCADIA_TESTNET:
'https://explorer.cascadia.foundation/graphiql', EthereumNetwork.CRONOS_MAINNET:
'https://cronos.org/explorer/graphiql', EthereumNetwork.CRONOS_TESTNET:
'https://cronos.org/explorer/testnet3/graphiql', EthereumNetwork.CROSSBELL:
'https://scan.crossbell.io/graphiql', EthereumNetwork.ENERGY_WEB_CHAIN:
'https://explorer.energyweb.org/graphiql', EthereumNetwork.ENERGY_WEB_VOLTA_TESTNET:
'https://volta-explorer.energyweb.org/graphiql', EthereumNetwork.ETHEREUM_CLASSIC:
'https://blockscout.com/etc/mainnet/graphiql', EthereumNetwork.EVMOS:
'https://evm.evmos.org/graphiql', EthereumNetwork.EVMOS_TESTNET:
'https://evm.evmos.dev/graphiql', EthereumNetwork.FUSE_MAINNET:
'https://explorer.fuse.io/graphiql', EthereumNetwork.GATHER_DEVNET_NETWORK:
'https://devnet-explorer.gather.network/graphiql',
EthereumNetwork.GATHER_MAINNET_NETWORK: 'https://explorer.gather.network/graphiql',
EthereumNetwork.GATHER_TESTNET_NETWORK:
'https://testnet-explorer.gather.network/graphiql', EthereumNetwork.GNOSIS:
'https://gnosis.blockscout.com/api/v1/graphql', EthereumNetwork.GODWOKEN_MAINNET:
'https://v1.gwscan.com/graphiql', EthereumNetwork.GODWOKEN_TESTNET_V1:
'https://v1.betanet.gwscan.com/graphiql', EthereumNetwork.HAQQ_CHAIN_TESTNET:
'https://explorer.testedge2.haqq.network/graphiql', EthereumNetwork.HAQQ_NETWORK:
'https://explorer.haqq.network/graphiql', EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET:
'https://mainnet.japanopenchain.org/graphiql',
EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org/graphiql',
EthereumNetwork.KARURA_NETWORK_TESTNET:
'https://blockscout.karura.network/graphiql', EthereumNetwork.KCC_MAINNET:
'https://scan.kcc.io/graphiql', EthereumNetwork.KCC_TESTNET:
'https://scan-testnet.kcc.network/graphiql', EthereumNetwork.KLAYTN_MAINNET_CYPRESS:
'https://scope.klaytn.com/graphiql', EthereumNetwork.KLAYTN_TESTNET_BAOBAB:
'https://baobab.scope.klaytn.com/graphiql', EthereumNetwork.KROMA:
'https://blockscout.kroma.network/graphiql', EthereumNetwork.KROMA_SEPOLIA:
'https://blockscout.sepolia.kroma.network/graphiql', EthereumNetwork.LINEA:
'https://explorer.linea.build/graphiql', EthereumNetwork.LINEA_TESTNET:
'https://explorer.goerli.linea.build/graphiql',
EthereumNetwork.LISK_SEPOLIA_TESTNET:
'https://sepolia-blockscout.lisk.com/graphiql',
EthereumNetwork.MANTA_PACIFIC_MAINNET:
'https://pacific-explorer.manta.network/graphiql', EthereumNetwork.MANTLE:
'https://explorer.mantle.xyz/graphiql', EthereumNetwork.MANTLE_SEPOLIA_TESTNET:
'https://explorer.sepolia.mantle.xyz/graphiql', EthereumNetwork.MANTLE_TESTNET:
'https://explorer.testnet.mantle.xyz/graphiql', EthereumNetwork.METER_MAINNET:
'https://scan.meter.io/graphiql', EthereumNetwork.METER_TESTNET:
'https://scan-warringstakes.meter.io/graphiql',
EthereumNetwork.METIS_ANDROMEDA_MAINNET:
'https://andromeda-explorer.metis.io/graphiql',
EthereumNetwork.METIS_GOERLI_TESTNET:
'https://goerli.explorer.metisdevops.link/graphiql', EthereumNetwork.MODE:
'https://explorer.mode.network/graphiql', EthereumNetwork.MODE_TESTNET:
'https://sepolia.explorer.mode.network/graphiql', EthereumNetwork.MORDOR_TESTNET:
'https://blockscout.com/etc/mordor/graphiql', EthereumNetwork.NEON_EVM_DEVNET:
'https://neon-devnet.blockscout.com/graphiql', EthereumNetwork.NEON_EVM_MAINNET:
'https://neon.blockscout.com/graphiql', EthereumNetwork.OASIS_SAPPHIRE:
'https://explorer.sapphire.oasis.io/graphiql',
EthereumNetwork.OASIS_SAPPHIRE_TESTNET:
'https://testnet.explorer.sapphire.oasis.dev/graphiql',
EthereumNetwork.OP_SEPOLIA_TESTNET:
```

build_url(*path: str*)

get_contract_metadata(*address: ChecksumAddress*) → *ContractMetadata* | None

exception gnosis.eth.clients.**BlockscoutClientException**

Bases: Exception

class gnosis.eth.clients.**ContractMetadata**(*name: str | None*, *abi: List[Dict[str, Any]]*, *partial_match: bool*)

Bases: object

abi: List[Dict[str, Any]]

name: str | None

partial_match: bool

class gnosis.eth.clients.**EtherscanClient**(*network:* EthereumNetwork, *api_key: str | None = None*, *request_timeout: int = 10*)

Bases: object

HTTP_HEADERS = {'User-Agent': 'curl/7.77.0'}

```
NETWORK_WITH_API_URL = {EthereumNetwork.ARBITRUM_GOERLI:
'https://api-goerli.arbiscan.io', EthereumNetwork.ARBITRUM_NOVA:
'https://api-nova.arbiscan.io', EthereumNetwork.ARBITRUM_ONE:
'https://api.arbiscan.io', EthereumNetwork.ARBITRUM_SEPOLIA:
'https://api-sepolia.arbiscan.io', EthereumNetwork.AVALANCHE_C_CHAIN:
'https://api.snowtrace.io', EthereumNetwork.BASE_GOERLI_TESTNET:
'https://api-goerli.basescan.org', EthereumNetwork.BNB_SMART_CHAIN_MAINNET:
'https://api.bscscan.com', EthereumNetwork.CELO_MAINNET: 'https://api.celoscan.io',
EthereumNetwork.CRONOS_MAINNET: 'https://api.cronoscan.com',
EthereumNetwork.CRONOS_TESTNET: 'https://api-testnet.cronoscan.com',
EthereumNetwork.FANTOM_OPERA: 'https://api.ftmscan.com',
EthereumNetwork.FANTOM_TESTNET: 'https://api-testnet.ftmscan.com',
EthereumNetwork.GNOSIS: 'https://api.gnosisscan.io', EthereumNetwork.GOERLI:
'https://api-goerli.etherscan.io', EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET:
'https://mainnet.japanopenchain.org/api', EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org/api', EthereumNetwork.KROMA:
'https://api.kromascan.com', EthereumNetwork.KROMA_SEPOLIA:
'https://api-sepolia.kromascan.com', EthereumNetwork.LINEA:
'https://api.lineascan.build', EthereumNetwork.LINEA_TESTNET:
'https://api-testnet.lineascan.build', EthereumNetwork.MAINNET:
'https://api.etherscan.io', EthereumNetwork.MANTLE: 'https://explorer.mantle.xyz',
EthereumNetwork.MANTLE_TESTNET: 'https://explorer.testnet.mantle.xyz',
EthereumNetwork.MOONBASE_ALPHA: 'https://api-moonbase.moonscan.io',
EthereumNetwork.MOONBEAM: 'https://api-moonbeam.moonscan.io',
EthereumNetwork.MOONRIVER: 'https://api-moonriver.moonscan.io',
EthereumNetwork.NEON_EVM_DEVNET: 'https://devnet-api.neonscan.org',
EthereumNetwork.NEON_EVM_MAINNET: 'https://api.neonscan.org',
EthereumNetwork.OPTIMISM: 'https://api-optimistic.etherscan.io',
EthereumNetwork.POLYGON: 'https://api.polygonscan.com',
EthereumNetwork.POLYGON_ZKEVM: 'https://api-zkevm.polygonscan.com',
EthereumNetwork.RINKEBY: 'https://api-rinkeby.etherscan.io',
EthereumNetwork.ROPSTEN: 'https://api-ropsten.etherscan.io', EthereumNetwork.SCROLL:
'https://api.scrollscan.com', EthereumNetwork.SCROLL_SEPOLIA_TESTNET:
'https://api-sepolia.scrollscan.dev', EthereumNetwork.SEPOLIA:
'https://api-sepolia.etherscan.io', EthereumNetwork.ZKSYNC_MAINNET:
'https://block-explorer-api.mainnet.zksync.io/'}
```

NETWORK_WITH_URL = {EthereumNetwork.ARBITRUM_GOERLI: 'https://goerli.arbiscan.io',
EthereumNetwork.ARBITRUM_NOVA: 'https://nova.arbiscan.io',
EthereumNetwork.ARBITRUM_ONE: 'https://arbiscan.io',
EthereumNetwork.AVALANCHE_C_CHAIN: 'https://snowtrace.io',
EthereumNetwork.BASE_GOERLI_TESTNET: 'https://goerli.basescan.org',
EthereumNetwork.BNB_SMART_CHAIN_MAINNET: 'https://bscscan.com',
EthereumNetwork.CELO_MAINNET: 'https://celoscan.io', EthereumNetwork.CRONOS_MAINNET:
'https://cronoscan.com', EthereumNetwork.CRONOS_TESTNET:
'https://testnet.cronoscan.com', EthereumNetwork.FANTOM_OPERA:
'https://ftmscan.com', EthereumNetwork.FANTOM_TESTNET:
'https://testnet.ftmscan.com/', EthereumNetwork.GNOSIS: 'https://gnosisscan.io',
EthereumNetwork.GOERLI: 'https://goerli.etherscan.io',
EthereumNetwork.JAPAN_OPEN_CHAIN_MAINNET: 'https://mainnet.japanopenchain.org',
EthereumNetwork.JAPAN_OPEN_CHAIN_TESTNET:
'https://explorer.testnet.japanopenchain.org', EthereumNetwork.KROMA:
'https://kromascan.com', EthereumNetwork.KROMA_SEPOLIA:
'https://sepolia.kromascan.com', EthereumNetwork.LINEA:
'https://www.lineascan.build', EthereumNetwork.LINEA_TESTNET:
'https://goerli.lineascan.build', EthereumNetwork.MAINNET: 'https://etherscan.io',
EthereumNetwork.MANTLE: 'https://explorer.mantle.xyz',
EthereumNetwork.MANTLE_TESTNET: 'https://explorer.testnet.mantle.xyz',
EthereumNetwork.MOONBASE_ALPHA: 'https://moonbase.moonscan.io',
EthereumNetwork.MOONBEAM: 'https://moonscan.io', EthereumNetwork.MOONRIVER:
'https://moonriver.moonscan.io', EthereumNetwork.NEON_EVM_DEVNET:
'https://devnet.neonscan.org', EthereumNetwork.NEON_EVM_MAINNET:
'https://neonscan.org', EthereumNetwork.OPTIMISM: 'https://optimistic.etherscan.io',
EthereumNetwork.POLYGON: 'https://polygonscan.com', EthereumNetwork.POLYGON_ZKEVM:
'https://zkevm.polygonscan.com', EthereumNetwork.RINKEBY:
'https://rinkeby.etherscan.io', EthereumNetwork.ROPSTEN:
'https://ropsten.etherscan.io', EthereumNetwork.SCROLL: 'https://scrollscan.com',
EthereumNetwork.SCROLL_SEPOLIA_TESTNET: 'https://sepolia.scrollscan.dev',
EthereumNetwork.SEPOLIA: 'https://sepolia.etherscan.io',
EthereumNetwork.ZKSYNC_MAINNET: 'https://explorer.zksync.io/'}

build_url(*path: str*)

get_contract_abi(*contract_address: str*, *retry: bool = True*)

get_contract_metadata(*contract_address: str*, *retry: bool = True*) → *ContractMetadata* | None

get_contract_source_code(*contract_address: str*, *retry: bool = True*)

Get source code for a contract. Source code query also returns:

- ContractName: “”,
- CompilerVersion: “”,
- OptimizationUsed: “”,
- Runs: “”,
- ConstructorArguments: “”
- EVMVersion: “Default”,
- Library: “”,
- LicenseType: “”,

- Proxy: “0”,
- Implementation: “”,
- SwarmSource: “”

Parameters

- **contract_address** –
- **retry** – if `True`, try again if there’s Rate Limit Error

Returns

exception gnosis.eth.clients.**EtherscanClientConfigurationProblem**

Bases: `Exception`

exception gnosis.eth.clients.**EtherscanClientException**

Bases: `Exception`

exception gnosis.eth.clients.**EtherscanRateLimitError**

Bases: *EtherscanClientException*

class gnosis.eth.clients.**SourcifyClient**(*network: EthereumNetwork = EthereumNetwork.MAINNET*, *base_url_api: str = 'https://sourcify.dev'*, *base_url_repo: str = 'https://repo.sourcify.dev/'*, *request_timeout: int = 10*)

Bases: `object`

Get contract metadata from Sourcify. Matches can be full or partial:

- Full: Both the source files as well as the meta data files were an exact match between the deployed bytecode and the published files.
- Partial: Source code compiles to the same bytecode and thus the contract behaves in the same way, but the source code can be different: Variables can have misleading names, comments can be different and especially the NatSpec comments could have been modified.

get_chains() → Dict[str, Any]

get_contract_metadata(*contract_address: str*) → *ContractMetadata* | None

is_chain_supported(*chain_id: int*) → bool

exception gnosis.eth.clients.**SourcifyClientConfigurationProblem**

Bases: `Exception`

exception gnosis.eth.clients.**SourcifyClientException**

Bases: `Exception`

gnosis.eth.contracts package

Module contents

Safe Addresses. Should be the same for every chain except for the ones with *chainId* protection. Check: https://github.com/safe-global/safe-deployments/tree/main/src/assets

Safe V1.4.1: 0x41675C099F32341bf84BFc5382aF534df5C7461a GnosisSafe V1.3.0: 0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552 GnosisSafe V1.1.1: 0x34CfAC646f301356fAa8B21e94227e3583Fe3F5F

GnosisSafe V1.1.0: 0xaE32496491b53841efb51829d6f886387708F99B GnosisSafe V1.0.0: 0xb6029EA3B2c51D09a50B53CA8012FeEB05bDa35A

Factories SafeProxyFactory V1.4.1: 0x4e1DCf7AD4e460CfD30791CCC4F9c8a4f820ec67 ProxyFactory V1.3.0: 0xa6B71E26C5e0845f74c812102Ca7114b6a896AB2 ProxyFactory V1.1.0: 0x50e55Af101C777bA7A1d560a774A82eF002ced9F ProxyFactory V1.0.0: 0x12302fE9c02ff50939BaAaaf415fc226C078613C

FallbackHandler CompatibilityFallBackHandler V1.4.1: 0xfd0732Dc9E303f09fCEf3a7388Ad10A83459Ec99 CompatibilityFallBackHandler V1.3.0: 0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4

Libraries CreateAndAddModules: 0x1a56aE690ab0818aF5cA349b7D21f1d7e76a3d36 MultiSend: 0x38869bf66a61cF6bDB996A6aE40D5853Fd43B526 MultiSendCallOnly: 0x9641d764fc13c8B624c04430C7356C1C7C8102e2 SimulateTxAccessor: 0x3d4BA2E0884aa488718476ca2FB8Efc291A46199 SignMessageLib: 0xd53cd0aB83D845Ac265BE939c57F53AD838012c9

gnosis.eth.contracts.**generate_contract_fn**(*contract: Dict[str, Any]*) → Callable[[Web3, ChecksumAddress | None], Contract]

Dynamically generate a function to build a Web3 Contract for the provided contract ABI

Parameters
contract –

Returns
function that will return a Web3 Contract from an ABI

gnosis.eth.contracts.**get_compatibility_fallback_handler_V1_3_0_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_compatibility_fallback_handler_V1_4_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_compatibility_fallback_handler_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

Parameters

- **w3** –
- **address** – Usually a Safe address

Returns
Latest available Compatibility Fallback handler contract

gnosis.eth.contracts.**get_cpk_factory_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_delegate_constructor_proxy_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_erc1155_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_erc20_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_erc721_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_example_erc20_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_kyber_network_proxy_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_multi_send_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_multicall_v3_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*)

gnosis.eth.contracts.**get_paying_proxy_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_paying_proxy_deployed_bytecode**() → bytes

gnosis.eth.contracts.**get_proxy_1_0_0_deployed_bytecode**() → bytes

gnosis.eth.contracts.**get_proxy_1_1_1_deployed_bytecode**() → bytes

gnosis.eth.contracts.**get_proxy_1_1_1_mainnet_deployed_bytecode**() → bytes

Somehow it's different from the generated version compiling the contracts

gnosis.eth.contracts.**get_proxy_1_3_0_deployed_bytecode**() → bytes

gnosis.eth.contracts.**get_proxy_1_4_1_deployed_bytecode**() → bytes

gnosis.eth.contracts.**get_proxy_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_proxy_factory_V1_0_0_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_proxy_factory_V1_1_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_proxy_factory_V1_3_0_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_proxy_factory_V1_4_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_proxy_factory_contract**(*w3: Web3*, *address: str | None = None*) → Contract

Parameters

- **w3** –
- **address** –

Returns

Latest available Safe Proxy Factory

gnosis.eth.contracts.**get_safe_V0_0_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_safe_V1_0_0_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_safe_V1_1_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_safe_V1_3_0_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_safe_V1_4_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_safe_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

Parameters

- **w3** –
- **address** –

Returns

Latest available Safe Contract

gnosis.eth.contracts.**get_sign_message_lib_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_simulate_tx_accessor_V1_4_1_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_uniswap_exchange_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_uniswap_factory_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_uniswap_v2_factory_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_uniswap_v2_pair_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**get_uniswap_v2_router_contract**(*w3: Web3*, *address: ChecksumAddress | None = None*) → Contract

gnosis.eth.contracts.**load_contract_interface**(*file_name: str*) → Dict[str, Any]

Parameters

file_name –

Returns

Get parsed JSON to ABI with the relative filename to this file path

gnosis.eth.django package

Subpackages

Submodules

gnosis.eth.django.filters module

gnosis.eth.django.models module

gnosis.eth.django.serializers module

class gnosis.eth.django.serializers.**EthereumAddressField**(**args*, ***kwargs*)

Bases: Field

Ethereum address checksumed https://github.com/ethereum/EIPs/blob/master/EIPS/eip-55.md

to_internal_value(*data*)

Transform the *incoming* primitive data into a native value.

to_representation(*obj*)

Transform the *outgoing* native value into primitive data.

class gnosis.eth.django.serializers.**HexadecimalField**(**args*, ***kwargs*)

Bases: Field

Serializes hexadecimal values starting by *0x*. Empty values should be None or just *0x*.

default_error_messages

to_internal_value(*data*)

Transform the *incoming* primitive data into a native value.

to_representation(*obj*)

Transform the *outgoing* native value into primitive data.

class gnosis.eth.django.serializers.**Sha3HashField**(**args*, ***kwargs*)

Bases: *HexadecimalField*

class gnosis.eth.django.serializers.**SignatureSerializer**(**args*, ***kwargs*)

Bases: Serializer

class gnosis.eth.django.serializers.**TransactionResponseSerializer**(**args*, ***kwargs*)

Bases: Serializer

Use chars to avoid problems with big ints (i.e. JavaScript)

get_fields()

Returns a dictionary of {field_name: field_instance}.

class gnosis.eth.django.serializers.**TransactionSerializer**(**args*, ***kwargs*)

Bases: Serializer

get_fields()

Returns a dictionary of {field_name: field_instance}.

gnosis.eth.django.validators module

gnosis.eth.django.validators.**validate_checksumed_address**(*address*)

Module contents

gnosis.eth.oracles package

Subpackages

gnosis.eth.oracles.abis package

Submodules

gnosis.eth.oracles.abis.aave_abis module

gnosis.eth.oracles.abis.balancer_abis module

gnosis.eth.oracles.abis.curve_abis module

gnosis.eth.oracles.abis.mooniswap_abis module

gnosis.eth.oracles.abis.yearn_abis module

Module contents

Submodules

gnosis.eth.oracles.oracles module

class gnosis.eth.oracles.oracles.**AaveOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PriceOracle*

get_price(*token_address: str*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.oracles.**BalancerOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PricePoolOracle*

Oracle for Balancer. More info on https://balancer.exchange

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate balancer pool token price based on its components

Parameters
pool_token_address – Balancer pool token address

Returns
Eth price for pool token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.oracles.**BaseOracle**

Bases: ABC

abstract classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.oracles.**ComposedPriceOracle**

Bases: *BaseOracle*

abstract get_underlying_tokens(**args*) → List[Tuple[*UnderlyingToken*]]

class gnosis.eth.oracles.oracles.**CreamOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PriceOracle*

get_price(*token_address: str*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.oracles.**CurveOracle**(*ethereum_client:* EthereumClient, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

Curve pool Oracle. More info on https://curve.fi/

ZERION_ADAPTER_ADDRESS = '0x99b0bEadc3984eab9842AF81f9fad0C2219108cc'

get_underlying_tokens(*token_address: ChecksumAddress*) → List[*UnderlyingToken*]

Check if passed token address is a Curve gauge deposit token, if it's a gauge we replace the address with the corresponding LP token address More info on https://resources.curve.fi/base-features/understanding-gauges

class gnosis.eth.oracles.oracles.**EnzymeOracle**(*ethereum_client: EthereumClient*, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

Enzyme pool Oracle. More info on https://enzyme.finance/

ZERION_ADAPTER_ADDRESS = '0x9e71455D748C23566b19493D09435574097C7D67'

class gnosis.eth.oracles.oracles.**MooniswapOracle**(*ethereum_client: EthereumClient*, *price_oracle: PriceOracle*)

Bases: *BalancerOracle*

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate balancer pool token price based on its components

Parameters
pool_token_address – Moniswap pool token address

Returns
Eth price for pool token

Raises
CannotGetPriceFromOracle

class gnosis.eth.oracles.oracles.**PoolTogetherOracle**(*ethereum_client: EthereumClient*, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

PoolTogether pool Oracle. More info on https://pooltogether.com/

ZERION_ADAPTER_ADDRESS = '0xb4E0E1672fFd9b128784dB9f3BE9158fac3f1DFc'

class gnosis.eth.oracles.oracles.**PriceOracle**

Bases: *BaseOracle*

abstract get_price(**args*) → float

class gnosis.eth.oracles.oracles.**PricePoolOracle**

Bases: *BaseOracle*

abstract get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

class gnosis.eth.oracles.oracles.**UnderlyingToken**(*address: eth_typing.evm.ChecksumAddress*, *quantity: int*)

Bases: object

address: ChecksumAddress

quantity: int

class gnosis.eth.oracles.oracles.**UniswapOracle**(*ethereum_client: EthereumClient*, *uniswap_factory_address: str | None = None*)

Bases: *PriceOracle*

Uniswap V1 Oracle

https://docs.uniswap.org/protocol/V1/guides/connect-to-uniswap

ADDRESSES = {EthereumNetwork.GOERLI: '0x6Ce570d02D73d4c384b46135E87f8C592A8c86dA', EthereumNetwork.MAINNET: '0xc0a47dFe034B400B47bDaD5FecDa2621de6c4d95', EthereumNetwork.RINKEBY: '0xf5D915570BC477f9B8D6C0E980aA81757A3AaC36', EthereumNetwork.ROPSTEN: '0x9c83dCE8CA20E9aAF9D3efc003b2ea62aBC08351'}

get_price(*token_address: str*) → float

get_uniswap_exchange(*token_address: str*) → str

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property uniswap_factory

property uniswap_factory_address

class gnosis.eth.oracles.oracles.**UniswapV2Oracle**(*ethereum_client:* EthereumClient, *router_address: str | None = None*)

Bases: *PricePoolOracle*, *PriceOracle*

PAIR_INIT_CODE = HexBytes('0x96e8ac4277198ff8b6f785478aa9a39f403cb768dd02cbee326c3e7da348845f')

ROUTER_ADDRESSES = {EthereumNetwork.MAINNET: '0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D'}

calculate_pair_address(*token_address: str*, *token_address_2: str*)

Calculate pair address without querying blockchain. https://uniswap.org/docs/v2/smart-contract-integration/getting-pair-addresses/#docs-header

Parameters

- **token_address** –
- **token_address_2** –

Returns
Checksummed address for token pair. It could be not created yet

property factory

property factory_address: str

Returns
Uniswap factory checksummed address

Raises
Web3Exception: If router contract is not deployed

get_pair_address(*token_address: str*, *token_address_2: str*) → str | None

Get uniswap pair address. *token_address* and *token_address_2* are interchangeable. https://uniswap.org/docs/v2/smart-contracts/factory/

Parameters

- **token_address** –

- **token_address_2** –

Returns
Address of the pair for *token_address* and *token_address_2*, if it has been created, else *None*.

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate pool token price based on its components

Parameters
pool_token_address –

Returns
Pool token eth price per unit (total pool token supply / 1e18)

Raises
CannotGetPriceFromOracle

get_price(*token_address: str*, *token_address_2: str | None = None*) → float

get_price_without_exception(*token_address: str*, *token_address_2: str | None = None*) → float

Parameters

- **token_address** –
- **token_address_2** –

Returns
Call *get_price*, return 0. instead on an exception if there's any issue

get_reserves(*pair_address: str*) → Tuple[int, int]

Returns the number of tokens in the pool. *getReserves()* also returns the block.timestamp (mod 2**32) of the last block during which an interaction occurred for the pair, but it's ignored. https://uniswap.org/docs/v2/smart-contracts/pair/

Returns
Reserves of *token_address* and *token_address_2* used to price trades and distribute liquidity.

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property weth_address: str

Returns
Wrapped ether checksummed address

Raises
Web3Exception: If router contract is not deployed

class gnosis.eth.oracles.oracles.**YearnOracle**(*ethereum_client:* EthereumClient, *yearn_vault_token_adapter: str | None = '0xb460FcC1B6c1CBD7D03F47B6BD5F03994d286c75'*, *iearn_token_adapter: str | None = '0x65B23774daE2a5be02dD275918DDF048d177a5B4'*)

Bases: *ComposedPriceOracle*

Yearn oracle. More info on https://docs.yearn.finance

get_underlying_tokens(*token_address: ChecksumAddress*) → List[Tuple[float, ChecksumAddress]]

Parameters
token_address –

Returns
Price per share and underlying token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.oracles.**ZerionComposedOracle**(*ethereum_client:* EthereumClient, *zerion_adapter_address: str | None = None*)

Bases: *ComposedPriceOracle*

ZERION_ADAPTER_ADDRESS = None

get_underlying_tokens(*token_address: ChecksumAddress*) → List[*UnderlyingToken*]

Use Zerion Token adapter to return underlying components for pool

Parameters
token_address – Pool token address

Returns
Price per share and underlying token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property zerion_adapter_contract: Contract | None

Returns
https://curve.readthedocs.io/registry-registry.html

Module contents

class gnosis.eth.oracles.**AaveOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PriceOracle*

get_price(*token_address: str*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**BalancerOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PricePoolOracle*

Oracle for Balancer. More info on https://balancer.exchange

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate balancer pool token price based on its components

Parameters
pool_token_address – Balancer pool token address

Returns
Eth price for pool token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**ComposedPriceOracle**

Bases: *BaseOracle*

abstract get_underlying_tokens(**args*) → List[Tuple[*UnderlyingToken*]]

class gnosis.eth.oracles.**CowswapOracle**(*ethereum_client:* EthereumClient)

Bases: *PriceOracle*

CowSwap Oracle implementation

https://docs.cow.fi/off-chain-services/api

get_price(*token_address_1: str*, *token_address_2: str | None = None*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if CowSwap is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**CreamOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *PriceOracle*

get_price(*token_address: str*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**CurveOracle**(*ethereum_client:* EthereumClient, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

Curve pool Oracle. More info on https://curve.fi/

ZERION_ADAPTER_ADDRESS = '0x99b0bEadc3984eab9842AF81f9fad0C2219108cc'

get_underlying_tokens(*token_address: ChecksumAddress*) → List[*UnderlyingToken*]

Check if passed token address is a Curve gauge deposit token, if it's a gauge we replace the address with the corresponding LP token address More info on https://resources.curve.fi/base-features/understanding-gauges

class gnosis.eth.oracles.**EnzymeOracle**(*ethereum_client:* EthereumClient, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

Enzyme pool Oracle. More info on https://enzyme.finance/

ZERION_ADAPTER_ADDRESS = '0x9e71455D748C23566b19493D09435574097C7D67'

class gnosis.eth.oracles.**KyberOracle**(*ethereum_client:* EthereumClient, *kyber_network_proxy_address: str | None = None*)

Bases: *PriceOracle*

KyberSwap Legacy Oracle

https://docs.kyberswap.com/Legacy/addresses/addresses-mainnet

ADDRESSES = {EthereumNetwork.MAINNET: '0x9AAb3f75489902f3a48495025729a0AF77d4b11e',
EthereumNetwork.RINKEBY: '0x0d5371e5EE23dec7DF251A8957279629aa79E9C5',
EthereumNetwork.ROPSTEN: '0xd719c34261e099Fdb33030ac8909d5788D3039C4'}

ETH_TOKEN_ADDRESS = '0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE'

get_price(*token_address_1: str*, *token_address_2: str = '0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE'*) → float

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property kyber_network_proxy_address

property kyber_network_proxy_contract

class gnosis.eth.oracles.**MooniswapOracle**(*ethereum_client:* EthereumClient, *price_oracle:* PriceOracle)

Bases: *BalancerOracle*

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate balancer pool token price based on its components

Parameters
pool_token_address – Moniswap pool token address

Returns
Eth price for pool token

Raises
CannotGetPriceFromOracle

class gnosis.eth.oracles.**PoolTogetherOracle**(*ethereum_client: EthereumClient*, *zerion_adapter_address: str | None = None*)

Bases: *ZerionComposedOracle*

PoolTogether pool Oracle. More info on https://pooltogether.com/

ZERION_ADAPTER_ADDRESS = '0xb4E0E1672fFd9b128784dB9f3BE9158fac3f1DFc'

class gnosis.eth.oracles.**PriceOracle**

Bases: *BaseOracle*

abstract get_price(**args*) → float

class gnosis.eth.oracles.**PricePoolOracle**

Bases: *BaseOracle*

abstract get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

class gnosis.eth.oracles.**SuperfluidOracle**(*ethereum_client: EthereumClient*, *price_oracle: PriceOracle*)

Bases: *PriceOracle*

get_price(*token_address: str*) → float

classmethod is_available(*ethereum_client: EthereumClient*) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**SushiswapOracle**(*ethereum_client: EthereumClient*, *router_address: str | None = None*)

Bases: *UniswapV2Oracle*

PAIR_INIT_CODE = HexBytes('0xe18a34eb0e04b04f7a0ac29a6e80748dca96319b42c54d679cb821dca90c6303')

```
ROUTER_ADDRESSES = {EthereumNetwork.ARBITRUM_ONE:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.AVALANCHE_C_CHAIN:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506',
EthereumNetwork.BNB_SMART_CHAIN_MAINNET:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.CELO_MAINNET:
'0x1421bDe4B10e8dd459b3BCb598810B1337D56842', EthereumNetwork.FANTOM_OPERA:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.FUSE_MAINNET:
'0xF4d73326C13a4Fc5FD7A064217e12780e9Bd62c3', EthereumNetwork.GNOSIS:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506',
EthereumNetwork.HUOBI_ECO_CHAIN_MAINNET:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.MAINNET:
'0xd9e1cE17f2641f24aE83637ab66a2cca9C378B9F', EthereumNetwork.MOONBEAM:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.MOONRIVER:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.OKXCHAIN_MAINNET:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.PALM:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506', EthereumNetwork.POLYGON:
'0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506'}
```

class gnosis.eth.oracles.**UnderlyingToken**(*address: eth_typing.evm.ChecksumAddress*, *quantity: int*)

Bases: object

address: ChecksumAddress

quantity: int

class gnosis.eth.oracles.**UniswapOracle**(*ethereum_client:* EthereumClient, *uniswap_factory_address: str | None = None*)

Bases: *PriceOracle*

Uniswap V1 Oracle

https://docs.uniswap.org/protocol/V1/guides/connect-to-uniswap

```
ADDRESSES = {EthereumNetwork.GOERLI: '0x6Ce570d02D73d4c384b46135E87f8C592A8c86dA',
EthereumNetwork.MAINNET: '0xc0a47dFe034B400B47bDaD5FecDa2621de6c4d95',
EthereumNetwork.RINKEBY: '0xf5D915570BC477f9B8D6C0E980aA81757A3AaC36',
EthereumNetwork.ROPSTEN: '0x9c83dCE8CA20E9aAF9D3efc003b2ea62aBC08351'}
```

get_price(*token_address: str*) → float

get_uniswap_exchange(*token_address: str*) → str

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property uniswap_factory

property uniswap_factory_address

class gnosis.eth.oracles.**UniswapV2Oracle**(*ethereum_client:* EthereumClient, *router_address: str | None = None*)

Bases: *PricePoolOracle*, *PriceOracle*

```
PAIR_INIT_CODE =
HexBytes('0x96e8ac4277198ff8b6f785478aa9a39f403cb768dd02cbee326c3e7da348845f')

ROUTER_ADDRESSES = {EthereumNetwork.MAINNET:
'0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D'}
```

calculate_pair_address(*token_address: str*, *token_address_2: str*)

Calculate pair address without querying blockchain. https://uniswap.org/docs/v2/smart-contract-integration/getting-pair-addresses/#docs-header

Parameters

- **token_address** –
- **token_address_2** –

Returns
Checksummed address for token pair. It could be not created yet

property factory

property factory_address: str

Returns
Uniswap factory checksummed address

Raises
Web3Exception: If router contract is not deployed

get_pair_address(*token_address: str*, *token_address_2: str*) → str | None

Get uniswap pair address. *token_address* and *token_address_2* are interchangeable. https://uniswap.org/docs/v2/smart-contracts/factory/

Parameters

- **token_address** –
- **token_address_2** –

Returns
Address of the pair for *token_address* and *token_address_2*, if it has been created, else *None*.

get_pool_token_price(*pool_token_address: ChecksumAddress*) → float

Estimate pool token price based on its components

Parameters
pool_token_address –

Returns
Pool token eth price per unit (total pool token supply / 1e18)

Raises
CannotGetPriceFromOracle

get_price(*token_address: str*, *token_address_2: str | None = None*) → float

get_price_without_exception(*token_address: str*, *token_address_2: str | None = None*) → float

Parameters

- **token_address** –
- **token_address_2** –

Returns
Call *get_price*, return 0. instead on an exception if there's any issue

`get_reserves`(*pair_address: str*) → Tuple[int, int]

Returns the number of tokens in the pool. *getReserves()* also returns the block.timestamp (mod 2**32) of the last block during which an interaction occurred for the pair, but it's ignored. https://uniswap.org/docs/v2/smart-contracts/pair/

Returns
Reserves of *token_address* and *token_address_2* used to price trades and distribute liquidity.

`classmethod is_available`(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

`router_address: str`

`property weth_address: str`

Returns
Wrapped ether checksummed address

Raises
Web3Exception: If router contract is not deployed

`class gnosis.eth.oracles.UniswapV3Oracle`(*ethereum_client:* EthereumClient, *uniswap_v3_router_address: ChecksumAddress | None = None*)

Bases: *PriceOracle*

`DEFAULT_ROUTER_ADDRESS = '0x68b3465833fb72A70ecDF485E0e4C7bD8665Fc45'`

`PRICE_CONVERSION_CONSTANT = 6277101735386680763835789423207666416102355444464034512896`

`ROUTER_ADDRESSES = {EthereumNetwork.CELO_MAINNET: '0x5615CDAb10dc425a742d643d949a7F474C01abc4', EthereumNetwork.MAINNET: '0x68b3465833fb72A70ecDF485E0e4C7bD8665Fc45'}`

`get_factory`() → Contract

Factory contract creates the pools for token pairs

Returns
Uniswap V3 Factory Contract

`get_pool_address`(*token_address: str*, *token_address_2: str*, *fee: int | None = 3000*) → ChecksumAddress | None

Get pool address for tokens with a given fee (by default, 0.3)

Parameters

- **token_address** –
- **token_address_2** –
- **fee** – Uniswap V3 uses 0.3 as the default fee

Returns
Pool address

get_price(*token_address: str*, *token_address_2: str | None = None*) → float

Parameters

- **token_address** –
- **token_address_2** –

Returns
price for *token_address* related to *token_address_2*. If *token_address_2* is not provided, *Wrapped Eth* address will be used

classmethod is_available(*ethereum_client:* EthereumClient, *uniswap_v3_router_address: ChecksumAddress | None = None*) → bool

Parameters

- **ethereum_client** –
- **uniswap_v3_router_address** – Provide a custom *SwapRouter02* address

Returns
True if Uniswap V3 is available for the EthereumClient provided, *False* otherwise

property router: Contract

Router knows about the *Uniswap Factory* and *Wrapped Eth* addresses for the network

Returns
Uniswap V3 Router Contract

property weth_address: ChecksumAddress

Returns
Wrapped ether checksummed address

class gnosis.eth.oracles.**YearnOracle**(*ethereum_client:* EthereumClient, *yearn_vault_token_adapter: str | None = '0xb460FcC1B6c1CBD7D03F47B6BD5F03994d286c75'*, *iearn_token_adapter: str | None = '0x65B23774daE2a5be02dD275918DDF048d177a5B4'*)

Bases: *ComposedPriceOracle*

Yearn oracle. More info on https://docs.yearn.finance

get_underlying_tokens(*token_address: ChecksumAddress*) → List[Tuple[float, ChecksumAddress]]

Parameters
token_address –

Returns
Price per share and underlying token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

class gnosis.eth.oracles.**ZerionComposedOracle**(*ethereum_client:* EthereumClient, *zerion_adapter_address: str | None = None*)

Bases: *ComposedPriceOracle*

ZERION_ADAPTER_ADDRESS = None

get_underlying_tokens(*token_address: ChecksumAddress*) → List[*UnderlyingToken*]

Use Zerion Token adapter to return underlying components for pool

Parameters
token_address – Pool token address

Returns
Price per share and underlying token

Raises
CannotGetPriceFromOracle

classmethod is_available(*ethereum_client:* EthereumClient) → bool

Parameters
ethereum_client –

Returns
True if Oracle is available for the EthereumClient provided, *False* otherwise

property zerion_adapter_contract: Contract | None

Returns
https://curve.readthedocs.io/registry-registry.html

Submodules

gnosis.eth.constants module

gnosis.eth.ethereum_client module

class gnosis.eth.ethereum_client.**BatchCallManager**(*ethereum_client:* EthereumClient)

Bases: *EthereumClientManager*

batch_call(*contract_functions: Iterable[ContractFunction]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[Any | None]

Do batch requests of multiple contract calls

Parameters

- **contract_functions** – Iterable of contract functions using web3.py contracts. For instance, a valid argument would be [erc20_contract.functions.balanceOf(address), erc20_contract.functions.decimals()]
- **from_address** – Use this address as *from* in every call if provided
- **block_identifier** – *latest* by default
- **raise_exception** – If False, exception will not be raised if there's any problem and instead *None* will be returned as the value.

Returns
List with the ABI decoded return values

batch_call_custom(*payloads: Iterable[Dict[str, Any]]*, *raise_exception: bool = True*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*, *batch_size: int | None = None*) → List[Any | None]

Do batch requests of multiple contract calls (*eth_call*)

Parameters

- **payloads** – Iterable of Dictionaries with at least {'data': '<hex-string>', 'output_type': <solidity-output-type>, 'to': '<checksummed-address>'}. *from* can also be provided and if *fn_name* is provided it will be used for debugging purposes
- **raise_exception** – If False, exception will not be raised if there's any problem and instead *None* will be returned as the value
- **block_identifier** – *latest* by default
- **batch_size** – If *payload* length is bigger than size, it will be split into smaller chunks before sending to the server

Returns
List with the ABI decoded return values

Raises
ValueError if raise_exception=True

batch_call_same_function(*contract_function: ContractFunction*, *contract_addresses: Sequence[ChecksumAddress]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[Any | None]

Do batch requests using the same function to multiple address. `batch_call` could be used to achieve that, but generating the ContractFunction is slow, so this function allows to use the same contract_function for multiple addresses

Parameters

- **contract_function** –
- **contract_addresses** –
- **from_address** –
- **raise_exception** –
- **block_identifier** –

Returns

class gnosis.eth.ethereum_client.**Erc20Info**(*name*, *symbol*, *decimals*)

Bases: `NamedTuple`

decimals: int
Alias for field number 2

name: str
Alias for field number 0

symbol: str
Alias for field number 1

class gnosis.eth.ethereum_client.**Erc20Manager**(*ethereum_client:* EthereumClient)

Bases: *EthereumClientManager*

Manager for ERC20 operations

TRANSFER_TOPIC = HexBytes('0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef')

decode_logs(*logs: Sequence[LogReceipt]*)

get_balance(*address: ChecksumAddress*, *token_address: ChecksumAddress*) → int

Get balance of address for *erc20_address*

Parameters

- **address** – owner address
- **token_address** – erc20 token address

Returns
balance

get_balances(*address: ChecksumAddress*, *token_addresses: Sequence[ChecksumAddress]*) → List[*BalanceDict*]

Get balances for Ether and tokens for an *address*

Parameters

- **address** – Owner address checksummed
- **token_addresses** – token addresses to check

Returns
`List[BalanceDict]`

get_decimals(*erc20_address: ChecksumAddress*) → int

get_info(*erc20_address: ChecksumAddress*) → *Erc20Info*

Get erc20 information (*name*, *symbol* and *decimals*). Use batching to get all info in the same request.

Parameters
erc20_address –

Returns
Erc20Info

Raises
InvalidERC20Info

get_name(*erc20_address: ChecksumAddress*) → str

get_symbol(*erc20_address: ChecksumAddress*) → str

get_total_transfer_history(*addresses: Sequence[ChecksumAddress] | None = None*, *from_block: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int = 0*, *to_block: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = None*, *token_address: ChecksumAddress | None = None*) → List[*LogReceiptDecoded*]

Get events for erc20 and erc721 transfers from and to an *address*. We decode it manually. Example of an erc20 event:

```
{'logIndex': 0,
 'transactionIndex': 0,
 'transactionHash': HexBytes(
↪'0x4d0f25313603e554e3b040667f7f391982babbd195c7ae57a8c84048189f7794'),
 'blockHash': HexBytes(
↪'0x90fa67d848a0eaf3be625235dae28815389f5292d4465c48d1139f0c207f8d42'),
 'blockNumber': 791,
 'address': '0xf7d0Bd47BF3214494E7F5B40E392A25cb4788620',
 'data': '0x0000000000000000000000000000000000000000000000002001f716742000',
 'topics': [HexBytes(
↪'0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef'),
  HexBytes('0x000000000000000000000000f5984365fca2e3bc7d2e020abb2c701df9070eb7
↪'),
  HexBytes('0x0000000000000000000000001df62f291b2e969fb0849d99d9ce41e2f137006e
↪')],
 'type': 'mined'
 'args': {'from': '0xf5984365FcA2e3bc7D2E020AbB2c701DF9070eB7',
          'to': '0x1dF62f291b2E969fB0849d99D9Ce41e2F137006e',
          'value': 90093600000000000
         }
}
An example of an erc721 event
{'address': '0x6631FcbB50677DfC6c02CCDcc03a8f68Db427a64',
 'blockHash': HexBytes(
↪'0x95c71c6c9373e9a8ca2c767dda1cd5083eb6addcce36fc216c9e1f458d6970f9'),
 'blockNumber': 5341681,
 'data': '0x',
 'logIndex': 0,
 'removed': False,
 'topics': [HexBytes(
↪'0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef'),
  HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000
↪'),
  HexBytes('0x000000000000000000000000b5239c032ab9fb5abfc3903e770a4b6a9095542c
↪'),
  HexBytes('0x0000000000000000000000000000000000000000000000000000000000000063
↪')],
 'transactionHash': HexBytes(
↪'0xce8c8af0503e6f8a421345c10cdf92834c95186916a3f5b1437d2bba63d2db9e'),
 'transactionIndex': 0,
 'transactionLogIndex': '0x0',
 'type': 'mined',
 'args': {'from': '0x0000000000000000000000000000000000000000',
          'to': '0xb5239C032AB9fB5aBFc3903e770A4B6a9095542C',
          'tokenId': 99
         }
 }
An example of unknown transfer event (no indexed parts), could be a ERC20 or␣
↪ERC721 transfer:
{'address': '0x6631FcbB50677DfC6c02CCDcc03a8f68Db427a64',
 'blockHash': HexBytes(
↪'0x95c71c6c9373e9a8ca2c767dda1cd5083eb6addcce36fc216c9e1f458d6970f9'),
 'blockNumber': 5341681,
```

(continues on next page)

(continued from previous page)

```
 'data': '0x',
 'logIndex': 0,
 'removed': False,
 'topics': [HexBytes(
↪'0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef'),
  HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000
↪'),
  HexBytes('0x000000000000000000000000b5239c032ab9fb5abfc3903e770a4b6a9095542c
↪'),
  HexBytes('0x0000000000000000000000000000000000000000000000000000000000000063
↪')],
 'transactionHash': HexBytes(
↪'0xce8c8af0503e6f8a421345c10cdf92834c95186916a3f5b1437d2bba63d2db9e'),
 'transactionIndex': 0,
 'transactionLogIndex': '0x0',
 'type': 'mined',
 'args': {'from': '0x0000000000000000000000000000000000000000',
          'to': '0xb5239C032AB9fB5aBFc3903e770A4B6a9095542C',
          'unknown': 99
         }
 }
```

Parameters

- **addresses** – Search events *from* and *to* these *addresses*. If not, every transfer event within the range will be retrieved
- **from_block** – Block to start querying from
- **to_block** – Block to stop querying from
- **token_address** – Address of the token

Returns

List of events sorted by blockNumber

get_transfer_history(*from_block: int*, *to_block: int | None = None*, *from_address: str | None = None*, *to_address: str | None = None*, *token_address: str | None = None*) → List[Dict[str, Any]]

DON'T USE, it will fail in some cases until they fix https://github.com/ethereum/web3.py/issues/1351 Get events for erc20/erc721 transfers. At least one of *from_address*, *to_address* or *token_address* must be defined. Example of decoded event:

```
{
    "args": {
        "from": "0x1Ce67Ea59377A163D47DFFc9BaAB99423BE6EcF1",
        "to": "0xaE9E15896fd32E59C7d89ce7a95a9352D6ebD70E",
        "value": 150000000000000000
    },
    "event": "Transfer",
    "logIndex": 42,
    "transactionIndex": 60,
    "transactionHash":
↪"0x71d6d83fef3347bad848e83dfa0ab28296e2953de946ee152ea81c6dfb42d2b3",
```

(continues on next page)

(continued from previous page)

```
    "address": "0xfecA834E7da9D437645b474450688DA9327112a5",
    "blockHash":
↪"0x054de9a496fc7d10303068cbc7ee3e25181a3b26640497859a5e49f0342e7db2",
    "blockNumber": 7265022
}
```

Parameters

- **from_block** – Block to start querying from
- **to_block** – Block to stop querying from
- **from_address** – Address sending the erc20 transfer
- **to_address** – Address receiving the erc20 transfer
- **token_address** – Address of the token

Returns
List of events (decoded)

Throws
ReadTimeout

send_tokens(*to: str*, *amount: int*, *erc20_address: ChecksumAddress*, *private_key: str*, *nonce: int | None = None*, *gas_price: int | None = None*, *gas: int | None = None*) → bytes

Send tokens to address

Parameters

- **to** –
- **amount** –
- **erc20_address** –
- **private_key** –
- **nonce** –
- **gas_price** –
- **gas** –

Returns
tx_hash

class gnosis.eth.ethereum_client.**Erc721Info**(*name*, *symbol*)

Bases: NamedTuple

name: str

Alias for field number 0

symbol: str

Alias for field number 1

class gnosis.eth.ethereum_client.**Erc721Manager**(*ethereum_client: EthereumClient*)

Bases: *EthereumClientManager*

TRANSFER_TOPIC = HexBytes('0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef')

get_balance(*address: ChecksumAddress*, *token_address: ChecksumAddress*) → int

Get balance of address for *erc20_address*

Parameters

- **address** – owner address
- **token_address** – erc721 token address

Returns
balance

get_balances(*address: ChecksumAddress*, *token_addresses: Sequence[ChecksumAddress]*) → List[*TokenBalance*]

Get balances for tokens for an *address*. If there's a problem with a token_address *0* will be returned for balance

Parameters

- **address** – Owner address checksummed
- **token_addresses** – token addresses to check

Returns

get_info(*token_address: ChecksumAddress*) → *Erc721Info*

Get erc721 information (*name*, *symbol*). Use batching to get all info in the same request.

Parameters
token_address –

Returns
Erc721Info

get_owners(*token_addresses_with_token_ids: Sequence[Tuple[ChecksumAddress, int]]*) → List[ChecksumAddress | None]

Parameters
token_addresses_with_token_ids – Tuple(token_address: str, token_id: int)

Returns
List of owner addresses, *None* if not found

get_token_uris(*token_addresses_with_token_ids: Sequence[Tuple[ChecksumAddress, int]]*) → List[str | None]

Parameters
token_addresses_with_token_ids – Tuple(token_address: str, token_id: int)

Returns
List of token_uris, *None* if not found

class gnosis.eth.ethereum_client.**EthereumClient**(*ethereum_node_url: URI = 'http://localhost:8545'*, *provider_timeout: int = 15*, *slow_provider_timeout: int = 60*, *retry_count: int = 1*, *use_caching_middleware: bool = True*, *batch_request_max_size: int = 500*)

Bases: object

Manage ethereum operations. Uses web3 for the most part, but some other stuff is implemented from scratch. Note: If you want to use *pending* state with *Parity*, it must be run with *–pruning=archive* or *–force-sealing*

NULL_ADDRESS = '0x0000000000000000000000000000000000000000'

batch_call(*contract_functions: Iterable[ContractFunction]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *force_batch_call: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[bytes | Any | None]

Call multiple functions. `Multicall` contract by MakerDAO will be used by default if available

Parameters

- **contract_functions** –
- **from_address** – Only available when `Multicall` is not used
- **raise_exception** – If `True`, raise `BatchCallException` if one of the calls fails
- **force_batch_call** – If `True`, ignore multicall and always use batch calls to get the result (less optimal). If `False`, more optimal way will be tried.
- **block_identifier** –

Returns

List of elements decoded to their types, `None` if they cannot be decoded and bytes if a revert error is returned and `raise_exception=False`

Raises

BatchCallException

batch_call_same_function(*contract_function: ContractFunction*, *contract_addresses: Sequence[ChecksumAddress]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *force_batch_call: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[bytes | Any | None]

Call the same function in multiple contracts. Way more optimal than using `batch_call` generating multiple `ContractFunction` objects.

Parameters

- **contract_function** –
- **contract_addresses** –
- **from_address** – Only available when `Multicall` is not used
- **raise_exception** – If `True`, raise `BatchCallException` if one of the calls fails
- **force_batch_call** – If `True`, ignore multicall and always use batch calls to get the result (less optimal). If `False`, more optimal way will be tried.
- **block_identifier** –

Returns

List of elements decoded to the same type, `None` if they cannot be decoded and bytes if a revert error is returned and `raise_exception=False`

Raises

BatchCallException

static build_tx_params(*from_address: ChecksumAddress | None = None*, *to_address: ChecksumAddress | None = None*, *value: int | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*, *chain_id: int | None = None*, *tx_params: TxParams | None = None*) → TxParams

Build tx params dictionary. If an existing TxParams dictionary is provided the fields will be replaced by the provided ones

Parameters

- **from_address** –
- **to_address** –
- **value** –
- **gas** –
- **gas_price** –
- **nonce** –
- **chain_id** –
- **tx_params** – An existing TxParams dictionary will be replaced by the providen values

Returns

check_tx_with_confirmations(*tx_hash: Hash32 | HexBytes | HexStr*, *confirmations: int*) → bool

Check tx hash and make sure it has the confirmations required

Parameters

- **tx_hash** – Hash of the tx
- **confirmations** – Minimum number of confirmations required

Returns

True if tx was mined with the number of confirmations required, False otherwise

property current_block_number

deploy_and_initialize_contract(*deployer_account: LocalAccount*, *constructor_data: bytes*, *initializer_data: bytes = b''*, *check_receipt: bool = True*) → *EthereumTxSent*

static estimate_data_gas(*data: bytes*)

Estimate gas costs only for "storage" of the `data` bytes provided

Parameters

data –

Returns

estimate_fee_eip1559(*tx_speed:* TxSpeed = *TxSpeed.NORMAL*) → Tuple[int, int]

Check https://github.com/ethereum/execution-apis/blob/main/src/eth/fee_market.json#L15

Returns

Tuple[BaseFeePerGas, MaxPriorityFeePerGas]

Raises

ValueError if not supported on the network

estimate_gas(*to: str*, *from_: str | None = None*, *value: int | None = None*, *data: bytes | HexStr | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = None*) → int

Estimate gas calling *eth_estimateGas*

Parameters

- **from** –
- **to** –
- **value** –
- **data** –
- **gas** –
- **gas_price** –
- **block_identifier** – Be careful, *Geth* does not support *pending* when estimating

Returns
Amount of gas needed for transaction

Raises
ValueError

get_balance(*address: ChecksumAddress*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = None*)

get_block(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int*, *full_transactions: bool = False*) → BlockData | None

get_blocks(*block_identifiers: Iterable[Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int]*, *full_transactions: bool = False*) → List[BlockData | None]

get_chain_id() → int

Returns
ChainId returned by the RPC *eth_chainId* method. It should never change, so it's cached.

get_client_version() → str

Returns
RPC version information

get_network() → *EthereumNetwork*

Get network name based on the chainId. This method is not cached as the method for getting the *chainId* already is.

Returns
EthereumNetwork based on the chainId. If network is not on our list, *EthereumNetwork.UNKNOWN* is returned

get_nonce_for_account(*address: ChecksumAddress*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*)

Get nonce for account. *getTransactionCount* is the only method for what *pending* is currently working (Geth and Parity)

Parameters

- **address** –
- **block_identifier** –

Returns

get_transaction(*tx_hash: Hash32 | HexBytes | HexStr*) → TxData | None

get_transaction_receipt(*tx_hash: Hash32 | HexBytes | HexStr*, *timeout=None*) → TxReceipt | None

get_transaction_receipts(*tx_hashes: Sequence[bytes | HexStr]*) → List[TxReceipt | None]

get_transactions(*tx_hashes: Sequence[Hash32 | HexBytes | HexStr]*) → List[TxData | None]

is_contract(*contract_address: ChecksumAddress*) → bool

is_eip1559_supported() → bool

Returns

True if EIP1559 is supported by the node, *False* otherwise

property multicall: Multicall

static private_key_to_address(*private_key*)

raw_batch_request(*payload: Sequence[Dict[str, Any]]*, *batch_size: int | None = None*) → Iterable[Dict[str, Any] | None]

Perform a raw batch JSON RPC call

Parameters

- **payload** – Batch request payload. Make sure all provided *ids* inside the payload are different
- **batch_size** – If *payload* length is bigger than size, it will be split into smaller chunks before sending to the server

Returns

Raises

ValueError

send_eth_to(*private_key: str*, *to: str*, *gas_price: int*, *value: Wei*, *gas: int | None = None*, *nonce: int | None = None*, *retry: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'pending'*) → bytes

Send ether using configured account

Parameters

- **private_key** – to
- **to** – to
- **gas_price** – gas_price
- **value** – value(wei)
- **gas** – gas, defaults to 22000
- **retry** – Retry if a problem is found
- **nonce** – Nonce of sender account
- **block_identifier** – Block identifier for nonce calculation

Returns
tx_hash

send_raw_transaction(*raw_transaction: bytes | HexStr*) → HexBytes

send_transaction(*transaction_dict: TxParams*) → HexBytes

send_unsigned_transaction(*tx: TxParams*, *private_key: str | None = None*, *public_key: str | None = None*, *retry: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'pending'*) → HexBytes

Send a tx using an unlocked public key in the node or a private key. Both *public_key* and *private_key* cannot be *None*

Parameters

- **tx** –
- **private_key** –
- **public_key** –
- **retry** – Retry if a problem with nonce is found
- **block_identifier** – For nonce calculation, recommended is *pending*

Returns
tx hash

set_eip1559_fees(*tx: TxParams*, *tx_speed:* TxSpeed *= TxSpeed.NORMAL*) → TxParams

Returns
TxParams in EIP1559 format

Raises
ValueError if EIP1559 not supported

class gnosis.eth.ethereum_client.**EthereumClientManager**(*ethereum_client:* EthereumClient)

Bases: object

class gnosis.eth.ethereum_client.**EthereumClientProvider**

Bases: object

class gnosis.eth.ethereum_client.**EthereumTxSent**(*tx_hash*, *tx*, *contract_address*)

Bases: NamedTuple

contract_address: ChecksumAddress | None

Alias for field number 2

tx: TxParams

Alias for field number 1

tx_hash: bytes

Alias for field number 0

class gnosis.eth.ethereum_client.**TokenBalance**(*token_address*, *balance*)

Bases: NamedTuple

balance: int

Alias for field number 1

token_address: str

Alias for field number 0

class gnosis.eth.ethereum_client.**TracingManager**(*ethereum_client:* EthereumClient)

Bases: *EthereumClientManager*

filter_out_errored_traces(*internal_txs: Sequence[Dict[str, Any]]*) → Sequence[Dict[str, Any]]

Filter out errored transactions (traces that are errored or that have an errored parent)

Parameters

internal_txs – Traces for the SAME ethereum tx, sorted ascending by *trace_address sorted(t, key = lambda i: i['traceAddress'])*. It's the default output from methods returning *traces* like *trace_block* or *trace_transaction*

Returns

List of not errored traces

get_next_traces(*tx_hash: Hash32 | HexBytes | HexStr*, *trace_address: Sequence[int]*, *remove_delegate_calls: bool = False*, *remove_calls: bool = False*) → List[FilterTrace]

Parameters

- **tx_hash** –
- **trace_address** –
- **remove_delegate_calls** – If True remove delegate calls from result
- **remove_calls** – If True remove calls from result

Returns

Children for a trace, E.g. if address is [0, 1] and number_traces = 1, it will return [0, 1, x]

Raises

ValueError if tracing is not supported

get_previous_trace(*tx_hash: Hash32 | HexBytes | HexStr*, *trace_address: Sequence[int]*, *number_traces: int = 1*, *skip_delegate_calls: bool = False*) → Dict[str, Any] | None

Parameters

- **tx_hash** –
- **trace_address** –
- **number_traces** – Number of traces to skip, by default get the immediately previous one
- **skip_delegate_calls** – If True filter out delegate calls

Returns

Parent trace for a trace

Raises

ValueError if tracing is not supported

trace_block(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int*) → List[BlockTrace]

trace_blocks(*block_identifiers: Sequence[Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int]*) → List[List[Dict[str, Any]]]

trace_filter(*from_block: int = 1*, *to_block: int | None = None*, *from_address: Sequence[ChecksumAddress] | None = None*, *to_address: Sequence[ChecksumAddress] | None = None*, *after: int | None = None*, *count: int | None = None*) → List[FilterTrace]

Get events using `trace_filter` method

Parameters

- **from_block** – Quantity or Tag - (optional) From this block. *0* is not working, it needs to be >= *1*
- **to_block** – Quantity or Tag - (optional) To this block.
- **from_address** – Array - (optional) Sent from these addresses.
- **to_address** – Address - (optional) Sent to these addresses.
- **after** – Quantity - (optional) The offset trace number
- **count** – Quantity - (optional) Integer number of traces to display in a batch.

Returns


```
[
    {
        "action": {
            "callType": "call",
            "from": "0x32be343b94f860124dc4fee278fdcbd38c102d88",
            "gas": "0x4c40d",
            "input": "0x",
            "to": "0x8bbb73bcb5d553b5a556358d27625323fd781d37",
            "value": "0x3f0650ec47fd240000"
        },
        "blockHash":
↪"0x86df301bcdd8248d982dbf039f09faf792684e1aeee99d5b58b77d620008b80f",
        "blockNumber": 3068183,
        "result": {
            "gasUsed": "0x0",
            "output": "0x"
        },
        "subtraces": 0,
        "traceAddress": [],
        "transactionHash":
↪"0x3321a7708b1083130bd78da0d62ead9f6683033231617c9d268e2c7e3fa6c104",
        "transactionPosition": 3,
        "type": "call"
    },
    {
        "action": {
            "from": "0x3b169a0fb55ea0b6bafe54c272b1fe4983742bf7",
            "gas": "0x49b0b",
            "init":
↪"0x60806040523480156100105760008fd5b506040516060a38038061060a8339810180604052810190808051906
↪",
            "value": "0x0"
        },
        "blockHash":
↪"0x03f9f64dfeb7807b5df608e6957dd4d521fd71685aac5533451d27f0abe03660",
```


(continues on next page)

(continued from previous page)

```
        "blockNumber": 3793534,
        "result": {
            "address": "0x61a7cc907c47c133d5ff5b685407201951fcbd08",
            "code":
↪"0x60806040526004361061004c576000357c0100000000000000000000000000000000000000000000000000000000000000
↪",
            "gasUsed": "0x4683f"
        },
        "subtraces": 2,
        "traceAddress": [],
        "transactionHash":
↪"0x6c7e8f8778d33d81b29c4bd7526ee50a4cea340d69eed6c89ada4e6fab731789",
        "transactionPosition": 1,
        "type": "create"
    },
    {
        'action': {
            'address': '0x4440adafbc6c4e45c299451c0eedc7c8b98c14ac',
            'balance': '0x0',
            'refundAddress': '0x0000000000000000000000000000000000000000'
        },
        'blockHash':
↪'0x8512d367492371edf44ebcbbbd935bc434946dddc2b126cb558df5906012186c',
        'blockNumber': 7829689,
        'result': None,
        'subtraces': 0,
        'traceAddress': [0, 0, 0, 0, 0, 0],
        'transactionHash':
↪'0x5f7af6aa390f9f8dd79ee692c37cbde76bb7869768b1bac438b6d176c94f637d',
        'transactionPosition': 35,
        'type': 'suicide'
    }
]
```

trace_transaction(*tx_hash: Hash32 | HexBytes | HexStr*) → List[FilterTrace]

Parameters
tx_hash –

Returns
List of internal txs for *tx_hash*

trace_transactions(*tx_hashes: Sequence[Hash32 | HexBytes | HexStr]*) → List[List[FilterTrace]]

Parameters
tx_hashes –

Returns
For every *tx_hash* a list of internal txs (in the same order as the *tx_hashes* were provided)

class gnosis.eth.ethereum_client.**TxSpeed**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

FAST = 4

FASTEST = 6

NORMAL = 3

SLOW = 2

SLOWEST = 0

VERY_FAST = 5

VERY_SLOW = 1

gnosis.eth.ethereum_client.**tx_with_exception_handling**(*func*)

Parity / OpenEthereum

- https://github.com/openethereum/openethereum/blob/main/rpc/src/v1/helpers/errors.rs

Geth

- https://github.com/ethereum/go-ethereum/blob/master/core/error.go
- https://github.com/ethereum/go-ethereum/blob/master/core/tx_pool.go

Comparison

- https://gist.github.com/kunal365roy/3c37ac9d1c3aaf31140f7c5faa083932

Parameters
func –

Returns

gnosis.eth.typing module

class gnosis.eth.typing.**BalanceDict**

Bases: TypedDict

balance: int

token_address: str | None

class gnosis.eth.typing.**LogReceiptDecoded**

Bases: LogReceipt

address: ChecksumAddress

args: Dict[str, Any]

blockHash: HexBytes

blockNumber: BlockNumber

data: HexBytes

logIndex: int

removed: bool

topics: Sequence[HexBytes]

transactionHash: HexBytes

transactionIndex: int

gnosis.eth.utils module

gnosis.eth.utils.**compare_byte_code**(*code_1: bytes*, *code_2: bytes*) → bool

Compare code, removing swarm metadata if necessary

Parameters

- **code_1** –
- **code_2** –

Returns
True if same code, False otherwise

gnosis.eth.utils.**decode_string_or_bytes32**(*data: bytes*) → str

gnosis.eth.utils.**fast_bytes_to_checksum_address**(*value: bytes*) → ChecksumAddress

Converts to checksum_address. Uses more optimal *pysha3* instead of *eth_utils* for keccak256 calculation. As input is already in bytes, some checks and conversions can be skipped, providing a speedup of ~50%

Parameters
value –

Returns

gnosis.eth.utils.**fast_is_checksum_address**(*value: AnyAddress | str | bytes*) → bool

Fast version to check if an address is a checksum_address

Parameters
value –

Returns
True if checksummed, *False* otherwise

gnosis.eth.utils.**fast_keccak**(*value: bytes*) → Hash32

Calculates ethereum keccak256 using fast library *pysha3* :param value: :return: Keccak256 used by ethereum as *bytes*

gnosis.eth.utils.**fast_keccak_hex**(*value: bytes*) → HexStr

Same as *fast_keccak*, but it's a little more optimal calling *hexdigest()* than calling *digest()* and then *hex()*

Parameters
value –

Returns
Keccak256 used by ethereum as a hex string (not 0x prefixed)

gnosis.eth.utils.**fast_to_checksum_address**(*value: AnyAddress | str | bytes*) → ChecksumAddress

Converts to checksum_address. Uses more optimal *pysha3* instead of *eth_utils* for keccak256 calculation

Parameters
value –

Returns

gnosis.eth.utils.**get_empty_tx_params**() → TxParams

Returns
Empty tx params, so calls like *build_transaction* don't call the RPC trying to get information

gnosis.eth.utils.**get_eth_address_with_invalid_checksum**() → str

gnosis.eth.utils.**mk_contract_address**(*address: str | bytes*, *nonce: int*) → ChecksumAddress
Generate expected contract address when using EVM CREATE

Parameters

- **address** –
- **nonce** –

Returns

gnosis.eth.utils.**mk_contract_address_2**(*from_: ChecksumAddress | bytes*, *salt: HexStr | bytes*, *init_code: HexStr | bytes*) → ChecksumAddress
Generate expected contract address when using EVM CREATE2.

Parameters

- **from** – The address which is creating this new address (need to be 20 bytes)
- **salt** – A salt (32 bytes)
- **init_code** – A init code of the contract being created

Returns
Address of the new contract

gnosis.eth.utils.**remove_swarm_metadata**(*code: bytes*) → bytes
Remove swarm metadata from Solidity bytecode

Parameters
code –

Returns
Code without metadata

Module contents

class gnosis.eth.**EthereumClient**(*ethereum_node_url: URI = 'http://localhost:8545'*, *provider_timeout: int = 15*, *slow_provider_timeout: int = 60*, *retry_count: int = 1*, *use_caching_middleware: bool = True*, *batch_request_max_size: int = 500*)

Bases: object

Manage ethereum operations. Uses web3 for the most part, but some other stuff is implemented from scratch. Note: If you want to use *pending* state with *Parity*, it must be run with *–pruning=archive* or *–force-sealing*

NULL_ADDRESS = '0x0000000000000000000000000000000000000000'

batch_call(*contract_functions: Iterable[ContractFunction]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *force_batch_call: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[bytes | Any | None]
Call multiple functions. Multicall contract by MakerDAO will be used by default if available

Parameters

- **contract_functions** –
- **from_address** – Only available when `Multicall` is not used
- **raise_exception** – If `True`, raise `BatchCallException` if one of the calls fails
- **force_batch_call** – If `True`, ignore multicall and always use batch calls to get the result (less optimal). If `False`, more optimal way will be tried.
- **block_identifier** –

Returns

List of elements decoded to their types, `None` if they cannot be decoded and bytes if a revert error is returned and `raise_exception=False`

Raises

BatchCallException

batch_call_manager: *BatchCallManager*

batch_call_same_function(*contract_function: ContractFunction*, *contract_addresses: Sequence[ChecksumAddress]*, *from_address: ChecksumAddress | None = None*, *raise_exception: bool = True*, *force_batch_call: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[bytes | Any | None]

Call the same function in multiple contracts. Way more optimal than using `batch_call` generating multiple `ContractFunction` objects.

Parameters

- **contract_function** –
- **contract_addresses** –
- **from_address** – Only available when `Multicall` is not used
- **raise_exception** – If `True`, raise `BatchCallException` if one of the calls fails
- **force_batch_call** – If `True`, ignore multicall and always use batch calls to get the result (less optimal). If `False`, more optimal way will be tried.
- **block_identifier** –

Returns

List of elements decoded to the same type, `None` if they cannot be decoded and bytes if a revert error is returned and `raise_exception=False`

Raises

BatchCallException

static build_tx_params(*from_address: ChecksumAddress | None = None*, *to_address: ChecksumAddress | None = None*, *value: int | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*, *chain_id: int | None = None*, *tx_params: TxParams | None = None*) → TxParams

Build tx params dictionary. If an existing TxParams dictionary is provided the fields will be replaced by the provided ones

Parameters

- **from_address** –

- **to_address** –
- **value** –
- **gas** –
- **gas_price** –
- **nonce** –
- **chain_id** –
- **tx_params** – An existing TxParams dictionary will be replaced by the providen values

Returns

check_tx_with_confirmations(*tx_hash: Hash32 | HexBytes | HexStr*, *confirmations: int*) → bool

Check tx hash and make sure it has the confirmations required

Parameters

- **tx_hash** – Hash of the tx
- **confirmations** – Minimum number of confirmations required

Returns

True if tx was mined with the number of confirmations required, False otherwise

property current_block_number

deploy_and_initialize_contract(*deployer_account: LocalAccount*, *constructor_data: bytes*, *initializer_data: bytes = b''*, *check_receipt: bool = True*) → *EthereumTxSent*

erc20: *Erc20Manager*

erc721: *Erc721Manager*

static estimate_data_gas(*data: bytes*)

Estimate gas costs only for "storage" of the `data` bytes provided

Parameters

data –

Returns

estimate_fee_eip1559(*tx_speed:* TxSpeed *= TxSpeed.NORMAL*) → Tuple[int, int]

Check https://github.com/ethereum/execution-apis/blob/main/src/eth/fee_market.json#L15

Returns

Tuple[BaseFeePerGas, MaxPriorityFeePerGas]

Raises

ValueError if not supported on the network

estimate_gas(*to: str*, *from_: str | None = None*, *value: int | None = None*, *data: bytes | HexStr | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = None*) → int

Estimate gas calling *eth_estimateGas*

Parameters

- **from** –

- **to** –
- **value** –
- **data** –
- **gas** –
- **gas_price** –
- **block_identifier** – Be careful, *Geth* does not support *pending* when estimating

Returns
Amount of gas needed for transaction

Raises
ValueError

ethereum_node_url: str

get_balance(*address: ChecksumAddress*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = None*)

get_block(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int*, *full_transactions: bool = False*) → BlockData | None

get_blocks(*block_identifiers: Iterable[Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int]*, *full_transactions: bool = False*) → List[BlockData | None]

get_chain_id() → int

Returns
ChainId returned by the RPC *eth_chainId* method. It should never change, so it's cached.

get_client_version() → str

Returns
RPC version information

get_network() → *EthereumNetwork*

Get network name based on the chainId. This method is not cached as the method for getting the *chainId* already is.

Returns
EthereumNetwork based on the chainId. If network is not on our list, *EthereumNetwork.UNKNOWN* is returned

get_nonce_for_account(*address: ChecksumAddress*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*)

Get nonce for account. *getTransactionCount* is the only method for what *pending* is currently working (Geth and Parity)

Parameters

- **address** –
- **block_identifier** –

Returns

get_transaction(*tx_hash: Hash32 | HexBytes | HexStr*) → TxData | None

get_transaction_receipt(*tx_hash: Hash32 | HexBytes | HexStr*, *timeout=None*) → TxReceipt | None

get_transaction_receipts(*tx_hashes: Sequence[bytes | HexStr]*) → List[TxReceipt | None]

get_transactions(*tx_hashes: Sequence[Hash32 | HexBytes | HexStr]*) → List[TxData | None]

is_contract(*contract_address: ChecksumAddress*) → bool

is_eip1559_supported() → bool

Returns
True if EIP1559 is supported by the node, *False* otherwise

property multicall: Multicall

static private_key_to_address(*private_key*)

raw_batch_request(*payload: Sequence[Dict[str, Any]]*, *batch_size: int | None = None*) → Iterable[Dict[str, Any] | None]

Perform a raw batch JSON RPC call

Parameters

- **payload** – Batch request payload. Make sure all provided *ids* inside the payload are different
- **batch_size** – If *payload* length is bigger than size, it will be split into smaller chunks before sending to the server

Returns

Raises
ValueError

send_eth_to(*private_key: str*, *to: str*, *gas_price: int*, *value: Wei*, *gas: int | None = None*, *nonce: int | None = None*, *retry: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'pending'*) → bytes

Send ether using configured account

Parameters

- **private_key** – to
- **to** – to
- **gas_price** – gas_price
- **value** – value(wei)
- **gas** – gas, defaults to 22000
- **retry** – Retry if a problem is found
- **nonce** – Nonce of sender account
- **block_identifier** – Block identifier for nonce calculation

Returns
tx_hash

send_raw_transaction(*raw_transaction: bytes | HexStr*) → HexBytes

send_transaction(*transaction_dict: TxParams*) → HexBytes

send_unsigned_transaction(*tx: TxParams*, *private_key: str | None = None*, *public_key: str | None = None*, *retry: bool = False*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'pending'*) → HexBytes

Send a tx using an unlocked public key in the node or a private key. Both *public_key* and *private_key* cannot be *None*

Parameters

- **tx** –
- **private_key** –
- **public_key** –
- **retry** – Retry if a problem with nonce is found
- **block_identifier** – For nonce calculation, recommended is *pending*

Returns
tx hash

set_eip1559_fees(*tx: TxParams*, *tx_speed:* TxSpeed *= TxSpeed.NORMAL*) → TxParams

Returns
TxParams in EIP1559 format

Raises
ValueError if EIP1559 not supported

slow_w3: Web3

tracing: *TracingManager*

w3: Web3

class gnosis.eth.**EthereumClientProvider**

Bases: object

class gnosis.eth.**EthereumNetwork**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Use https://chainlist.org/ as a reference

ACALA_MANDALA_TESTNET_TC9 = 595

ACALA_NETWORK = 787

ACALA_NETWORK_TESTNET = 597

ACRIA_INTELLICHAIN = 47

ADIL_CHAIN_V2_MAINNET = 7576

ADIL_DEVNET = 123456

ADIL_TESTNET = 7575

AERIE_NETWORK = 84886

AEROCHAIN_TESTNET = 788

AGUNG_NETWORK = 9990

AIA_MAINNET = 1319

AIA_TESTNET = 1320

AIOZ_NETWORK = 168

AIOZ_NETWORK_TESTNET = 4102

AIRDAO_MAINNET = 16718

AIRDAO_TESTNET = 22040

AKROMA = 200625

ALAYA_DEV_TESTNET = 201030

ALAYA_MAINNET = 201018

ALGOL = 2089

ALL_MAINNET = 651940

ALPHABET_MAINNET = 111222333444

ALPH_NETWORK = 8738

ALTAIR = 2088

ALTCOINCHAIN = 2330

ALTERIUM_L2_TESTNET = 420692

ALTLAYER_TESTNET = 9997

ALTLAYER_ZERO_GAS_NETWORK = 4000003

ALVEYCHAIN_MAINNET = 3797

ALYX_CHAIN_TESTNET = 135

ALYX_MAINNET = 1314

AMANA = 8134

AMANA_MIXNET = 81342

AMANA_PRIVNET = 81343

AMANA_TESTNET = 81341

AMBROS_CHAIN_MAINNET = 880

AME_CHAIN_MAINNET = 180

AMPLIFY_SUBNET = 78430

AMSTAR_MAINNET = 1388

AMSTAR_TESTNET = 1138

```
ANCIENT8_TESTNET = 28122024

ANCIENT8_TESTNET_DEPRECATED = 2863311531

ANDUSCHAIN_MAINNET = 14288640

ANTOFY_MAINNET = 2202

ANTOFY_TESTNET = 23006

ANYTYPE_EVM_CHAIN = 1701

AQUACHAIN = 61717561

ARBITRUM_GOERLI = 421613

ARBITRUM_NOVA = 42170

ARBITRUM_ONE = 42161

ARBITRUM_ON_XDAI = 200

ARBITRUM_RINKEBY = 421611

ARBITRUM_SEPOLIA = 421614

ARCOLOGY_TESTNET = 118

ARCTURUS_CHAIN_TESTNET = 5616

ARC_MAINNET = 1243

ARC_TESTNET = 1244

ARDENIUM_ATHENA = 7895

AREON_NETWORK_MAINNET = 463

AREON_NETWORK_TESTNET = 462

AREVIA = 2309

ARMONIA_EVA_CHAIN_MAINNET = 160

ARMONIA_EVA_CHAIN_TESTNET = 161

ARTHERA_MAINNET = 10242

ARTHERA_TESTNET = 10243

ARTIS_SIGMA1 = 246529

ARTIS_TESTNET_TAU1 = 246785

ARZIO_CHAIN = 456

ASTAR = 592

ASTAR_ZKEVM = 12611

ASTRA = 11110
```

ASTRA_TESTNET = 11115

ASTRIA_EVM_DUSKNET = 912559

ATELIER = 1971

ATHEIOS = 1620

ATHEREUM = 43110

ATOSHI_TESTNET = 167

AURORA_BETANET = 1313161556

AURORA_MAINNET = 1313161554

AURORA_TESTNET = 1313161555

AUTOBAHN_NETWORK = 45000

AUTONITY_BAKERLOO_BARADA_TESTNET = 65010001

AUTONITY_BAKERLOO_THAMES_TESTNET = 65010000

AUTONITY_PICCADILLY_BARADA_TESTNET = 65100001

AUTONITY_PICCADILLY_THAMES_TESTNET = 65100000

AUXILIUM_NETWORK_MAINNET = 28945486

AVALANCHE_C_CHAIN = 43114

AVALANCHE_FUJI_TESTNET = 43113

AVES_MAINNET = 33333

AVES_TESTNET = 333331

AVOCADO = 634

AXELCHAIN_DEV_NET = 61800

BAHAMUT = 5165

BANDAI_NAMCO_RESEARCH_VERSE_MAINNET = 876

BASE = 8453

BASE_GOERLI_TESTNET = 84531

BASE_SEPOLIA_TESTNET = 84532

BEAGLE_MESSAGING_CHAIN = 1515

BEAM = 4337

BEAM_TESTNET = 13337

BEANECO_SMARTCHAIN = 535037

BEAR_NETWORK_CHAIN_MAINNET = 641230

```
BEAR_NETWORK_CHAIN_TESTNET = 751230

BEONE_CHAIN_MAINNET = 818

BERACHAIN_ARTIO = 80085

BERESHEET_BEREEVM_TESTNET = 2022

BERYLBIT_MAINNET = 9012

BEVERLY_HILLS = 90210

BEVM_CANARY = 1501

BIFROST_MAINNET = 3068

BIFROST_TESTNET = 49088

BIGSHORTBETS = 2137

BITCHAIN_MAINNET = 198

BITCICHAIN_MAINNET = 1907

BITCICHAIN_TESTNET = 1908

BITCOIN_CHAIN = 8086

BITCOIN_EVM = 2203

BITFINITY_NETWORK_TESTNET = 355113

BITGERT_MAINNET = 32520

BITICA_CHAIN_MAINNET = 188710

BITINDI_MAINNET = 4099

BITINDI_TESTNET = 4096

BITKUB_CHAIN = 96

BITKUB_CHAIN_TESTNET = 25925

BITNET = 210

BITROCK_MAINNET = 7171

BITROCK_TESTNET = 7771

BITTEX_MAINNET = 3690

BITTORRENT_CHAIN_MAINNET = 199

BITTORRENT_CHAIN_TESTNET = 1028

BITYUAN_MAINNET = 2999

BLACKFORT_EXCHANGE_NETWORK = 4999

BLACKFORT_EXCHANGE_NETWORK_TESTNET = 4777
```

```
BLAST_MAINNET = 238

BLAST_TESTNET = 23888

BLG_TESTNET = 12321

BLITZ_SUBNET = 1343

BLOCKCHAIN_GENESIS_MAINNET = 10101

BLOCKCHAIN_STATION_MAINNET = 707

BLOCKCHAIN_STATION_TESTNET = 708

BLOCKTON_BLOCKCHAIN = 8272

BLOCX_MAINNET = 879151

BLOXBERG = 8995

BLUCRATES = 727

BLXQ_MAINNET = 1108

BLXQ_TESTNET = 1107

BMC_MAINNET = 188

BMC_TESTNET = 189

BNB_SMART_CHAIN_MAINNET = 56

BNB_SMART_CHAIN_TESTNET = 97

BOBABASE_TESTNET = 1297

BOBABEAM = 1294

BOBAFUJI_TESTNET = 4328

BOBAOPERA = 301

BOBAOPERA_TESTNET = 4051

BOBA_AVAX = 43288

BOBA_BNB_MAINNET = 56288

BOBA_BNB_MAINNET_OLD = 97288

BOBA_BNB_TESTNET = 9728

BOBA_NETWORK = 288

BOBA_NETWORK_GOERLI_TESTNET = 2888

BOBA_NETWORK_RINKEBY_TESTNET = 28

BOMB_CHAIN = 2300

BOMB_CHAIN_TESTNET = 2399
```

BON_NETWORK = 1898

BOSAGORA_MAINNET = 2151

BOTANIX_MAINNET = 3637

BOTANIX_TESTNET = 3636

BRC_CHAIN_MAINNET = 12123

BROCHAIN_MAINNET = 108801

BRONOS_MAINNET = 1039

BRONOS_TESTNET = 1038

BSL_MAINNET = 21912

BTACHAIN = 1657

BTC20_SMART_CHAIN = 963

BTCIX_NETWORK = 19845

BULLETIN_SUBNET = 78431

CALLISTO_MAINNET = 820

CALLISTO_TESTNET = 20729

CALLISTO_TESTNET_DEPRECATED = 821

CAMDL_MAINNET = 95

CAMELARK_MAINNET = 20001

CAMINO_C_CHAIN = 500

CANDLE = 534

CANTO = 7700

CANTO_TESNET = 7701

CANTO_TESTNET = 740

CANXIUM_MAINNET = 3003

CARBON_EVM = 9790

CARBON_EVM_TESTNET = 9792

CASCADIA_TESTNET = 6102

CATALYST = 2032

CATECOIN_CHAIN_MAINNET = 1618

CELO_ALFAJORES_TESTNET = 44787

CELO_BAKLAVA_TESTNET = 62320

```
CELO_MAINNET = 42220

CENNZNET_AZALEA = 21337

CENNZNET_NIKAU = 3001

CENNZNET_RATA = 3000

CENTRIFUGE = 2031

CERIUM_TESTNET = 30103

CHAIN_VERSE_MAINNET = 5555

CHAOS_SKALE_TESTNET = 1351057110

CHEAPETH = 777

CHILIZ_SCOVILLE_TESTNET = 88880

CIC_CHAIN_MAINNET = 1353

CIC_CHAIN_TESTNET = 1252

CLOUDTX_MAINNET = 31223

CLOUDTX_TESTNET = 31224

CLOUDVERSE_SUBNET = 33210

CLOUDWALK_MAINNET = 2009

CLOUDWALK_TESTNET = 2008

CLOVER_TESTNET = 1023

CLV_PARACHAIN = 1024

CMP_MAINNET = 256256

CMP_TESTNET = 512512

CODEFIN_MAINNET = 9223

COINBIT_MAINNET = 112

COINEX_SMART_CHAIN_MAINNET = 52

COINEX_SMART_CHAIN_TESTNET = 53

COLUMBUS_TEST_NETWORK = 501

COMBO_MAINNET = 9980

COMBO_TESTNET = 91715

COMPVERSE_MAINNET = 6779

CONDOR_TEST_NETWORK = 188881

CONDRIEU = 69420
```

```
CONDUIT_SUBNET = 78432

CONET_HOLESKY = 224433

CONET_SEBOLIA_TESTNET = 224422

CONFLUX_ESPACE = 1030

CONFLUX_ESPACE_TESTNET = 71

CONNECTORMANAGER = 38400

CONNECTORMANAGER_ROBIN = 38401

CONSTA_TESTNET = 371

CORE_BLOCKCHAIN_MAINNET = 1116

CORE_BLOCKCHAIN_TESTNET = 1115

COSMIC_CHAIN = 67588

CRAB_NETWORK = 44

CREDIT_SMART_CHAIN = 13308

CREDIT_SMART_CHAIN_MAINNET = 4400

CRONOS_MAINNET = 25

CRONOS_TESTNET = 338

CROSSBELL = 3737

CRYPTOCOINPAY = 10823

CRYPTO_EMERGENCY = 193

CRYSTALEUM = 103090

CTEX_SCAN_BLOCKCHAIN = 1455

CUBE_CHAIN_MAINNET = 1818

CUBE_CHAIN_TESTNET = 1819

CURVE_MAINNET = 827431

CYBERDECKNET = 1146703430

CYBERTRUST = 85449

CYBRIA_MAINNET = 6661

CYBRIA_TESTNET = 6666

DARWINIA_NETWORK = 46

DARWINIA_PANGOLIN_TESTNET = 43

DARWINIA_PANGORO_TESTNET = 45
```

```
DATAHOPPER = 2021121117

DAX_CHAIN = 142

DBCHAIN_TESTNET = 67

DEAMCHAIN_MAINNET = 136

DEAMCHAIN_TESTNET = 236

DEBANK_MAINNET = 116

DEBANK_TESTNET = 2021398

DEBANK_TESTNET_DEPRECATED = 115

DEBOUNCE_SUBNET_TESTNET = 3306

DECENTRABONE_LAYER1_TESTNET = 910

DECENTRALIZED_WEB_MAINNET = 124

DECIMAL_SMART_CHAIN_MAINNET = 75

DECIMAL_SMART_CHAIN_TESTNET = 202020

DEELANCE_MAINNET = 45510

DEFICHAIN_EVM_NETWORK_MAINNET = 1130

DEFICHAIN_EVM_NETWORK_TESTNET = 1131

DEFIMETACHAIN_CHANGI_TESTNET = 1133

DEFI_ORACLE_META_MAINNET = 138

DEFI_ORACLE_META_TESTNET = 2138

DEHVO = 113

DEPRECATED_CHI = 100100

DEXALOT_SUBNET = 432204

DEXALOT_SUBNET_TESTNET = 432201

DEXILLA_TESTNET = 1954

DEXIT_NETWORK = 877

DFK_CHAIN = 53935

DFK_CHAIN_TEST = 335

DIGIT_SOUL_SMART_CHAIN = 6363

DIGIT_SOUL_SMART_CHAIN_2 = 363636

DIODE_PRENET = 15

DIODE_TESTNET_STAGING = 13
```

```
DM2_VERSE_MAINNET = 68770

DODAO = 855456

DOGCOIN_MAINNET = 1117

DOGCOIN_TESTNET = 9339

DOGECHAIN_MAINNET = 2000

DOGECHAIN_TESTNET = 568

DOGELAYER_MAINNET = 9888

DOGETHER_MAINNET = 1248

DOKEN_SUPER_CHAIN_MAINNET = 61916

DOS_CHAIN = 7979

DOS_FUJI_SUBNET = 1311

DOS_TESNET = 3939

DOUBLE_A_CHAIN_MAINNET = 512

DOUBLE_A_CHAIN_TESTNET = 513

DPU_CHAIN = 2611555

DRACONES_FINANCIAL_SERVICES = 8387

DRAC_NETWORK = 3912

DRAGONFLY_MAINNET_HEXAPOD = 78281

DUBXCOIN_NETWORK = 3269

DUBXCOIN_TESTNET = 3270

DXCHAIN_MAINNET = 36

DXCHAIN_TESTNET = 72

DYNO_MAINNET = 3966

DYNO_TESTNET = 3967

D_CHAIN_MAINNET = 1951

EBRO_NETWORK = 2306

ECOBALL_MAINNET = 2100

ECOBALL_TESTNET_ESPUMA = 2101

ECREDITS_MAINNET = 63000

ECREDITS_TESTNET = 63001

ECROX_CHAIN_MAINNET = 988207
```

```
EDEXA_TESTNET = 1995

EDGEWARE_EDGEEVM_MAINNET = 2021

EGONCOIN_MAINNET = 271

EGONCOIN_TESTNET = 271271

EKTA = 1994

ELASTOS_SMART_CHAIN = 20

ELASTOS_SMART_CHAIN_TESTNET = 21

ELA_DID_SIDECHAIN_MAINNET = 22

ELA_DID_SIDECHAIN_TESTNET = 23

ELEANOR = 1967

ELECTRONEUM_MAINNET = 52014

ELECTRONEUM_TESTNET = 5201420

ELIBERTY_MAINNET = 990

ELIBERTY_TESTNET = 99099

ELLAISM = 64

ELLA_THE_HEART = 7027

ELUVIO_CONTENT_FABRIC = 955305

ELYSIUM_MAINNET = 1339

ELYSIUM_TESTNET = 1338

EMPIRE_NETWORK = 3693

ENDURANCE_SMART_CHAIN_MAINNET = 648

ENERGI_MAINNET = 39797

ENERGI_TESTNET = 49797

ENERGY_WEB_CHAIN = 246

ENERGY_WEB_VOLTA_TESTNET = 73799

ENGRAM_TESTNET = 131

ENNOTHEM_MAINNET_PROTEROZOIC = 48

ENNOTHEM_TESTNET_PIONEER = 49

ENTERCHAIN_MAINNET = 1214

ENULS_MAINNET = 119

ENULS_TESTNET = 120
```

```
EOS_EVM_LEGACY = 59

EOS_EVM_NETWORK = 17777

EOS_EVM_NETWORK_TESTNET = 15557

ERASWAP_MAINNET = 5197

ETHEREUM_CLASSIC = 61

ETHEREUM_FAIR = 513100

ETHERGEM = 1987

ETHERINC = 101

ETHERLINK_TESTNET = 128123

ETHERLITE_CHAIN = 111

ETHERSOCIAL_NETWORK = 31102

ETHO_PROTOCOL = 1313114

ETICA_MAINNET = 61803

ETND_CHAIN_MAINNETS = 131419

EURUS_MAINNET = 1008

EURUS_TESTNET = 1984

EVANESCO_MAINNET = 2213

EVANESCO_TESTNET = 1201

EVMOS = 9001

EVMOS_TESTNET = 9000

EVOKE_MAINNET = 9395

EVOKE_TESTNET = 31414

EVRICE_NETWORK = 1010

EXCELON_MAINNET = 22052002

EXCOINCIAL_CHAIN_MAINNET = 27082022

EXCOINCIAL_CHAIN_VOLTA_TESTNET = 27082017

EXOSAMA_NETWORK = 2109

EXPANSE_NETWORK = 2

EXZO_NETWORK_MAINNET = 1229

EZCHAIN_C_CHAIN_MAINNET = 2612

EZCHAIN_C_CHAIN_TESTNET = 2613
```

E_DOLLAR = 8087

FACTORY_127_MAINNET = 127

FANTASIA_CHAIN_MAINNET = 868

FANTOM_OPERA = 250

FANTOM_TESTNET = 4002

FASTEX_CHAIN_BAHAMUT_OASIS_TESTNET = 4090

FASTEX_CHAIN_TESTNET = 424242

FERRUM_TESTNET = 26026

FIBONACCI_MAINNET = 12306

FILECOIN_BUTTERFLY_TESTNET = 3141592

FILECOIN_CALIBRATION_TESTNET = 314159

FILECOIN_HYPERSPACE_TESTNET = 3141

FILECOIN_LOCAL_TESTNET = 31415926

FILECOIN_MAINNET = 314

FILECOIN_WALLABY_TESTNET = 31415

FINDORA_FORGE = 2154

FINDORA_MAINNET = 2152

FINDORA_TESTNET = 2153

FIRECHAIN_MAINNET = 529

FIRECHAIN_MAINNET_OLD = 5290

FIRECHAIN_ZKEVM = 814

FIRECHAIN_ZKEVM_GHOSTRIDER = 3885

FIRENZE_TEST_NETWORK = 78110

FLACHAIN_MAINNET = 29032022

FLANA = 8135

FLANA_MIXNET = 81352

FLANA_PRIVNET = 81353

FLANA_TESTNET = 81351

FLARE_MAINNET = 14

FLARE_TESTNET_COSTON = 16

FLARE_TESTNET_COSTON2 = 114

```
FNCY = 73
FNCY_TESTNET = 923018
FOUNDRY_CHAIN_TESTNET = 77238
FOX_TESTNET_NETWORK = 6565
FRAME_TESTNET = 68840142
FRAXTAL_MAINNET = 252
FRAXTAL_TESTNET = 2522
FREIGHT_TRUST_NETWORK = 211
FRENCHAIN = 44444
FRONTIER_OF_DREAMS_TESTNET = 18000
FUSE_MAINNET = 122
FUSE_SPARKNET = 123
FUSION_MAINNET = 32659
FUSION_TESTNET = 46688
F_XCORE_MAINNET_NETWORK = 530
G8CHAIN_MAINNET = 17171
G8CHAIN_TESTNET = 18181
GANACHE = 1337
GARIZON_STAGE0 = 90
GARIZON_STAGE1 = 91
GARIZON_STAGE2 = 92
GARIZON_STAGE3 = 93
GARIZON_TESTNET_STAGE0 = 900
GARIZON_TESTNET_STAGE1 = 901
GARIZON_TESTNET_STAGE2 = 902
GARIZON_TESTNET_STAGE3 = 903
GATECHAIN_MAINNET = 86
GATECHAIN_TESTNET = 85
GATHER_DEVNET_NETWORK = 486217935
GATHER_MAINNET_NETWORK = 192837465
GATHER_TESTNET_NETWORK = 356256156
```

GAUSS_MAINNET = 1777

GEAR_ZERO_NETWORK_MAINNET = 516

GEAR_ZERO_NETWORK_TESTNET = 266256

GENECHAIN = 80

GENESIS_COIN = 9100

GENESIS_L1 = 29

GENESIS_L1_TESTNET = 26

GENESYS_CODE_MAINNET = 59971

GENESYS_MAINNET = 16507

GESOTEN_VERSE_TESTNET = 42801

GESO_VERSE = 428

GIANT_MAMMOTH_MAINNET = 8989

GIL_TESTNET = 1452

GITSHOCK_CARTENZ_TESTNET = 1881

GLOBEL_CHAIN = 4893

GNOSIS = 100

GNOSIS_CHIADO_TESTNET = 10200

GOCHAIN = 60

GOCHAIN_TESTNET = 31337

GODWOKEN_MAINNET = 71402

GODWOKEN_TESTNET_V1 = 71401

GOERLI = 5

GOLDXCHAIN_MAINNET = 42355

GOLDXCHAIN_TESTNET = 22324

GOLD_SMART_CHAIN_MAINNET = 6789

GOLD_SMART_CHAIN_TESTNET = 79879

GON_CHAIN = 10024

GOODDATA_MAINNET = 33

GOODDATA_TESTNET = 32

GRAPHLINQ_BLOCKCHAIN_MAINNET = 614

GROK_CHAIN_MAINNET = 72992

```
GTON_MAINNET = 1000

GTON_TESTNET = 50021

GUAPCOINX = 71111

HAIC = 803

HALO_MAINNET = 1280

HAMMER_CHAIN_MAINNET = 25888

HAPCHAIN = 8794598

HAPCHAIN_TESTNET = 373737

HAQQ_CHAIN_TESTNET = 54211

HAQQ_NETWORK = 11235

HARADEV_TESTNET = 197710212031

HARMONY_DEVNET_SHARD_0 = 1666900000

HARMONY_DEVNET_SHARD_1 = 1666900001

HARMONY_MAINNET_SHARD_0 = 1666600000

HARMONY_MAINNET_SHARD_1 = 1666600001

HARMONY_MAINNET_SHARD_2 = 1666600002

HARMONY_MAINNET_SHARD_3 = 1666600003

HARMONY_TESTNET_SHARD_0 = 1666700000

HARMONY_TESTNET_SHARD_1 = 1666700001

HASHBIT_MAINNET = 11119

HASHKEY_CHAIN_TESTNET = 230315

HAVEN1_TESTNET = 810

HAYMO_TESTNET = 234666

HAZLOR_TESTNET = 7878

HEDERA_LOCALNET = 298

HEDERA_MAINNET = 295

HEDERA_PREVIEWNET = 297

HEDERA_TESTNET = 296

HELA_OFFICIAL_RUNTIME_MAINNET = 8668

HELA_OFFICIAL_RUNTIME_TESTNET = 666888

HELP_THE_HOMELESS = 7118
```

HERTZ_NETWORK_MAINNET = 26600

HIGHBURY = 710

HIGH_PERFORMANCE_BLOCKCHAIN = 269

HIKA_NETWORK_TESTNET = 5729

HOKUM = 8080808

HOKUM_TESTNET = 20482050

HOLESKY = 17000

HOME_VERSE_MAINNET = 19011

HOO_SMART_CHAIN = 70

HOO_SMART_CHAIN_TESTNET = 170

HORIZEN_EON_MAINNET = 7332

HORIZEN_GOBI_TESTNET = 1663

HTMLCOIN_MAINNET = 4444

HUMANODE_MAINNET = 5234

HUMANODE_TESTNET_5_ISRAFEL = 14853

HUMANS_AI_MAINNET = 1089

HUMANS_AI_TESTNET = 4139

HUMAN_PROTOCOL = 1273227453

HUOBI_ECO_CHAIN_MAINNET = 128

HUOBI_ECO_CHAIN_TESTNET = 256

HYBRID_CHAIN_NETWORK_MAINNET = 2468

HYBRID_CHAIN_NETWORK_TESTNET = 2458

HYPERONCHAIN_TESTNET = 400

HYPRA_MAINNET = 622277

ICHAIN_NETWORK = 3639

ICPLAZA_MAINNET = 142857

IDCHAIN_MAINNET = 74

IEXEC_SIDECHAIN = 134

IMMU3_EVM = 3100

IMMUTABLE_ZKEVM = 13371

IMMUTABLE_ZKEVM_DEVNET = 15003

```
    IMMUTABLE_ZKEVM_TESTNET = 13473
    IMPERIUM_MAINNET = 9819
    IMPERIUM_TESTNET = 9818
    IMVERSED_MAINNET = 5555555
    IMVERSED_TESTNET = 5555558
    IOLITE = 18289463
    IORA_CHAIN = 1197
    IOTEX_NETWORK_MAINNET = 4689
    IOTEX_NETWORK_TESTNET = 4690
    IPOS_NETWORK = 1122334455
    IRISHUB = 6688
    IRISHUB_TESTNET = 16688
    IVAR_CHAIN_MAINNET = 88888
    IVAR_CHAIN_TESTNET = 16888
    J2O_TARO = 35011
    JANUS_TESTNET = 66988
    JAPAN_OPEN_CHAIN_MAINNET = 81
    JAPAN_OPEN_CHAIN_TESTNET = 10081
    JELLIE = 202624
    JFIN_CHAIN = 3501
    JIBCHAIN_L1 = 8899
    JOSEON_MAINNET = 1392
    JOULEVERSE_MAINNET = 3666
    JOYS_DIGITAL_MAINNET = 35855456
    JOYS_DIGITAL_TESTNET = 99415706
    JUNCACHAIN = 668
    JUNCACHAIN_TESTNET = 669
    KAIBA_LIGHTNING_CHAIN_TESTNET = 104
    KALAR_CHAIN = 1379
    KALICHAIN = 654
    KALICHAIN_TESTNET = 653
```

KALYCHAIN_MAINNET = 3888

KALYCHAIN_TESTNET = 3889

KANAZAWA = 222000222

KARDIACHAIN_MAINNET = 24

KARURA_NETWORK = 686

KARURA_NETWORK_TESTNET = 596

KAVA = 2222

KAVA_TESTNET = 2221

KCC_MAINNET = 321

KCC_TESTNET = 322

KEKCHAIN = 420420

KEKCHAIN_KEKTEST = 420666

KERLEANO = 1804

KILN = 1337802

KINTO_TESTNET = 42888

KINTSUGI = 1337702

KIWI_SUBNET = 2037

KLAYTN_MAINNET_CYPRESS = 8217

KLAYTN_TESTNET_BAOBAB = 1001

KLYNTAR = 7331

KORTHOTEST = 8285

KORTHO_MAINNET = 2559

KOTTI_TESTNET = 6

KREST_NETWORK = 2241

KROMA = 255

KROMA_SEPOLIA = 2358

KYOTO_TESTNET = 1998

K_LAOS = 2718

LACHAIN = 274

LACHAIN_MAINNET = 225

LACHAIN_TESTNET = 226

```
    LAMBDA_TESTNET = 92001
    LAOS_ARRAKIS = 667
    LATAM_BLOCKCHAIN_RESIL_TESTNET = 172
    LATESTNET = 418
    LATEST_CHAIN_TESTNET = 6660
    LIGHTLINK_PEGASUS_TESTNET = 1891
    LIGHTLINK_PHOENIX_MAINNET = 1890
    LIGHTSTREAMS_MAINNET = 163
    LIGHTSTREAMS_TESTNET = 162
    LINEA = 59144
    LINEA_TESTNET = 59140
    LINQTO_DEVNET = 84
    LIQUICHAIN = 1662
    LISINSKI = 385
    LISK_SEPOLIA_TESTNET = 4202
    LIVEPLEX_ORACLEEVM = 50001
    LIVING_ASSETS_MAINNET = 1440
    LOOPNETWORK_MAINNET = 15551
    LUCID_BLOCKCHAIN = 800
    LUCKY_NETWORK = 998
    LUDAN_MAINNET = 1688
    LUKSO_MAINNET = 42
    LUKSO_TESTNET = 4201
    LUMOZ_TESTNET_ALPHA = 51178
    LYCAN_CHAIN = 721
    LYRA_CHAIN = 957
    MAALCHAIN_TESTNET = 7860
    MAAL_CHAIN = 786
    MAINNET = 1
    MAINNETZ_MAINNET = 2016
    MAINNETZ_TESTNET = 9768
```

```
MAISTESTSUBNET = 43214913

MAMMOTH_MAINNET = 8898

MANTA_PACIFIC_MAINNET = 169

MANTA_PACIFIC_TESTNET = 3441005

MANTIS_TESTNET_HEXAPOD = 96970

MANTLE = 5000

MANTLE_SEPOLIA_TESTNET = 5003

MANTLE_TESTNET = 5001

MAP_MAINNET = 22776

MAP_MAKALU = 212

MARKR_GO = 431140

MARO_BLOCKCHAIN_MAINNET = 8848

MAS_MAINNET = 220315

MATHCHAIN = 1139

MATHCHAIN_TESTNET = 1140

MAXXCHAIN_MAINNET = 10201

MCH_VERSE_MAINNET = 29548

MDGL_TESTNET = 8029

MELD = 333000333

MEMO_SMART_CHAIN_MAINNET = 985

MERKLE_SCAN = 1909

MESHNYAN_TESTNET = 600

METACHAIN_MAINNET = 571

METACHAIN_ONE_MAINNET = 112358

METADIUM_MAINNET = 11

METADIUM_TESTNET = 12

METADOT_MAINNET = 16000

METADOT_TESTNET = 16001

METAL_C_CHAIN = 381931

METAL_TAHOE_C_CHAIN = 381932

METAPLAYERONE_DUBAI_TESTNET = 2124
```

```
METAPLAYERONE_MAINNET = 2122

METER_MAINNET = 82

METER_TESTNET = 83

METIS_ANDROMEDA_MAINNET = 1088

METIS_GOERLI_TESTNET = 599

METIS_STARDUST_TESTNET = 588

MEVERSE_CHAIN_MAINNET = 7518

MEVERSE_CHAIN_TESTNET = 4759

MIEXS_SMARTCHAIN = 761412

MILKOMEDA_A1_MAINNET = 2002

MILKOMEDA_A1_TESTNET = 200202

MILKOMEDA_C1_MAINNET = 2001

MILKOMEDA_C1_TESTNET = 200101

MILVINE = 9322253

MIND_SMART_CHAIN_MAINNET = 9996

MIND_SMART_CHAIN_TESTNET = 9977

MINTARA_MAINNET = 1080

MINTARA_TESTNET = 1079

MINTME_COM_COIN = 24734

MIX = 76

MIXIN_VIRTUAL_MACHINE = 73927

MIZANA = 8136

MIZANA_MIXNET = 81362

MIZANA_PRIVNET = 81363

MIZANA_TESTNET = 81361

MOAC_MAINNET = 1099

MOAC_TESTNET = 201

MODE = 34443

MODE_TESTNET = 919

MODULARIUM = 776877

MOLEREUM_NETWORK = 6022140761023
```

```
MOONBASE_ALPHA = 1287

MOONBEAM = 1284

MOONRIVER = 1285

MOONROCK = 1288

MOONROCK_OLD = 1286

MOONSAMA_NETWORK = 2199

MORDEN_TESTNET = 62

MORDOR_TESTNET = 63

MORPH_TESTNET = 2710

MOVO_SMART_CHAIN_MAINNET = 2049

MULTIVAC_MAINNET = 62621

MUMBAI = 80001

MUNODE_TESTNET = 956

MUSICOIN = 7762959

MUSTER_MAINNET = 4078

MXC_WANNSEE_ZKEVM_TESTNET = 5167003

MXC_ZKEVM_MAINNET = 18686

MYOWN_TESTNET = 9999

MYTHICAL_CHAIN = 201804

NAHMII_3_MAINNET = 4061

NAHMII_3_TESTNET = 4062

NAHMII_MAINNET = 5551

NAHMII_TESTNET = 5553

NATIV3_MAINNET = 399

NATIV3_TESTNET = 333333

NAUTILUS_MAINNET = 22222

NAUTILUS_PROTEUS_TESTNET = 88002

NAUTILUS_TRITION_CHAIN = 91002

NEBULA_TESTNET = 107

NEONLINK_MAINNET = 259

NEONLINK_TESTNET = 9559
```

```
NEON_EVM_DEVNET = 245022926

NEON_EVM_MAINNET = 245022934

NEON_EVM_TESTNET = 245022940

NEPAL_BLOCKCHAIN_NETWORK = 977

NEUROCHAIN_MAINNET = 313

NEUROCHAIN_TESTNET = 303

NEUTRINOS_TESTNET = 197

NEWTON = 1012

NEWTON_TESTNET = 1007

NEXI_MAINNET = 4242

NEXI_V2_MAINNET = 4243

NORDEK_MAINNET = 81041

NOVA_NETWORK = 87

NTITY_MAINNET = 197710212030

NUMBERS_MAINNET = 10507

NUMBERS_TESTNET = 10508

NUME = 7100

OASISCHAIN_MAINNET = 26863

OASIS_EMERALD = 42262

OASIS_EMERALD_TESTNET = 42261

OASIS_SAPPHIRE = 23294

OASIS_SAPPHIRE_TESTNET = 23295

OASYS_MAINNET = 248

OCTASPACE = 800001

OEBLOCK_TESTNET = 156

OHO_MAINNET = 39815

OKEXCHAIN_TESTNET = 65

OKXCHAIN_MAINNET = 66

OLYMPIC = 0

OMAX_MAINNET = 311

OMCHAIN_MAINNET = 21816
```

OMNI_TESTNET = 165

OM_PLATFORM_MAINNET = 1246

ONELEDGER_MAINNET = 311752642

ONELEDGER_TESTNET_FRANKENSTEIN = 4216137055

ONTOLOGY_MAINNET = 58

ONTOLOGY_TESTNET = 5851

ONUS_CHAIN_MAINNET = 1975

ONUS_CHAIN_TESTNET = 1945

OONE_CHAIN_DEVNET = 333777

OONE_CHAIN_TESTNET = 333666

OORT_ASCRAEUS = 972

OORT_HUYGENS = 971

OORT_MAINNET = 970

OORT_MAINNETDEV = 9700

OPAL_TESTNET_BY_UNIQUE = 8882

OPBNB_MAINNET = 204

OPBNB_TESTNET = 5611

OPENCHAIN_MAINNET = 474142

OPENCHAIN_TESTNET = 776

OPENPIECE_MAINNET = 54

OPENPIECE_TESTNET = 141

OPENVESSEL = 7355310

OPSIDE_TESTNET = 23118

OPTIMISM = 10

OPTIMISM_BEDROCK_GOERLI_ALPHA_TESTNET = 28528

OPTIMISM_GOERLI_TESTNET = 420

OPTIMISM_KOVAN = 69

OPULENT_X_BETA = 41500

OP_SEPOLIA_TESTNET = 11155420

ORDERLY_MAINNET = 291

ORDERLY_SEPOLIA_TESTNET = 4460

```
ORIGINTRAIL_PARACHAIN = 2043

ORIGIN_TESTNET = 1170

ORLANDO_CHAIN = 3031

OYCHAIN_MAINNET = 126

OYCHAIN_TESTNET = 125

OZONE_CHAIN_MAINNET = 4000

OZONE_CHAIN_TESTNET = 401

P12_CHAIN = 20736

PALETTE_CHAIN_MAINNET = 1718

PALETTE_CHAIN_TESTNET = 17180

PALM = 11297108109

PALM_TESTNET = 11297108099

PANDOPROJECT_MAINNET = 3601

PANDOPROJECT_TESTNET = 3602

PARIBU_NET_MAINNET = 3400

PARIBU_NET_TESTNET = 3500

PARTYCHAIN = 1773

PATEX = 789

PATEX_SEPOLIA_TESTNET = 471100

PAWCHAIN_TESTNET = 542

PAXB_MAINNET = 6701

PDC_MAINNET = 666301171999

PEERPAY = 6502

PEGGLECOIN = 42069

PEGO_NETWORK = 20201022

PEPCHAIN_CHURCHILL = 13371337

PEPENETWORK_MAINNET = 9779

PEPERIUM_CHAIN_TESTNET = 4001

PEPE_CHAIN_MAINNET = 411

PERMISSION = 222

PGN_PUBLIC_GOODS_NETWORK = 424
```

PHALA_NETWORK = 2035

PHI_NETWORK_V1 = 4181

PHI_NETWORK_V2 = 144

PHOENIX_MAINNET = 13381

PIECE_TESTNET = 30067

PIRL = 3125659152

PIXIE_CHAIN_MAINNET = 6626

PIXIE_CHAIN_TESTNET = 666

PLANQ_MAINNET = 7070

PLATON_DEV_TESTNET2 = 2206132

PLATON_DEV_TESTNET_DEPRECATED = 2203181

PLATON_MAINNET = 210425

PLAYA3ULL_GAMES = 3011

PLIAN_MAINNET_MAIN = 2099156

PLIAN_MAINNET_SUBCHAIN_1 = 8007736

PLIAN_TESTNET_MAIN = 16658437

PLIAN_TESTNET_SUBCHAIN_1 = 10067275

PLINGA_MAINNET = 242

POA_NETWORK_CORE = 99

POA_NETWORK_SOKOL = 77

POCRNET = 2606

POLIS_MAINNET = 333999

POLIS_TESTNET = 333888

POLYGON = 137

POLYGON_SUPERNET_ARIANEE = 11891

POLYGON_ZKEVM = 1101

POLYGON_ZKEVM_TESTNET = 1442

POLYGON_ZKEVM_TESTNET_OLD = 1402

POLYGON_ZKEVM_TESTNET_PRE_AUDIT_UPGRADED = 1422

POLYJUICE_TESTNET = 71393

POLYSMARTCHAIN = 6999

```
POPCATEUM_MAINNET = 1213

PORTAL_FANTASY_CHAIN = 909

PORTAL_FANTASY_CHAIN_TEST = 808

POSICHAIN_DEVNET_SHARD_0 = 920000

POSICHAIN_DEVNET_SHARD_1 = 920001

POSICHAIN_MAINNET_SHARD_0 = 900000

POSICHAIN_TESTNET_SHARD_0 = 910000

PRIMUSCHAIN_MAINNET = 78

PROOF_OF_MEMES = 18159

PROTOJUMBO_TESTNET = 129

PROTON_TESTNET = 110

PROXY_NETWORK_TESTNET = 1031

PUBLICMINT_DEVNET = 2018

PUBLICMINT_MAINNET = 2020

PUBLICMINT_TESTNET = 2019

PULSECHAIN = 369

PULSECHAIN_TESTNET = 940

PULSECHAIN_TESTNET_V2B = 941

PULSECHAIN_TESTNET_V3 = 942

PULSECHAIN_TESTNET_V4 = 943

QEASYWEB3_TESTNET = 9528

QITMEER = 813

QITMEER_NETWORK_MIXNET = 8132

QITMEER_NETWORK_PRIVNET = 8133

QITMEER_NETWORK_TESTNET = 8131

QL1 = 766

QL1_TESTNET = 7668378

QUADRANS_BLOCKCHAIN = 10946

QUADRANS_BLOCKCHAIN_TESTNET = 10947

QUANTUM_CHAIN_MAINNET = 81720

QUANTUM_CHAIN_TESTNET = 12890
```

QUARIX = 8888888

QUARIX_TESTNET = 8888881

QUARKBLOCKCHAIN = 20181205

QUARKCHAIN_DEVNET_ROOT = 110000

QUARKCHAIN_DEVNET_SHARD_0 = 110001

QUARKCHAIN_DEVNET_SHARD_1 = 110002

QUARKCHAIN_DEVNET_SHARD_2 = 110003

QUARKCHAIN_DEVNET_SHARD_3 = 110004

QUARKCHAIN_DEVNET_SHARD_4 = 110005

QUARKCHAIN_DEVNET_SHARD_5 = 110006

QUARKCHAIN_DEVNET_SHARD_6 = 110007

QUARKCHAIN_DEVNET_SHARD_7 = 110008

QUARKCHAIN_MAINNET_ROOT = 100000

QUARKCHAIN_MAINNET_SHARD_0 = 100001

QUARKCHAIN_MAINNET_SHARD_1 = 100002

QUARKCHAIN_MAINNET_SHARD_2 = 100003

QUARKCHAIN_MAINNET_SHARD_3 = 100004

QUARKCHAIN_MAINNET_SHARD_4 = 100005

QUARKCHAIN_MAINNET_SHARD_5 = 100006

QUARKCHAIN_MAINNET_SHARD_6 = 100007

QUARKCHAIN_MAINNET_SHARD_7 = 100008

QUARTZ_BY_UNIQUE = 8881

QUOKKACOIN_MAINNET = 2077

Q_MAINNET = 35441

Q_TESTNET = 35443

RABA_NETWORK_MAINNET = 7484

RABBIT_ANALOG_TESTNET_CHAIN = 1807

RANGERS_PROTOCOL_MAINNET = 2025

RANGERS_PROTOCOL_TESTNET_ROBIN = 9527

RAPTORCHAIN = 1380996178

RAZOR_SKALE_CHAIN = 278611351

```
REALCHAIN_MAINNET = 121

REAPCHAIN_MAINNET = 221230

REAPCHAIN_TESTNET = 221231

REDBELLY_NETWORK_DEVNET = 152

REDBELLY_NETWORK_MAINNET = 151

REDBELLY_NETWORK_TESTNET = 153

REDBELLY_NETWORK_TGE = 154

REDECOIN = 1972

REDLIGHT_CHAIN_MAINNET = 2611

REDSTONE_HOLESKY_TESTNET = 17001

REI_CHAIN_MAINNET = 55555

REI_CHAIN_TESTNET = 55556

REI_NETWORK = 47805

RESINCOIN_MAINNET = 75000

RIKEZA_NETWORK_MAINNET = 1433

RIKEZA_NETWORK_TESTNET = 12715

RINIA_TESTNET = 917

RINIA_TESTNET_OLD = 9170

RINKEBY = 4

RISE_OF_THE_WARBOTS_TESTNET = 7777

ROLLUX_MAINNET = 570

ROLLUX_TESTNET = 57000

ROOTSTOCK_MAINNET = 30

ROOTSTOCK_TESTNET = 31

ROPSTEN = 3

RUBY_SMART_CHAIN_MAINNET = 1821

RUBY_SMART_CHAIN_TESTNET = 1912

RUPAYA = 499

SAAKURU_MAINNET = 7225878

SAAKURU_TESTNET = 247253

SAKURA = 1022
```

SANR_CHAIN = 11888

SAPPHIRE_BY_UNIQUE = 8883

SARDIS_MAINNET = 51712

SARDIS_TESTNET = 11612

SATOSHICHAIN_MAINNET = 12009

SATOSHICHAIN_TESTNET = 5758

SATOSHIE = 1985

SATOSHIE_TESTNET = 1986

SCALIND = 1911

SCALIND_TESTNET = 220

SCOLCOIN_MAINNET = 65450

SCOLCOIN_WEICHAIN_TESTNET = 6552

SCRIPT_TESTNET = 742

SCROLL = 534352

SCROLL_ALPHA_TESTNET = 534353

SCROLL_PRE_ALPHA_TESTNET = 534354

SCROLL_SEPOLIA_TESTNET = 534351

SECURECHAIN_MAINNET = 34

SECURECHAIN_TESTNET = 3434

SEELE_MAINNET = 186

SEI_DEVNET = 713715

SENJEPOWERS_MAINNET = 3699

SENJEPOWERS_TESTNET = 3698

SEPOLIA = 11155111

SEPOLIA_PGN_PUBLIC_GOODS_NETWORK = 58008

SETHEUM = 258

SHARDEUM_LIBERTY_1_X = 8080

SHARDEUM_LIBERTY_2_X = 8081

SHARDEUM_SPHINX_1_X = 8082

SHERPAX_MAINNET = 1506

SHERPAX_TESTNET = 1507

```
    SHIBACHAIN = 27

    SHIBARIUM = 109

    SHIBARIUM_BETA = 719

    SHIDEN = 336

    SHIMMEREVM = 148

    SHIMMEREVM_TESTNET = 1073

    SHIMMEREVM_TESTNET_DEPRECATED = 1071

    SHIMMEREVM_TESTNET_DEPRECATED_1072 = 1072

    SHINARIUM_BETA = 534849

    SHINARIUM_MAINNET = 214

    SHRAPNEL_SUBNET = 2044

    SHRAPNEL_TESTNET = 2038

    SHYFT_MAINNET = 7341

    SHYFT_TESTNET = 11437

    SIBERIUM_NETWORK = 111111

    SIBERIUM_TEST_NETWORK = 111000

    SINGULARITY_ZERO_MAINNET = 12052

    SINGULARITY_ZERO_TESTNET = 12051

    SIRIUSNET = 67390

    SIRIUSNET_V2 = 217

    SIX_PROTOCOL = 98

    SIX_PROTOCOL_TESTNET = 150

    SJATSH = 10086

    SKALE_CALYPSO_HUB = 1564830818

    SKALE_CALYPSO_HUB_TESTNET = 344106930

    SKALE_EUROPA_HUB = 2046399126

    SKALE_EUROPA_HUB_TESTNET = 476158412

    SKALE_NEBULA_HUB = 1482601649

    SKALE_NEBULA_HUB_TESTNET = 503129905

    SKALE_TITAN_HUB = 1350216234

    SKALE_TITAN_HUB_TESTNET = 1517929550
```

SMARTMESH_MAINNET = 20180430

SMART_BITCOIN_CASH = 10000

SMART_BITCOIN_CASH_TESTNET = 10001

SMART_HOST_TEKNOLOJI_TESTNET = 1177

SMART_LAYER_NETWORK = 5169

SMART_LAYER_NETWORK_TESTNET = 82459

SMART_TRADE_NETWORKS = 18122

SOCIAL_SMART_CHAIN_MAINNET = 281121

SOMA_NETWORK_MAINNET = 2332

SOMA_NETWORK_TESTNET = 2323

SONGBIRD_CANARY_NETWORK = 19

SOTERONE_MAINNET = 68

SOTERONE_MAINNET_OLD = 218

SOVERUN_MAINNET = 10101010

SOVERUN_TESTNET = 101010

SPORTS_CHAIN_NETWORK = 1904

SPS = 13000

SPS_TESTNET = 14000

STAR_SOCIAL_TESTNET = 700

STEP_NETWORK = 1234

STEP_TESTNET = 12345

STORAGECHAIN_MAINNET = 8726

STORAGECHAIN_TESTNET = 8727

STRATOS = 2048

STRATOS_TESTNET = 2047

STREAMUX_BLOCKCHAIN = 8098

STRUCTX_MAINNET = 208

SUPER_SMART_CHAIN_MAINNET = 1970

SUPER_SMART_CHAIN_TESTNET = 1969

SUR_BLOCKCHAIN_NETWORK = 262

SUSONO = 13812

```
SWAPDEX = 230

SWISSDLT = 94

SWISSTRONIK_TESTNET = 1291

SX_NETWORK_MAINNET = 416

SX_NETWORK_TESTNET = 647

SYMPLEXIA_SMART_CHAIN = 1149

SYNAPSE_CHAIN_TESTNET = 444

SYSCOIN_MAINNET = 57

SYSCOIN_TANENBAUM_TESTNET = 5700

TAF_ECO_CHAIN_MAINNET = 224168

TAIKO_ALPHA_2_TESTNET = 167004

TAIKO_ELDFELL_L3 = 167006

TAIKO_GRIMSVOTN_L2 = 167005

TAIKO_JOLNIR_L2 = 167007

TAIKO_KATLA_L2 = 167008

TANGLE_TESTNET = 3799

TANSSI_EVM_CONTAINERCHAIN = 5678

TAO_NETWORK = 558

TARAXA_MAINNET = 841

TARAXA_TESTNET = 842

TAYCAN = 22023

TAYCAN_TESTNET = 2023

TBSI_MAINNET = 1707

TBSI_TESTNET = 1708

TBWG_CHAIN = 35

TCG_VERSE_MAINNET = 2400

TECHPAY_MAINNET = 2569

TECTUM_EMISSION_TOKEN = 1003

TELEPORT = 8000

TELEPORT_TESTNET = 8001

TELOS_EVM_MAINNET = 40
```

TELOS_EVM_TESTNET = 41

TENET = 1559

TENET_TESTNET = 155

TEN_TESTNET = 443

TESLAFUNDS = 1856

TESTNET_BEONE_CHAIN = 8181

THAICHAIN = 7

THAICHAIN_2_0_THAIFI = 17

THETA_AMBER_TESTNET = 364

THETA_MAINNET = 361

THETA_SAPPHIRE_TESTNET = 363

THETA_TESTNET = 365

THE_ROOT_NETWORK_MAINNET = 7668

THE_ROOT_NETWORK_PORCINI_TESTNET = 7672

THINKIUM_MAINNET_CHAIN_0 = 70000

THINKIUM_MAINNET_CHAIN_1 = 70001

THINKIUM_MAINNET_CHAIN_103 = 70103

THINKIUM_MAINNET_CHAIN_2 = 70002

THINKIUM_TESTNET_CHAIN_0 = 60000

THINKIUM_TESTNET_CHAIN_1 = 60001

THINKIUM_TESTNET_CHAIN_103 = 60103

THINKIUM_TESTNET_CHAIN_2 = 60002

THUNDERCORE_MAINNET = 108

THUNDERCORE_TESTNET = 18

TILTYARD_SUBNET = 1127469

TIPBOXCOIN_MAINNET = 404040

TIPBOXCOIN_TESTNET = 4141

TITAN = 55004

TLCHAIN_NETWORK_MAINNET = 5177

TMY_CHAIN = 8768

TOKI_NETWORK = 8654

```
    TOKI_TESTNET = 8655

    TOMB_CHAIN_MAINNET = 6969

    TOMOCHAIN = 88

    TOMOCHAIN_TESTNET = 89

    TOOL_GLOBAL_MAINNET = 8723

    TOOL_GLOBAL_TESTNET = 8724

    TOP_MAINNET = 989

    TOP_MAINNET_EVM = 980

    TORONET_MAINNET = 77777

    TORONET_TESTNET = 54321

    TORUS_MAINNET = 8192

    TORUS_TESTNET = 8194

    TREASURENET_MAINNET_ALPHA = 5002

    TREASURENET_TESTNET = 5005

    TRES_MAINNET = 6066

    TRES_TESTNET = 6065

    TRITANIUM_TESTNET = 5353

    TRUST_EVM_TESTNET = 15555

    TTCOIN_SMART_CHAIN_MAINNET = 330844

    TURKEY_DEMO_DEV = 1731313

    T_EKTA = 1004

    T_E_A_M_BLOCKCHAIN = 88888888

    U2U_SOLARIS_MAINNET = 39

    UBIQ = 8

    UBIQ_NETWORK_TESTNET = 9

    UB_SMART_CHAIN = 99999

    UB_SMART_CHAIN_TESTNET = 99998

    ULTRA_PRO_MAINNET = 473861

    ULTRON_MAINNET = 1231

    ULTRON_TESTNET = 1230

    UNICORN_ULTRA_NEBULAS_TESTNET = 2484
```

UNIQUE = 8880

UNKNOWN = -1

UNREAL_TESTNET = 18231

UPTICK_MAINNET = 117

UPTN = 6119

UPTN_TESTNET = 6118

UZMI_NETWORK_MAINNET = 5315

VALORBIT = 38

VCHAIN_MAINNET = 2223

VECHAIN = 100009

VECHAIN_TESTNET = 100010

VELA1_CHAIN_MAINNET = 555

VELAS_EVM_MAINNET = 106

VELO_LABS_MAINNET = 56789

VENIDIUM_MAINNET = 4919

VENIDIUM_TESTNET = 4918

VENTION_SMART_CHAIN_MAINNET = 77612

VENTION_SMART_CHAIN_TESTNET = 741

VEX_EVM_TESTNET = 5522

VINE_TESTNET = 601

VINUCHAIN_NETWORK = 207

VINUCHAIN_TESTNET = 206

VISION_MAINNET = 888888

VISION_VPIONEER_TEST_CHAIN = 666666

VULTURE_EVM_BETA = 3102

VYVO_SMART_CHAIN = 8889

W3GAMEZ_HOLESKY_TESTNET = 32001

WAGMI = 11111

WANCHAIN = 888

WANCHAIN_TESTNET = 999

WEB3GAMES_DEVNET = 105

```
WEB3GAMES_TESTNET = 102

WEB3Q_GALILEO = 3334

WEB3Q_MAINNET = 333

WEB3Q_TESTNET = 3333

WEBCHAIN = 24484

WEELINK_TESTNET = 444900

WEGOCHAIN_RUBIDIUM_MAINNET = 5869

WEMIX3_0_MAINNET = 1111

WEMIX3_0_TESTNET = 1112

WHITEBIT_NETWORK = 1875

WHITEBIT_NETWORK_TESTNET = 2625

WIRESHAPE_FLORIPA_TESTNET = 49049

WOOPCHAIN_MAINNET = 139

WORLDLAND_MAINNET = 103

WORLDLAND_TESTNET = 10395

WORLDS_CALDERA = 4281033

WORLD_TRADE_TECHNICAL_CHAIN_MAINNET = 1202

WYZTH_TESTNET = 309

X1_DEVNET = 202212

X1_FASTNET = 4003

X1_MAINNET = 196

X1_NETWORK = 204005

X1_TESTNET = 195

XANACHAIN = 8888

XCAP = 9322252

XDC_APOTHEM_NETWORK = 51

XDC_NETWORK = 50

XEROM = 1313500

XODEX = 2415

XPLA_MAINNET = 37

XPLA_TESTNET = 3701
```

XT_SMART_CHAIN_MAINNET = 520

YIDARK_CHAIN_MAINNET = 927

YOOLDO_VERSE_MAINNET = 50005

YOOLDO_VERSE_TESTNET = 50006

YUANCHAIN_MAINNET = 3999

ZAFIRIUM_MAINNET = 1369

ZCORE_TESTNET = 3331

ZEETH_CHAIN = 427

ZEETH_CHAIN_DEV = 859

ZENIQ = 383414847825

ZENITH_MAINNET = 79

ZETACHAIN_ATHENS_3_TESTNET = 7001

ZETACHAIN_MAINNET = 7000

ZHEJIANG = 1337803

ZILLIQA_2_EVM_DEVNET = 33469

ZILLIQA_EVM = 32769

ZILLIQA_EVM_DEVNET = 33385

ZILLIQA_EVM_ISOLATED_SERVER = 32990

ZILLIQA_EVM_TESTNET = 33101

ZKATANA = 1261120

ZKFAIR_MAINNET = 42766

ZKFAIR_TESTNET = 43851

ZKSYNC_ERA_GOERLI_TESTNET_DEPRECATED = 280

ZKSYNC_MAINNET = 324

ZKSYNC_SEPOLIA_TESTNET = 300

ZORA = 7777777

ZORA_SEPOLIA_TESTNET = 999999999

ZYX_MAINNET = 55

exception gnosis.eth.**EthereumNetworkNotSupported**

Bases: Exception

class gnosis.eth.**EthereumTxSent**(*tx_hash*, *tx*, *contract_address*)

Bases: NamedTuple

contract_address: ChecksumAddress | None

Alias for field number 2

tx: TxParams

Alias for field number 1

tx_hash: bytes

Alias for field number 0

exception gnosis.eth.**FromAddressNotFound**

Bases: EthereumClientException

exception gnosis.eth.**GasLimitExceeded**

Bases: EthereumClientException

exception gnosis.eth.**InsufficientFunds**

Bases: EthereumClientException

exception gnosis.eth.**InvalidERC20Info**

Bases: EthereumClientException

exception gnosis.eth.**InvalidERC721Info**

Bases: EthereumClientException

exception gnosis.eth.**InvalidNonce**

Bases: EthereumClientException

exception gnosis.eth.**NonceTooHigh**

Bases: *InvalidNonce*

exception gnosis.eth.**NonceTooLow**

Bases: *InvalidNonce*

exception gnosis.eth.**ReplacementTransactionUnderpriced**

Bases: EthereumClientException

exception gnosis.eth.**SenderAccountNotFoundInNode**

Bases: EthereumClientException

exception gnosis.eth.**TransactionAlreadyImported**

Bases: EthereumClientException

exception gnosis.eth.**TransactionQueueLimitReached**

Bases: EthereumClientException

class gnosis.eth.**TxSpeed**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

FAST = 4

FASTEST = 6

NORMAL = 3

SLOW = 2

SLOWEST = 0

VERY_FAST = 5

VERY_SLOW = 1

exception gnosis.eth.UnknownAccount

Bases: EthereumClientException

gnosis.safe package

Subpackages

Submodules

gnosis.safe.exceptions module

exception gnosis.safe.exceptions.CannotEstimateGas

Bases: *SafeServiceException*

exception gnosis.safe.exceptions.CannotRetrieveSafeInfoException

Bases: *SafeServiceException*

exception gnosis.safe.exceptions.CouldNotFinishInitialization

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.CouldNotPayGasWithEther

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.CouldNotPayGasWithToken

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.HashHasNotBeenApproved

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.InvalidChecksumAddress

Bases: *SafeServiceException*

exception gnosis.safe.exceptions.InvalidContractSignatureLocation

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.InvalidInternalTx

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.InvalidMultisigTx

Bases: *SafeServiceException*

exception gnosis.safe.exceptions.InvalidOwnerProvided

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.InvalidPaymentToken

Bases: *SafeServiceException*

exception gnosis.safe.exceptions.InvalidSignaturesProvided

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**MethodCanOnlyBeCalledFromThisContract**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**ModuleManagerException**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**NotEnoughSafeTransactionGas**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**OnlyOwnersCanApproveAHash**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**OwnerManagerException**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**SafeServiceException**

Bases: Exception

exception gnosis.safe.exceptions.**SafeTransactionFailedWhenGasPriceAndSafeTxGasEmpty**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**SignatureNotProvidedByOwner**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**SignaturesDataTooShort**

Bases: *InvalidMultisigTx*

exception gnosis.safe.exceptions.**ThresholdNeedsToBeDefined**

Bases: *InvalidMultisigTx*

gnosis.safe.multi_send module

class gnosis.safe.multi_send.**MultiSend**(*ethereum_client: EthereumClient | None = None*, *address: ChecksumAddress | None = None*, *call_only: bool = True*)

Bases: object

MULTISEND_ADDRESSES = ('0xA238CBeb142c10Ef7Ad8442C6D1f9E89e07e7761', '0x998739BFdAAdde7C933B942a68053933098f9EDa')

MULTISEND_CALL_ONLY_ADDRESSES = ('0x40A2aCCbd92BCA938b02010E17A5b8929b49130D', '0xA1dabEF33b3B82c7814B6D82A79e50F4AC44102B')

build_tx_data(*multi_send_txs: List[MultiSendTx]*) → bytes

Txs don't need to be valid to get through

Parameters

multi_send_txs –

Returns

static deploy_contract(*ethereum_client: EthereumClient*, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy proxy factory contract

Parameters

- **ethereum_client** –

- **deployer_account** – Ethereum Account

Returns
EthereumTxSent with the deployed contract address

dummy_w3 = <web3.main.Web3 object>

classmethod from_bytes(*encoded_multisend_txs: str | bytes*) → List[*MultiSendTx*]

Decodes one or more multisend transactions from *bytes transactions* (Abi decoded)

Parameters
encoded_multisend_txs –

Returns
List of MultiSendTxs

classmethod from_transaction_data(*multisend_data: str | bytes*) → List[*MultiSendTx*]

Decodes multisend transactions from transaction data (ABI encoded with selector)

Returns

get_contract()

property w3

class gnosis.safe.multi_send.**MultiSendOperation**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

CALL = 0

DELEGATE_CALL = 1

class gnosis.safe.multi_send.**MultiSendTx**(*operation: MultiSendOperation*, *to: ChecksumAddress*, *value: int*, *data: bytes | HexStr*, *old_encoding: bool = False*)

Bases: object

Wrapper for a single MultiSendTx

property data_length: int

property encoded_data

classmethod from_bytes(*encoded_multisend_tx: str | bytes*) → *MultiSendTx*

Decoded one MultiSend transaction. ABI must be used to get the *transactions* parameter and use that data for this function :param encoded_multisend_tx: :return:

gnosis.safe.proxy_factory module

class gnosis.safe.proxy_factory.**ProxyFactory**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: ContractBase

calculate_proxy_address(*master_copy: ChecksumAddress*, *initializer: bytes*, *salt_nonce: int*, *chain_specific: bool = False*) → ChecksumAddress

Calculate proxy address for calling deploy_proxy_contract_with_nonce

Parameters

- **master_copy** –
- **initializer** –
- **salt_nonce** –
- **chain_specific** – Calculate chain specific address (to prevent same address in other chains)

Returns

check_proxy_code(*address: ChecksumAddress*) → bool

Check if proxy bytecode matches any of the deployed by the supported Proxy Factories

Parameters

address – Ethereum address to check

Returns

`True` if proxy is valid, `False` otherwise

classmethod deploy_contract(*ethereum_client:* EthereumClient, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy Proxy Factory contract

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum Account

Returns

`EthereumTxSent` with the deployed contract address

deploy_proxy_contract(*deployer_account: LocalAccount*, *master_copy: ChecksumAddress*, *initializer: bytes = b''*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*) → *EthereumTxSent*

Deploy proxy contract via ProxyFactory using *createProxy* function (CREATE opcode)

Parameters

- **deployer_account** – Ethereum account
- **master_copy** – Address the proxy will point at
- **initializer** – Initializer for the deployed proxy
- **gas** – Gas
- **gas_price** – Gas Price
- **nonce** – Nonce

Returns

EthereumTxSent

deploy_proxy_contract_with_nonce(*deployer_account: LocalAccount*, *master_copy: ChecksumAddress*, *initializer: bytes = b''*, *salt_nonce: int | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*, *chain_specific: bool = False*) → *EthereumTxSent*

Deploy proxy contract via Proxy Factory using *createProxyWithNonce* (CREATE2 opcode)

Parameters

- **deployer_account** – Ethereum account

- **master_copy** – Address the proxy will point at
- **initializer** – Initializer for the deployed proxy
- **salt_nonce** – Uint256 for CREATE2 salt. If not provided, a random one will be used
- **gas** – Gas
- **gas_price** – Gas Price
- **nonce** – Nonce
- **chain_specific** – Calculate chain specific address (to prevent same address in other chains)

Returns
EthereumTxSent

get_deploy_function(*chain_specific: bool*) → ContractFunction

get_proxy_creation_code() → bytes

Returns
Creation code used for the Proxy deployment. With this it is easily possible to calculate predicted address.

get_proxy_runtime_code() → bytes

Returns
Runtime code of a deployed Proxy. For v1.4.1 onwards the method is not available, so *None* will be returned

class gnosis.safe.proxy_factory.**ProxyFactoryV100**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: *ProxyFactory*

get_contract_fn() → Callable[[Web3, ChecksumAddress], Contract]

Returns
Contract function to get the proper contract

class gnosis.safe.proxy_factory.**ProxyFactoryV111**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: *ProxyFactory*

get_contract_fn() → Callable[[Web3, ChecksumAddress], Contract]

Returns
Contract function to get the proper contract

class gnosis.safe.proxy_factory.**ProxyFactoryV130**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: *ProxyFactory*

get_contract_fn() → Callable[[Web3, ChecksumAddress], Contract]

Returns
Contract function to get the proper contract

class gnosis.safe.proxy_factory.**ProxyFactoryV141**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: *ProxyFactory*

deploy_proxy_contract(**args*, ***kwargs*)

Deprecated since version ``createProxy``: function was deprecated in v1.4.1, use deploy_proxy_contract_with_nonce

Parameters

- **args** –
- **kwargs** –

Returns

get_contract_fn() → Callable[[Web3, ChecksumAddress], Contract]

Returns
Contract function to get the proper contract

get_deploy_function(*chain_specific: bool*) → ContractFunction

get_proxy_runtime_code() → bytes | None

Returns
From v1.4.1 onwards the method is not available

Raises
NotImplementedError

gnosis.safe.safe module

class gnosis.safe.safe.**Safe**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: SafeCreator, ContractBase

Collection of methods and utilies to handle a Safe

FALLBACK_HANDLER_STORAGE_SLOT = 49122629484629529244014240937346711770925847994644146912111677022347558721749

GUARD_STORAGE_SLOT = 3352823778259228016306855622497251643928256301472236617564181492812329492192 8

SAFE_MESSAGE_TYPEHASH = b'`\xb3\xcb\xf8\xb4\xa2#\xd6\x8dd\x1b;m\xdf\x9a)\x8e\x7f3q\ x0c\xf3\xd3\xa9\xd1\x14kZaP\xfb\xca'

build_multisig_tx(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int = 0*, *safe_tx_gas: int = 0*, *base_gas: int = 0*, *gas_price: int = 0*, *gas_token: ChecksumAddress = '0x0000000000000000000000000000000000000000'*, *refund_receiver: ChecksumAddress = '0x0000000000000000000000000000000000000000'*, *signatures: bytes = b''*, *safe_nonce: int | None = None*) → *SafeTx*

Allows to execute a Safe transaction confirmed by required number of owners and then pays the account that submitted the transaction. The fees are always transfered, even if the user transaction fails

Parameters

- **to** – Destination address of Safe transaction
- **value** – Ether value of Safe transaction
- **data** – Data payload of Safe transaction
- **operation** – Operation type of Safe transaction
- **safe_tx_gas** – Gas that should be used for the Safe transaction

- **base_gas** – Gas costs for that are independent of the transaction execution (e.g. base transaction fee, signature check, payment of the refund)
- **gas_price** – Gas price that should be used for the payment calculation
- **gas_token** – Token address (or *0x000..000* if ETH) that is used for the payment
- **refund_receiver** – Address of receiver of gas payment (or *0x000..000* if tx.origin).
- **signatures** – Packed signature data ({bytes32 r}{bytes32 s}{uint8 v})
- **safe_nonce** – Nonce of the safe (to calculate hash)
- **safe_version** – Safe version (to calculate hash)

Returns
SafeTx

property chain_id: int

check_funds_for_tx_gas(*safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: str*) → bool

Check safe has enough funds to pay for a tx

Parameters

- **safe_tx_gas** – Safe tx gas
- **base_gas** – Data gas
- **gas_price** – Gas Price
- **gas_token** – Gas Token, to use token instead of ether for the gas

Returns
True if enough funds, *False* otherwise

classmethod deploy_contract(*ethereum_client:* EthereumClient, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy master contract. Takes deployer_account (if unlocked in the node) or the deployer private key Safe with version > v1.1.1 doesn't need to be initialized as it already has a constructor

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum account

Returns
`EthereumTxSent` with the deployed contract address

property domain_separator: bytes | None

Returns
EIP721 DomainSeparator for the Safe. Returns *None* if not supported (for Safes < 1.0.0)

estimate_tx_base_gas(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *gas_token: ChecksumAddress*, *estimated_tx_gas: int*) → int

Calculate gas costs that are independent of the transaction execution(e.g. base transaction fee, signature check, payment of the refund...)

Parameters

- **to** –
- **value** –

- **data** –
- **operation** –
- **gas_token** –
- **estimated_tx_gas** – gas calculated with *estimate_tx_gas*

Returns

estimate_tx_gas(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*) → int

Estimate tx gas. Use *requiredTxGas* on the Safe contract and fallbacks to *eth_estimateGas* if that method fails. Note: *eth_estimateGas* cannot estimate delegate calls

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –

Returns
Estimated gas for Safe inner tx

Raises
CannotEstimateGas

estimate_tx_gas_by_trying(*to: ChecksumAddress*, *value: int*, *data: bytes | str*, *operation: int*)

Try to get an estimation with Safe's *requiredTxGas*. If estimation is successful, try to set a gas limit and estimate again. If gas estimation is ok, same gas estimation should be returned, if it's less than required estimation will not be completed, so estimation was not accurate and gas limit needs to be increased.

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –

Returns
Estimated gas calling *requiredTxGas* setting a gas limit and checking if *eth_call* is successful

Raises
CannotEstimateGas

estimate_tx_gas_with_safe(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *gas_limit: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

Estimate tx gas using safe *requiredTxGas* method

Returns
int: Estimated gas

Raises
CannotEstimateGas: If gas cannot be estimated

Raises
ValueError: Cannot decode received data

estimate_tx_gas_with_web3(*to: ChecksumAddress*, *value: int*, *data: bytes | HexStr*) → int

Parameters

- **to** –
- **value** –
- **data** –

Returns
Estimation using web3 *estimate_gas*

get_message_hash(*message: str | Hash32*) → Hash32

Return hash of a message that can be signed by owners.

Parameters
message – Message that should be hashed. A `Hash32` must be provided for EIP191 or EIP712 messages

Returns
Message hash

abstract get_version() → str

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

retrieve_all_info(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → *SafeInfo*

Get all Safe info in the same batch call.

Parameters
block_identifier –

Returns

Raises
CannotRetrieveSafeInfoException

retrieve_code() → HexBytes

retrieve_domain_separator(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → str

retrieve_fallback_handler(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_guard(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_is_hash_approved(*owner: str*, *safe_hash: bytes*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_is_message_signed(*message_hash: Hash32*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_is_owner(*owner: str*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_master_copy_address(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_modules(*pagination: int | None = 50*, *max_modules_to_retrieve: int | None = 500*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[ChecksumAddress]

Get modules enabled on the Safe From v1.1.1:

- `getModulesPaginated` is available
- `getModules` returns only 10 modules

Parameters

- **pagination** – Number of modules to get per request
- **max_modules_to_retrieve** – Maximum number of modules to retrieve
- **block_identifier** –

Returns

List of module addresses

retrieve_nonce(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

retrieve_owners(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[str]

retrieve_threshold(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

retrieve_version(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → str

send_multisig_tx(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: ChecksumAddress*, *refund_receiver: ChecksumAddress*, *signatures: bytes*, *tx_sender_private_key: HexStr*, *tx_gas=None*, *tx_gas_price=None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → *EthereumTxSent*

Build and send Safe tx

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –
- **safe_tx_gas** –
- **base_gas** –
- **gas_price** –

- **gas_token** –
- **refund_receiver** –
- **signatures** –
- **tx_sender_private_key** –
- **tx_gas** – Gas for the external tx. If not, *(safe_tx_gas + data_gas) * 2* will be used
- **tx_gas_price** – Gas price of the external tx. If not, *gas_price* will be used
- **block_identifier** –

Returns
Tuple(tx_hash, tx)

Raises
InvalidMultisigTx: If user tx cannot go through the Safe

property simulate_tx_accessor_address: ChecksumAddress

class gnosis.safe.safe.**SafeInfo**(*address: eth_typing.evm.ChecksumAddress*, *fallback_handler: eth_typing.evm.ChecksumAddress*, *guard: eth_typing.evm.ChecksumAddress*, *master_copy: eth_typing.evm.ChecksumAddress*, *modules: List[eth_typing.evm.ChecksumAddress]*, *nonce: int*, *owners: List[eth_typing.evm.ChecksumAddress]*, *threshold: int*, *version: str*)

Bases: `object`

address: ChecksumAddress

fallback_handler: ChecksumAddress

guard: ChecksumAddress

master_copy: ChecksumAddress

modules: List[ChecksumAddress]

nonce: int

owners: List[ChecksumAddress]

threshold: int

version: str

class gnosis.safe.safe.**SafeV001**(*address: ChecksumAddress*, *ethereum_client: EthereumClient*, **args*, ***kwargs*)

Bases: *Safe*

static deploy_contract(*ethereum_client: EthereumClient*, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy master contract. Takes deployer_account (if unlocked in the node) or the deployer private key

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum account

Returns
EthereumTxSent with the deployed contract address

get_contract_fn() → Callable[[Web3, ChecksumAddress], Contract]

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

class gnosis.safe.safe.**SafeV100**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: *Safe*

static deploy_contract(*ethereum_client:* EthereumClient, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy master contract. Takes deployer_account (if unlocked in the node) or the deployer private key

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum account

Returns
EthereumTxSent with the deployed contract address

get_contract_fn() → Contract

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

class gnosis.safe.safe.**SafeV111**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: *Safe*

get_contract_fn() → Contract

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

class gnosis.safe.safe.**SafeV120**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: *Safe*

get_contract_fn() → Contract

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

class gnosis.safe.safe.**SafeV130**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: *Safe*

get_contract_fn() → Contract

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

class gnosis.safe.safe.**SafeV141**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: *Safe*

estimate_tx_gas_with_safe(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *gas_limit: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

Estimate tx gas. Use *SimulateTxAccesor* and *simulate* on the *CompatibilityFallHandler*

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –
- **gas_limit** –
- **block_identifier** –

Returns

get_contract_fn() → Contract

Returns
Contract function to get the proper contract

get_version()

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

gnosis.safe.safe_create2_tx module

exception gnosis.safe.safe_create2_tx.**InvalidERC20Token**

Bases: Exception

class gnosis.safe.safe_create2_tx.**SafeCreate2Tx**(*salt_nonce*, *owners*, *threshold*, *fallback_handler*, *master_copy_address*, *proxy_factory_address*, *payment_receiver*, *payment_token*, *payment*, *gas*, *gas_price*, *payment_token_eth_value*, *fixed_creation_cost*, *safe_address*, *safe_setup_data*)

Bases: NamedTuple

fallback_handler: str

Alias for field number 3

fixed_creation_cost: int | None

Alias for field number 12

gas: int

Alias for field number 9

gas_price: int

Alias for field number 10

master_copy_address: str

Alias for field number 4

owners: List[str]

Alias for field number 1

payment: int

Alias for field number 8

property payment_ether

payment_receiver: str

Alias for field number 6

payment_token: str

Alias for field number 7

payment_token_eth_value: float

Alias for field number 11

proxy_factory_address: str

Alias for field number 5

safe_address: str

Alias for field number 13

safe_setup_data: bytes

Alias for field number 14

salt_nonce: int

Alias for field number 0

threshold: int

Alias for field number 2

class gnosis.safe.safe_create2_tx.**SafeCreate2TxBuilder**(*w3: Web3*, *master_copy_address: ChecksumAddress*, *proxy_factory_address: ChecksumAddress*)

Bases: object

build(*owners: List[str]*, *threshold: int*, *salt_nonce: int*, *gas_price: int*, *fallback_handler: str | None = None*, *payment_receiver: str | None = None*, *payment_token: str | None = None*, *payment_token_eth_value: float = 1.0*, *fixed_creation_cost: int | None = None*)

Prepare Safe creation

Parameters

- **owners** – Owners of the Safe
- **threshold** – Minimum number of users required to operate the Safe
- **fallback_handler** – Handler for fallback calls to the Safe
- **salt_nonce** – Web3 instance
- **gas_price** – Gas Price
- **payment_receiver** – Address to refund when the Safe is created. Address(0) if no need to refund
- **payment_token** – Payment token instead of paying the funder with ether. If None Ether will be used
- **payment_token_eth_value** – Value of payment token per 1 Ether
- **fixed_creation_cost** – Fixed creation cost of Safe (Wei)

calculate_create2_address(*safe_setup_data: bytes*, *salt_nonce: int*)

gnosis.safe.safe_creation_tx module

gnosis.safe.safe_signature module

exception gnosis.safe.safe_signature.**CannotCheckEIP1271ContractSignature**

Bases: *SafeSignatureException*

class gnosis.safe.safe_signature.**SafeSignature**(*signature: bytes | str*, *safe_hash: bytes | str*)

Bases: ABC

export_signature() → HexBytes

Exports signature in a format that's valid individually. That's important for contract signatures, as it will fix the offset

Returns

classmethod export_signatures(*safe_signatures: Sequence[SafeSignature]*) → HexBytes

Takes a list of SafeSignature objects and exports them as a valid signature for the contract

Parameters
safe_signatures –

Returns
Valid signature for the Safe contract

abstract is_valid(*ethereum_client:* EthereumClient, *safe_address: str*) → bool

Parameters

- **ethereum_client** – Required for Contract Signature and Approved Hash check
- **safe_address** – Required for Approved Hash check

Returns
True if signature is valid, *False* otherwise

abstract property owner

Returns
Decode owner from signature, without any further validation (signature can be not valid)

classmethod parse_signature(*signatures: bytes | str*, *safe_hash: bytes | str*, *safe_hash_preimage: bytes | str | None = None*, *ignore_trailing: bool = True*) → List[*SafeSignature*]

Parameters

- **signatures** – One or more signatures appended. EIP1271 data at the end is supported.
- **safe_hash** – Signed hash for the Safe (message or transaction)
- **safe_hash_preimage** – `safe_hash` preimage for EIP1271 validation
- **ignore_trailing** – Ignore trailing data on the signature. Some libraries pad it and add some zeroes at the end

Returns
List of SafeSignatures decoded

abstract property signature_type: *SafeSignatureType*

class gnosis.safe.safe_signature.**SafeSignatureApprovedHash**(*signature: bytes | str*, *safe_hash: bytes | str*)

Bases: *SafeSignature*

classmethod build_for_owner(*owner: str*, *safe_hash: str*) → *SafeSignatureApprovedHash*

is_valid(*ethereum_client:* EthereumClient, *safe_address: str*) → bool

Parameters

- **ethereum_client** – Required for Contract Signature and Approved Hash check
- **safe_address** – Required for Approved Hash check

Returns
True if signature is valid, *False* otherwise

property owner

Returns
Decode owner from signature, without any further validation (signature can be not valid)

property signature_type

class gnosis.safe.safe_signature.**SafeSignatureContract**(*signature: bytes | str*, *safe_hash: bytes | str*, *safe_hash_preimage: bytes | str*, *contract_signature: bytes | str*)

Bases: *SafeSignature*

EIP1271_MAGIC_VALUE = HexBytes('0x20c13b0b')

EIP1271_MAGIC_VALUE_UPDATED = HexBytes('0x1626ba7e')

export_signature() → HexBytes

Fix offset (s) and append *contract_signature* at the end of the signature

Returns

classmethod from_values(*safe_owner: ChecksumAddress*, *safe_hash: bytes | str*, *safe_hash_preimage: bytes | str*, *contract_signature: bytes | str*) → *SafeSignatureContract*

is_valid(*ethereum_client:* EthereumClient, **args*) → bool

Parameters

- **ethereum_client** – Required for Contract Signature and Approved Hash check
- **safe_address** – Required for Approved Hash check

Returns

True if signature is valid, *False* otherwise

property owner: ChecksumAddress

Returns

Address of contract signing. No further checks to get the owner are needed, but it could be a non-existing contract

property signature_type: *SafeSignatureType*

class gnosis.safe.safe_signature.**SafeSignatureEOA**(*signature: bytes | str*, *safe_hash: bytes | str*)

Bases: *SafeSignature*

is_valid(**args*) → bool

Parameters

- **ethereum_client** – Required for Contract Signature and Approved Hash check
- **safe_address** – Required for Approved Hash check

Returns

True if signature is valid, *False* otherwise

property owner

Returns

Decode owner from signature, without any further validation (signature can be not valid)

property signature_type

class gnosis.safe.safe_signature.**SafeSignatureEthSign**(*signature: bytes | str*, *safe_hash: bytes | str*)

Bases: *SafeSignature*

is_valid(**args*) → bool

Parameters

- **ethereum_client** – Required for Contract Signature and Approved Hash check
- **safe_address** – Required for Approved Hash check

Returns
True if signature is valid, *False* otherwise

property owner

Returns
Decode owner from signature, without any further validation (signature can be not valid)

property signature_type

exception gnosis.safe.safe_signature.**SafeSignatureException**

Bases: Exception

class gnosis.safe.safe_signature.**SafeSignatureType**(*value*, *names=None*, *, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

APPROVED_HASH = 1

CONTRACT_SIGNATURE = 0

EOA = 2

ETH_SIGN = 3

static **from_v**(*v: int*)

gnosis.safe.safe_signature.**uint_to_address**(*value: int*) → ChecksumAddress

Convert a Solidity *uint* value to a checksummed *address*, removing invalid padding bytes if present

Returns
Checksummed address

gnosis.safe.safe_tx module

class gnosis.safe.safe_tx.**SafeTx**(*ethereum_client: EthereumClient*, *safe_address: ChecksumAddress*, *to: ChecksumAddress | None*, *value: int*, *data: bytes*, *operation: int*, *safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: ChecksumAddress | None*, *refund_receiver: ChecksumAddress | None*, *signatures: bytes | None = None*, *safe_nonce: int | None = None*, *safe_version: str | None = None*, *chain_id: int | None = None*)

Bases: object

call(*tx_sender_address: str | None = None*, *tx_gas: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

Parameters

- **tx_sender_address** –
- **tx_gas** – Force a gas limit
- **block_identifier** –

Returns
1 if everything ok

property chain_id: int

property contract

property eip712_structured_data: Dict[str, Any]

execute(*tx_sender_private_key: str*, *tx_gas: int | None = None*, *tx_gas_price: int | None = None*, *tx_nonce: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*, *eip1559_speed:* TxSpeed *| None = None*) → Tuple[HexBytes, TxParams]

Send multisig tx to the Safe

Parameters

- **tx_sender_private_key** – Sender private key
- **tx_gas** – Gas for the external tx. If not, *(safe_tx_gas + base_gas) * 2* will be used
- **tx_gas_price** – Gas price of the external tx. If not, *gas_price* will be used
- **tx_nonce** – Force nonce for *tx_sender*
- **block_identifier** – *latest* or *pending*
- **eip1559_speed** – If provided, use EIP1559 transaction

Returns

Tuple(tx_hash, tx)

Raises

InvalidMultisigTx: If user tx cannot go through the Safe

recommended_gas() → Wei

Returns

Recommended gas to use on the ethereum_tx

property safe_nonce: int

property safe_tx_hash: HexBytes

property safe_tx_hash_preimage: HexBytes

property safe_version: str

sign(*private_key: str*) → bytes

{bytes32 r}{bytes32 s}{uint8 v} :param private_key: :return: Signature

property signers: List[str]

property sorted_signers

unsign(*address: str*) → bool

property w3

property w3_tx: ContractFunction

Returns

Web3 contract tx prepared for *call*, *transact* or *build_transaction*

gnosis.safe.serializers module

class gnosis.safe.serializers.**SafeMultisigEstimateTxSerializer**(**args*, ***kwargs*)

Bases: Serializer

validate(*data*)

validate_operation(*value*)

class gnosis.safe.serializers.**SafeMultisigTxSerializer**(**args*, ***kwargs*)

Bases: *SafeMultisigEstimateTxSerializer*

DEPRECATED, use *SafeMultisigTxSerializerV1* instead

class gnosis.safe.serializers.**SafeMultisigTxSerializerV1**(**args*, ***kwargs*)

Bases: *SafeMultisigEstimateTxSerializer*

Version 1.0.0 of the Safe changes *data_gas* to *base_gas*

class gnosis.safe.serializers.**SafeSignatureSerializer**(**args*, ***kwargs*)

Bases: Serializer

When using safe signatures *v* can have more values

check_r(*r*)

check_s(*s*)

check_v(*v*)

validate(*data*)

validate_v(*v*)

gnosis.safe.signatures module

gnosis.safe.signatures.**get_signing_address**(*signed_hash: bytes | str*, *v: int*, *r: int*, *s: int*) → str

Returns
checksummed ethereum address, for example *0x568c93675A8dEb121700A6FAdDdfE7DFAb66Ae4A*

Return type
str or *NULL_ADDRESS* if signature is not valid

gnosis.safe.signatures.**signature_split**(*signatures: bytes | str*, *pos: int = 0*) → Tuple[int, int, int]

Parameters

- **signatures** – signatures in form of {bytes32 r}{bytes32 s}{uint8 v}
- **pos** – position of the signature

Returns
Tuple with v, r, s

gnosis.safe.signatures.**signature_to_bytes**(*v: int*, *r: int*, *s: int*) → bytes

Convert ecdsa signature to bytes :param v: :param r: :param s: :return: signature in form of {bytes32 r}{bytes32 s}{uint8 v}

gnosis.safe.signatures.**signatures_to_bytes**(*signatures: List[Tuple[int, int, int]]*) → bytes

Convert signatures to bytes :param signatures: list of tuples(v, r, s) :return: 65 bytes per signature

Module contents

exception gnosis.safe.**CannotEstimateGas**

Bases: *SafeServiceException*

exception gnosis.safe.**CouldNotPayGasWithEther**

Bases: *InvalidMultisigTx*

exception gnosis.safe.**InvalidChecksumAddress**

Bases: *SafeServiceException*

exception gnosis.safe.**InvalidInternalTx**

Bases: *InvalidMultisigTx*

exception gnosis.safe.**InvalidMultisigTx**

Bases: *SafeServiceException*

exception gnosis.safe.**InvalidPaymentToken**

Bases: *SafeServiceException*

exception gnosis.safe.**InvalidSignaturesProvided**

Bases: *InvalidMultisigTx*

class gnosis.safe.**ProxyFactory**(**args*, *version: str = '1.4.1'*, ***kwargs*)

Bases: ContractBase

calculate_proxy_address(*master_copy: ChecksumAddress*, *initializer: bytes*, *salt_nonce: int*, *chain_specific: bool = False*) → ChecksumAddress

Calculate proxy address for calling deploy_proxy_contract_with_nonce

Parameters

- **master_copy** –
- **initializer** –
- **salt_nonce** –
- **chain_specific** – Calculate chain specific address (to prevent same address in other chains)

Returns

check_proxy_code(*address: ChecksumAddress*) → bool

Check if proxy bytecode matches any of the deployed by the supported Proxy Factories

Parameters

address – Ethereum address to check

Returns

True if proxy is valid, False otherwise

classmethod deploy_contract(*ethereum_client:* EthereumClient, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy Proxy Factory contract

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum Account

Returns
EthereumTxSent with the deployed contract address

deploy_proxy_contract(*deployer_account: LocalAccount*, *master_copy: ChecksumAddress*, *initializer: bytes = b''*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*) → *EthereumTxSent*

Deploy proxy contract via ProxyFactory using *createProxy* function (CREATE opcode)

Parameters

- **deployer_account** – Ethereum account
- **master_copy** – Address the proxy will point at
- **initializer** – Initializer for the deployed proxy
- **gas** – Gas
- **gas_price** – Gas Price
- **nonce** – Nonce

Returns
EthereumTxSent

deploy_proxy_contract_with_nonce(*deployer_account: LocalAccount*, *master_copy: ChecksumAddress*, *initializer: bytes = b''*, *salt_nonce: int | None = None*, *gas: int | None = None*, *gas_price: int | None = None*, *nonce: int | None = None*, *chain_specific: bool = False*) → *EthereumTxSent*

Deploy proxy contract via Proxy Factory using *createProxyWithNonce* (CREATE2 opcode)

Parameters

- **deployer_account** – Ethereum account
- **master_copy** – Address the proxy will point at
- **initializer** – Initializer for the deployed proxy
- **salt_nonce** – Uint256 for CREATE2 salt. If not provided, a random one will be used
- **gas** – Gas
- **gas_price** – Gas Price
- **nonce** – Nonce
- **chain_specific** – Calculate chain specific address (to prevent same address in other chains)

Returns
EthereumTxSent

get_deploy_function(*chain_specific: bool*) → ContractFunction

get_proxy_creation_code() → bytes

Returns
Creation code used for the Proxy deployment. With this it is easily possible to calculate predicted address.

get_proxy_runtime_code() → bytes

Returns
Runtime code of a deployed Proxy. For v1.4.1 onwards the method is not available, so *None* will be returned

class gnosis.safe.**Safe**(*address: ChecksumAddress*, *ethereum_client:* EthereumClient, **args*, ***kwargs*)

Bases: SafeCreator, ContractBase

Collection of methods and utilies to handle a Safe

FALLBACK_HANDLER_STORAGE_SLOT = 49122629484629529244014240937346711770925847994644146912111677022347558721749

GUARD_STORAGE_SLOT = 3352823778259228016306855622497251643928256301472236617564181492812329492192 8

SAFE_MESSAGE_TYPEHASH = b'`\xb3\xcb\xf8\xb4\xa2#\xd6\x8dd\x1b;m\xdf\x9a)\x8e\x7f3q\x0c\xf3\xd3\xa9\xd1\x14kZaP\xfb\xca'

build_multisig_tx(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int = 0*, *safe_tx_gas: int = 0*, *base_gas: int = 0*, *gas_price: int = 0*, *gas_token: ChecksumAddress = '0x0000000000000000000000000000000000000000'*, *refund_receiver: ChecksumAddress = '0x0000000000000000000000000000000000000000'*, *signatures: bytes = b''*, *safe_nonce: int | None = None*) → *SafeTx*

Allows to execute a Safe transaction confirmed by required number of owners and then pays the account that submitted the transaction. The fees are always transfered, even if the user transaction fails

Parameters

- **to** – Destination address of Safe transaction
- **value** – Ether value of Safe transaction
- **data** – Data payload of Safe transaction
- **operation** – Operation type of Safe transaction
- **safe_tx_gas** – Gas that should be used for the Safe transaction
- **base_gas** – Gas costs for that are independent of the transaction execution (e.g. base transaction fee, signature check, payment of the refund)
- **gas_price** – Gas price that should be used for the payment calculation
- **gas_token** – Token address (or *0x000..000* if ETH) that is used for the payment
- **refund_receiver** – Address of receiver of gas payment (or *0x000..000* if tx.origin).
- **signatures** – Packed signature data ({bytes32 r}{bytes32 s}{uint8 v})
- **safe_nonce** – Nonce of the safe (to calculate hash)
- **safe_version** – Safe version (to calculate hash)

Returns
SafeTx

property chain_id: int

check_funds_for_tx_gas(*safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: str*) → bool

Check safe has enough funds to pay for a tx

Parameters

- **safe_tx_gas** – Safe tx gas

- **base_gas** – Data gas
- **gas_price** – Gas Price
- **gas_token** – Gas Token, to use token instead of ether for the gas

Returns
True if enough funds, *False* otherwise

classmethod deploy_contract(*ethereum_client:* EthereumClient, *deployer_account: LocalAccount*) → *EthereumTxSent*

Deploy master contract. Takes deployer_account (if unlocked in the node) or the deployer private key Safe with version > v1.1.1 doesn't need to be initialized as it already has a constructor

Parameters

- **ethereum_client** –
- **deployer_account** – Ethereum account

Returns
`EthereumTxSent` with the deployed contract address

property domain_separator: bytes | None

Returns
EIP721 DomainSeparator for the Safe. Returns *None* if not supported (for Safes < 1.0.0)

estimate_tx_base_gas(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *gas_token: ChecksumAddress*, *estimated_tx_gas: int*) → int

Calculate gas costs that are independent of the transaction execution(e.g. base transaction fee, signature check, payment of the refund...)

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –
- **gas_token** –
- **estimated_tx_gas** – gas calculated with *estimate_tx_gas*

Returns

estimate_tx_gas(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*) → int

Estimate tx gas. Use *requiredTxGas* on the Safe contract and fallbacks to *eth_estimateGas* if that method fails. Note: *eth_estimateGas* cannot estimate delegate calls

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –

Returns
Estimated gas for Safe inner tx

Raises
CannotEstimateGas

`estimate_tx_gas_by_trying`(*to: ChecksumAddress*, *value: int*, *data: bytes | str*, *operation: int*)

Try to get an estimation with Safe's *requiredTxGas*. If estimation is successful, try to set a gas limit and estimate again. If gas estimation is ok, same gas estimation should be returned, if it's less than required estimation will not be completed, so estimation was not accurate and gas limit needs to be increased.

Parameters

- `to` –
- `value` –
- `data` –
- `operation` –

Returns
Estimated gas calling *requiredTxGas* setting a gas limit and checking if *eth_call* is successful

Raises
CannotEstimateGas

`estimate_tx_gas_with_safe`(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *gas_limit: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

Estimate tx gas using safe *requiredTxGas* method

Returns
int: Estimated gas

Raises
CannotEstimateGas: If gas cannot be estimated

Raises
ValueError: Cannot decode received data

`estimate_tx_gas_with_web3`(*to: ChecksumAddress*, *value: int*, *data: bytes | HexStr*) → int

Parameters

- `to` –
- `value` –
- `data` –

Returns
Estimation using web3 *estimate_gas*

`get_message_hash`(*message: str | Hash32*) → Hash32

Return hash of a message that can be signed by owners.

Parameters
`message` – Message that should be hashed. A `Hash32` must be provided for EIP191 or EIP712 messages

Returns
Message hash

abstract get_version() → str

Returns
String with Safe Master Copy semantic version, must match *retrieve_version()*

retrieve_all_info(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → *SafeInfo*

Get all Safe info in the same batch call.

Parameters
block_identifier –

Returns

Raises
CannotRetrieveSafeInfoException

retrieve_code() → HexBytes

retrieve_domain_separator(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → str

retrieve_fallback_handler(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_guard(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_is_hash_approved(*owner: str*, *safe_hash: bytes*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_is_message_signed(*message_hash: Hash32*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_is_owner(*owner: str*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → bool

retrieve_master_copy_address(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → ChecksumAddress

retrieve_modules(*pagination: int | None = 50*, *max_modules_to_retrieve: int | None = 500*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[ChecksumAddress]

Get modules enabled on the Safe From v1.1.1:

- `getModulesPaginated` is available
- `getModules` returns only 10 modules

Parameters

- **pagination** – Number of modules to get per request
- **max_modules_to_retrieve** – Maximum number of modules to retrieve
- **block_identifier** –

Returns
List of module addresses

retrieve_nonce(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

retrieve_owners(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → List[str]

retrieve_threshold(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

retrieve_version(*block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → str

send_multisig_tx(*to: ChecksumAddress*, *value: int*, *data: bytes*, *operation: int*, *safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: ChecksumAddress*, *refund_receiver: ChecksumAddress*, *signatures: bytes*, *tx_sender_private_key: HexStr*, *tx_gas=None*, *tx_gas_price=None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → *EthereumTxSent*

Build and send Safe tx

Parameters

- **to** –
- **value** –
- **data** –
- **operation** –
- **safe_tx_gas** –
- **base_gas** –
- **gas_price** –
- **gas_token** –
- **refund_receiver** –
- **signatures** –
- **tx_sender_private_key** –
- **tx_gas** – Gas for the external tx. If not, *(safe_tx_gas + data_gas) * 2* will be used
- **tx_gas_price** – Gas price of the external tx. If not, *gas_price* will be used
- **block_identifier** –

Returns
Tuple(tx_hash, tx)

Raises
InvalidMultisigTx: If user tx cannot go through the Safe

property simulate_tx_accessor_address: ChecksumAddress

class gnosis.safe.**SafeOperationEnum**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

CALL = 0

CREATE = 2

DELEGATE_CALL = 1

exception gnosis.safe.**SafeServiceException**

Bases: Exception

class gnosis.safe.**SafeTx**(*ethereum_client: EthereumClient*, *safe_address: ChecksumAddress*, *to: ChecksumAddress | None*, *value: int*, *data: bytes*, *operation: int*, *safe_tx_gas: int*, *base_gas: int*, *gas_price: int*, *gas_token: ChecksumAddress | None*, *refund_receiver: ChecksumAddress | None*, *signatures: bytes | None = None*, *safe_nonce: int | None = None*, *safe_version: str | None = None*, *chain_id: int | None = None*)

Bases: object

call(*tx_sender_address: str | None = None*, *tx_gas: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*) → int

Parameters

- **tx_sender_address** –
- **tx_gas** – Force a gas limit
- **block_identifier** –

Returns

1 if everything ok

property chain_id: int

property contract

property eip712_structured_data: Dict[str, Any]

execute(*tx_sender_private_key: str*, *tx_gas: int | None = None*, *tx_gas_price: int | None = None*, *tx_nonce: int | None = None*, *block_identifier: Literal['latest', 'earliest', 'pending', 'safe', 'finalized'] | BlockNumber | Hash32 | HexStr | HexBytes | int | None = 'latest'*, *eip1559_speed: TxSpeed | None = None*) → Tuple[HexBytes, TxParams]

Send multisig tx to the Safe

Parameters

- **tx_sender_private_key** – Sender private key
- **tx_gas** – Gas for the external tx. If not, *(safe_tx_gas + base_gas) * 2* will be used
- **tx_gas_price** – Gas price of the external tx. If not, *gas_price* will be used
- **tx_nonce** – Force nonce for *tx_sender*
- **block_identifier** – *latest* or *pending*
- **eip1559_speed** – If provided, use EIP1559 transaction

Returns

Tuple(tx_hash, tx)

Raises

InvalidMultisigTx: If user tx cannot go through the Safe

recommended_gas() → Wei

Returns
Recommended gas to use on the ethereum_tx

property safe_nonce: int

property safe_tx_hash: HexBytes

property safe_tx_hash_preimage: HexBytes

property safe_version: str

sign(*private_key: str*) → bytes
{bytes32 r}{bytes32 s}{uint8 v} :param private_key: :return: Signature

property signers: List[str]

property sorted_signers

unsign(*address: str*) → bool

property w3

property w3_tx: ContractFunction

Returns
Web3 contract tx prepared for *call*, *transact* or *build_transaction*

exception gnosis.safe.**SignatureNotProvidedByOwner**
Bases: *InvalidMultisigTx*

Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

gnosis, 127
gnosis.eth, 53
gnosis.eth.clients, 13
gnosis.eth.clients.blockscout_client, 7
gnosis.eth.clients.contract_metadata, 9
gnosis.eth.clients.etherscan_client, 9
gnosis.eth.clients.sourcify_client, 12
gnosis.eth.constants, 36
gnosis.eth.contracts, 18
gnosis.eth.django, 23
gnosis.eth.django.serializers, 22
gnosis.eth.django.validators, 23
gnosis.eth.ethereum_client, 36
gnosis.eth.oracles, 28
gnosis.eth.oracles.abis, 23
gnosis.eth.oracles.abis.aave_abis, 23
gnosis.eth.oracles.abis.balancer_abis, 23
gnosis.eth.oracles.abis.curve_abis, 23
gnosis.eth.oracles.abis.mooniswap_abis, 23
gnosis.eth.oracles.abis.yearn_abis, 23
gnosis.eth.oracles.oracles, 23
gnosis.eth.typing, 51
gnosis.eth.utils, 52
gnosis.safe, 119
gnosis.safe.exceptions, 99
gnosis.safe.multi_send, 100
gnosis.safe.proxy_factory, 101
gnosis.safe.safe, 104
gnosis.safe.safe_create2_tx, 112
gnosis.safe.safe_signature, 113
gnosis.safe.safe_tx, 116
gnosis.safe.serializers, 118
gnosis.safe.signatures, 118

INDEX

A

AaveOracle (*class in gnosis.eth.oracles*), 28
AaveOracle (*class in gnosis.eth.oracles.oracles*), 23
abi (*gnosis.eth.clients.contract_metadata.ContractMetadata attribute*), 9
abi (*gnosis.eth.clients.ContractMetadata attribute*), 15
ACALA_MANDALA_TESTNET_TC9 (*gnosis.eth.EthereumNetwork attribute*), 58
ACALA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 58
ACALA_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 58
ACRIA_INTELLICHAIN (*gnosis.eth.EthereumNetwork attribute*), 58
address (*gnosis.eth.oracles.oracles.UnderlyingToken attribute*), 25
address (*gnosis.eth.oracles.UnderlyingToken attribute*), 32
address (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
address (*gnosis.safe.safe.SafeInfo attribute*), 109
ADDRESSES (*gnosis.eth.oracles.KyberOracle attribute*), 30
ADDRESSES (*gnosis.eth.oracles.oracles.UniswapOracle attribute*), 25
ADDRESSES (*gnosis.eth.oracles.UniswapOracle attribute*), 32
ADIL_CHAIN_V2_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 58
ADIL_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 58
ADIL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 58
AERIE_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 58
AEROCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 58
AGUNG_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 58
AIA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
AIOZ_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 59
AIOZ_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
AIRDAO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AIRDAO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
AKROMA (*gnosis.eth.EthereumNetwork attribute*), 59
ALAYA_DEV_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALAYA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALGOL (*gnosis.eth.EthereumNetwork attribute*), 59
ALL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALPH_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 59
ALPHABET_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALTAIR (*gnosis.eth.EthereumNetwork attribute*), 59
ALTCOINCHAIN (*gnosis.eth.EthereumNetwork attribute*), 59
ALTERIUM_L2_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALTLAYER_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALTLAYER_ZERO_GAS_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 59
ALVEYCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALYX_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ALYX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMANA (*gnosis.eth.EthereumNetwork attribute*), 59
AMANA_MIXNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMANA_PRIVNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMANA_TESTNET (*gnosis.eth.EthereumNetwork attribute*),

59
AMBROS_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AME_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMPLIFY_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMSTAR_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 59
AMSTAR_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ANCIENT8_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 59
ANCIENT8_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 60
ANDUSCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
ANTOFY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
ANTOFY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ANYTYPE_EVM_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 60
APPROVED_HASH (*gnosis.safe.safe_signature.SafeSignatureType attribute*), 116
AQUACHAIN (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_GOERLI (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_NOVA (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_ON_XDAI (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_ONE (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_RINKEBY (*gnosis.eth.EthereumNetwork attribute*), 60
ARBITRUM_SEPOLIA (*gnosis.eth.EthereumNetwork attribute*), 60
ARC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARC_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARCOLOGY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARCTURUS_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARDENIUM_ATHENA (*gnosis.eth.EthereumNetwork attribute*), 60
AREON_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
AREON_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
AREVIA (*gnosis.eth.EthereumNetwork attribute*), 60
args (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
ARMONIA_EVA_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARMONIA_EVA_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARTHERA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARTHERA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ARTIS_SIGMA1 (*gnosis.eth.EthereumNetwork attribute*), 60
ARTIS_TESTNET_TAU1 (*gnosis.eth.EthereumNetwork attribute*), 60
ARZIO_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 60
ASTAR (*gnosis.eth.EthereumNetwork attribute*), 60
ASTAR_ZKEVM (*gnosis.eth.EthereumNetwork attribute*), 60
ASTRA (*gnosis.eth.EthereumNetwork attribute*), 60
ASTRA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 60
ASTRIA_EVM_DUSKNET (*gnosis.eth.EthereumNetwork attribute*), 61
ATELIER (*gnosis.eth.EthereumNetwork attribute*), 61
ATHEIOS (*gnosis.eth.EthereumNetwork attribute*), 61
ATHEREUM (*gnosis.eth.EthereumNetwork attribute*), 61
ATOSHI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AURORA_BETANET (*gnosis.eth.EthereumNetwork attribute*), 61
AURORA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 61
AURORA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AUTOBAHN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 61
AUTONITY_BAKERLOO_BARADA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AUTONITY_BAKERLOO_THAMES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AUTONITY_PICCADILLY_BARADA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AUTONITY_PICCADILLY_THAMES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AUXILIUM_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 61
AVALANCHE_C_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 61
AVALANCHE_FUJI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
AVES_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 61
AVES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61

AVOCADO (*gnosis.eth.EthereumNetwork attribute*), 61
AXELCHAIN_DEV_NET (*gnosis.eth.EthereumNetwork attribute*), 61

B

BAHAMUT (*gnosis.eth.EthereumNetwork attribute*), 61
balance (*gnosis.eth.ethereum_client.TokenBalance attribute*), 47
balance (*gnosis.eth.typing.BalanceDict attribute*), 51
BalanceDict (*class in gnosis.eth.typing*), 51
BalancerOracle (*class in gnosis.eth.oracles*), 29
BalancerOracle (*class in gnosis.eth.oracles.oracles*), 23
BANDAI_NAMCO_RESEARCH_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 61
BASE (*gnosis.eth.EthereumNetwork attribute*), 61
BASE_GOERLI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
BASE_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
BaseOracle (*class in gnosis.eth.oracles.oracles*), 24
batch_call() (*gnosis.eth.ethereum_client.BatchCallManager method*), 36
batch_call() (*gnosis.eth.ethereum_client.EthereumClient method*), 43
batch_call() (*gnosis.eth.EthereumClient method*), 53
batch_call_custom() (*gnosis.eth.ethereum_client.BatchCallManager method*), 37
batch_call_manager (*gnosis.eth.EthereumClient attribute*), 54
batch_call_same_function() (*gnosis.eth.ethereum_client.BatchCallManager method*), 37
batch_call_same_function() (*gnosis.eth.ethereum_client.EthereumClient method*), 43
batch_call_same_function() (*gnosis.eth.EthereumClient method*), 54
BatchCallManager (*class in gnosis.eth.ethereum_client*), 36
BEAGLE_MESSAGING_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 61
BEAM (*gnosis.eth.EthereumNetwork attribute*), 61
BEAM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
BEANECO_SMARTCHAIN (*gnosis.eth.EthereumNetwork attribute*), 61
BEAR_NETWORK_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 61
BEAR_NETWORK_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 61
BEONE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BERACHAIN_ARTIO (*gnosis.eth.EthereumNetwork attribute*), 62
BERESHEET_BEREEVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BERYLBIT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BEVERLY_HILLS (*gnosis.eth.EthereumNetwork attribute*), 62
BEVM_CANARY (*gnosis.eth.EthereumNetwork attribute*), 62
BIFROST_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BIFROST_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BIGSHORTBETS (*gnosis.eth.EthereumNetwork attribute*), 62
BITCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITCICHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITCICHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITCOIN_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 62
BITCOIN_EVM (*gnosis.eth.EthereumNetwork attribute*), 62
BITFINITY_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITGERT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITICA_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITINDI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITINDI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITKUB_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 62
BITKUB_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITROCK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITROCK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITTEX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITTORRENT_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITTORRENT_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BITYUAN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BLACKFORT_EXCHANGE_NETWORK (*gno-*

sis.eth.EthereumNetwork attribute), 62
BLACKFORT_EXCHANGE_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 62
BLAST_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 62
BLAST_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLG_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLITZ_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLOCKCHAIN_GENESIS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLOCKCHAIN_STATION_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLOCKCHAIN_STATION_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
blockHash (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
blockNumber (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
BlockscoutClient (*class in gnosis.eth.clients*), 13
BlockscoutClient (*class in gnosis.eth.clients.blockscout_client*), 7
BlockscoutClientException, 9, 15
BlockScoutConfigurationProblem, 7, 13
BLOCKTON_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 63
BLOCX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLOXBERG (*gnosis.eth.EthereumNetwork attribute*), 63
BLUCRATES (*gnosis.eth.EthereumNetwork attribute*), 63
BLXQ_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BLXQ_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BMC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BMC_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BNB_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BNB_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_AVAX (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_BNB_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_BNB_MAINNET_OLD (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_BNB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_NETWORK_GOERLI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBA_NETWORK_RINKEBY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBABASE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBABEAM (*gnosis.eth.EthereumNetwork attribute*), 63
BOBAFUJI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOBAOPERA (*gnosis.eth.EthereumNetwork attribute*), 63
BOBAOPERA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BOMB_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 63
BOMB_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 63
BON_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 63
BOSAGORA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BOTANIX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BOTANIX_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
BRC_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BROCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BRONOS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BRONOS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
BSL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
BTACHAIN (*gnosis.eth.EthereumNetwork attribute*), 64
BTC20_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 64
BTCIX_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 64
build() (*gnosis.safe.safe_create2_tx.SafeCreate2TxBuilder method*), 113
build_for_owner() (*gnosis.safe.safe_signature.SafeSignatureApprovedHash class method*), 114
build_multisig_tx() (*gnosis.safe.Safe method*), 121
build_multisig_tx() (*gnosis.safe.safe.Safe method*), 104
build_tx_data() (*gnosis.safe.multi_send.MultiSend method*), 100
build_tx_params() (*gnosis.eth.ethereum_client.EthereumClient static method*), 43
build_tx_params() (*gnosis.eth.EthereumClient static method*), 54
build_url() (*gnosis.eth.clients.blockscout_client.BlockscoutClient method*), 9

build_url() (*gnosis.eth.clients.BlockscoutClient method*), 15
build_url() (*gnosis.eth.clients.etherscan_client.EtherscanClient method*), 11
build_url() (*gnosis.eth.clients.EtherscanClient method*), 17
BULLETIN_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 64

C

calculate_create2_address() (*gnosis.safe.safe_create2_tx.SafeCreate2TxBuilder method*), 113
calculate_pair_address() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 26
calculate_pair_address() (*gnosis.eth.oracles.UniswapV2Oracle method*), 33
calculate_proxy_address() (*gnosis.safe.proxy_factory.ProxyFactory method*), 101
calculate_proxy_address() (*gnosis.safe.ProxyFactory method*), 119
CALL (*gnosis.safe.multi_send.MultiSendOperation attribute*), 101
CALL (*gnosis.safe.SafeOperationEnum attribute*), 125
call() (*gnosis.safe.safe_tx.SafeTx method*), 116
call() (*gnosis.safe.SafeTx method*), 126
CALLISTO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CALLISTO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CALLISTO_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 64
CAMDL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CAMELARK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CAMINO_C_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 64
CANDLE (*gnosis.eth.EthereumNetwork attribute*), 64
CannotCheckEIP1271ContractSignature, 113
CannotEstimateGas, 99, 119
CannotRetrieveSafeInfoException, 99
CANTO (*gnosis.eth.EthereumNetwork attribute*), 64
CANTO_TESNET (*gnosis.eth.EthereumNetwork attribute*), 64
CANTO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CANXIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CARBON_EVM (*gnosis.eth.EthereumNetwork attribute*), 64
CARBON_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CASCADIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CATALYST (*gnosis.eth.EthereumNetwork attribute*), 64
CATECOIN_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CELO_ALFAJORES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CELO_BAKLAVA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 64
CELO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 64
CENNZNET_AZALEA (*gnosis.eth.EthereumNetwork attribute*), 65
CENNZNET_NIKAU (*gnosis.eth.EthereumNetwork attribute*), 65
CENNZNET_RATA (*gnosis.eth.EthereumNetwork attribute*), 65
CENTRIFUGE (*gnosis.eth.EthereumNetwork attribute*), 65
CERIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
chain_id (*gnosis.safe.Safe property*), 121
chain_id (*gnosis.safe.safe.Safe property*), 105
chain_id (*gnosis.safe.safe_tx.SafeTx property*), 116
chain_id (*gnosis.safe.SafeTx property*), 126
CHAIN_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
CHAOS_SKALE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CHEAPETH (*gnosis.eth.EthereumNetwork attribute*), 65
check_funds_for_tx_gas() (*gnosis.safe.Safe method*), 121
check_funds_for_tx_gas() (*gnosis.safe.safe.Safe method*), 105
check_proxy_code() (*gnosis.safe.proxy_factory.ProxyFactory method*), 102
check_proxy_code() (*gnosis.safe.ProxyFactory method*), 119
check_r() (*gnosis.safe.serializers.SafeSignatureSerializer method*), 118
check_s() (*gnosis.safe.serializers.SafeSignatureSerializer method*), 118
check_tx_with_confirmations() (*gnosis.eth.ethereum_client.EthereumClient method*), 44
check_tx_with_confirmations() (*gnosis.eth.EthereumClient method*), 55
check_v() (*gnosis.safe.serializers.SafeSignatureSerializer method*), 118
CHILIZ_SCOVILLE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CIC_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork at-*

tribute), 65
CIC_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOUDTX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOUDTX_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOUDVERSE_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOUDWALK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOUDWALK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLOVER_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CLV_PARACHAIN (*gnosis.eth.EthereumNetwork attribute*), 65
CMP_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
CMP_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
CODEFIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
COINBIT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
COINEX_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
COINEX_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
COLUMBUS_TEST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 65
COMBO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
COMBO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 65
compare_byte_code() (*in module gnosis.eth.utils*), 52
ComposedPriceOracle (*class in gnosis.eth.oracles*), 29
ComposedPriceOracle (*class in gnosis.eth.oracles.oracles*), 24
COMPVERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 65
CONDOR_TEST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 65
CONDRIEU (*gnosis.eth.EthereumNetwork attribute*), 65
CONDUIT_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 65
CONET_HOLESKY (*gnosis.eth.EthereumNetwork attribute*), 66
CONET_SEBOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
CONFLUX_ESPACE (*gnosis.eth.EthereumNetwork attribute*), 66
CONFLUX_ESPACE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
CONNECTORMANAGER (*gnosis.eth.EthereumNetwork attribute*), 66
CONNECTORMANAGER_ROBIN (*gnosis.eth.EthereumNetwork attribute*), 66
CONSTA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
contract (*gnosis.safe.safe_tx.SafeTx property*), 117
contract (*gnosis.safe.SafeTx property*), 126
contract_address (*gnosis.eth.ethereum_client.EthereumTxSent attribute*), 47
contract_address (*gnosis.eth.EthereumTxSent attribute*), 97
CONTRACT_SIGNATURE (*gnosis.safe.safe_signature.SafeSignatureType attribute*), 116
ContractMetadata (*class in gnosis.eth.clients*), 15
ContractMetadata (*class in gnosis.eth.clients.contract_metadata*), 9
CORE_BLOCKCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CORE_BLOCKCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
COSMIC_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 66
CouldNotFinishInitialization, 99
CouldNotPayGasWithEther, 99, 119
CouldNotPayGasWithToken, 99
CowswapOracle (*class in gnosis.eth.oracles*), 29
CRAB_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 66
CreamOracle (*class in gnosis.eth.oracles*), 29
CreamOracle (*class in gnosis.eth.oracles.oracles*), 24
CREATE (*gnosis.safe.SafeOperationEnum attribute*), 126
CREDIT_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 66
CREDIT_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CRONOS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CRONOS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
CROSSBELL (*gnosis.eth.EthereumNetwork attribute*), 66
CRYPTO_EMERGENCY (*gnosis.eth.EthereumNetwork attribute*), 66
CRYPTOCOINPAY (*gnosis.eth.EthereumNetwork attribute*), 66
CRYSTALEUM (*gnosis.eth.EthereumNetwork attribute*), 66
CTEX_SCAN_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 66
CUBE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CUBE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66

current_block_number (*gnosis.eth.ethereum_client.EthereumClient property*), 44
current_block_number (*gnosis.eth.EthereumClient property*), 55
CURVE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CurveOracle (*class in gnosis.eth.oracles*), 30
CurveOracle (*class in gnosis.eth.oracles.oracles*), 24
CYBERDECKNET (*gnosis.eth.EthereumNetwork attribute*), 66
CYBERTRUST (*gnosis.eth.EthereumNetwork attribute*), 66
CYBRIA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 66
CYBRIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66

D

D_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DARWINIA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 66
DARWINIA_PANGOLIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
DARWINIA_PANGORO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 66
data (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
data_length (*gnosis.safe.multi_send.MultiSendTx property*), 101
DATAHOPPER (*gnosis.eth.EthereumNetwork attribute*), 66
DAX_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 67
DBCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEAMCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEAMCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEBANK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEBANK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEBANK_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 67
DEBOUNCE_SUBNET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DECENTRABONE_LAYER1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DECENTRALIZED_WEB_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DECIMAL_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DECIMAL_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
decimals (*gnosis.eth.ethereum_client.Erc20Info attribute*), 37
decode_logs() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
decode_string_or_bytes32() (*in module gnosis.eth.utils*), 52
DEELANCE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
default_error_messages (*gnosis.eth.django.serializers.HexadecimalField attribute*), 22
DEFAULT_ROUTER_ADDRESS (*gnosis.eth.oracles.UniswapV3Oracle attribute*), 34
DEFI_ORACLE_META_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEFI_ORACLE_META_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEFICHAIN_EVM_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEFICHAIN_EVM_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEFIMETACHAIN_CHANGI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEHVO (*gnosis.eth.EthereumNetwork attribute*), 67
DELEGATE_CALL (*gnosis.safe.multi_send.MultiSendOperation attribute*), 101
DELEGATE_CALL (*gnosis.safe.SafeOperationEnum attribute*), 126
deploy_and_initialize_contract() (*gnosis.eth.ethereum_client.EthereumClient method*), 44
deploy_and_initialize_contract() (*gnosis.eth.EthereumClient method*), 55
deploy_contract() (*gnosis.safe.multi_send.MultiSend static method*), 100
deploy_contract() (*gnosis.safe.proxy_factory.ProxyFactory class method*), 102
deploy_contract() (*gnosis.safe.ProxyFactory class method*), 119
deploy_contract() (*gnosis.safe.Safe class method*), 122
deploy_contract() (*gnosis.safe.safe.Safe class method*), 105
deploy_contract() (*gnosis.safe.safe.SafeV001 static method*), 109
deploy_contract() (*gnosis.safe.safe.SafeV100 static method*), 110
deploy_proxy_contract() (*gnosis.safe.proxy_factory.ProxyFactory method*), 102
deploy_proxy_contract() (*gnosis.safe.proxy_factory.ProxyFactoryV141*

method), 103
deploy_proxy_contract() (*gnosis.safe.ProxyFactory method*), 120
deploy_proxy_contract_with_nonce() (*gnosis.safe.proxy_factory.ProxyFactory method*), 102
deploy_proxy_contract_with_nonce() (*gnosis.safe.ProxyFactory method*), 120
DEPRECATED_CHI (*gnosis.eth.EthereumNetwork attribute*), 67
DEXALOT_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEXALOT_SUBNET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEXILLA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 67
DEXIT_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 67
DFK_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 67
DFK_CHAIN_TEST (*gnosis.eth.EthereumNetwork attribute*), 67
DIGIT_SOUL_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 67
DIGIT_SOUL_SMART_CHAIN_2 (*gnosis.eth.EthereumNetwork attribute*), 67
DIODE_PRENET (*gnosis.eth.EthereumNetwork attribute*), 67
DIODE_TESTNET_STAGING (*gnosis.eth.EthereumNetwork attribute*), 67
DM2_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 67
DODAO (*gnosis.eth.EthereumNetwork attribute*), 68
DOGCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOGCOIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOGECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOGECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOGELAYER_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOGETHER_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOKEN_SUPER_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
domain_separator (*gnosis.safe.Safe property*), 122
domain_separator (*gnosis.safe.safe.Safe property*), 105
DOS_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 68
DOS_FUJI_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOS_TESNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOUBLE_A_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DOUBLE_A_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
DPU_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 68
DRAC_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 68
DRACONES_FINANCIAL_SERVICES (*gnosis.eth.EthereumNetwork attribute*), 68
DRAGONFLY_MAINNET_HEXAPOD (*gnosis.eth.EthereumNetwork attribute*), 68
DUBXCOIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 68
DUBXCOIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
dummy_w3 (*gnosis.safe.multi_send.MultiSend attribute*), 101
DXCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DXCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
DYNO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
DYNO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68

E

E_DOLLAR (*gnosis.eth.EthereumNetwork attribute*), 70
EBRO_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 68
ECOBALL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
ECOBALL_TESTNET_ESPUMA (*gnosis.eth.EthereumNetwork attribute*), 68
ECREDITS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
ECREDITS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
ECROX_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 68
EDEXA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 68
EDGEWARE_EDGEEVM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
EGONCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
EGONCOIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
EIP1271_MAGIC_VALUE (*gnosis.safe.safe_signature.SafeSignatureContract attribute*), 114
EIP1271_MAGIC_VALUE_UPDATED (*gnosis.safe.safe_signature.SafeSignatureContract attribute*), 115
eip712_structured_data (*gnosis.safe.safe_tx.SafeTx property*), 117

eip712_structured_data (*gnosis.safe.SafeTx property*), 126
EKTA (*gnosis.eth.EthereumNetwork attribute*), 69
ELA_DID_SIDECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELA_DID_SIDECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELASTOS_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 69
ELASTOS_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELEANOR (*gnosis.eth.EthereumNetwork attribute*), 69
ELECTRONEUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELECTRONEUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELIBERTY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELIBERTY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELLA_THE_HEART (*gnosis.eth.EthereumNetwork attribute*), 69
ELLAISM (*gnosis.eth.EthereumNetwork attribute*), 69
ELUVIO_CONTENT_FABRIC (*gnosis.eth.EthereumNetwork attribute*), 69
ELYSIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ELYSIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
EMPIRE_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 69
encoded_data (*gnosis.safe.multi_send.MultiSendTx property*), 101
ENDURANCE_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENERGI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENERGI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENERGY_WEB_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 69
ENERGY_WEB_VOLTA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENGRAM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENNOTHEM_MAINNET_PROTEROZOIC (*gnosis.eth.EthereumNetwork attribute*), 69
ENNOTHEM_TESTNET_PIONEER (*gnosis.eth.EthereumNetwork attribute*), 69
ENTERCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENULS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 69
ENULS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 69
EnzymeOracle (*class in gnosis.eth.oracles*), 30
EnzymeOracle (*class in gnosis.eth.oracles.oracles*), 25
EOA (*gnosis.safe.safe_signature.SafeSignatureType attribute*), 116
EOS_EVM_LEGACY (*gnosis.eth.EthereumNetwork attribute*), 69
EOS_EVM_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 70
EOS_EVM_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
ERASWAP_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
erc20 (*gnosis.eth.EthereumClient attribute*), 55
Erc20Info (*class in gnosis.eth.ethereum_client*), 37
Erc20Manager (*class in gnosis.eth.ethereum_client*), 37
erc721 (*gnosis.eth.EthereumClient attribute*), 55
Erc721Info (*class in gnosis.eth.ethereum_client*), 41
Erc721Manager (*class in gnosis.eth.ethereum_client*), 41
estimate_data_gas() (*gnosis.eth.ethereum_client.EthereumClient static method*), 44
estimate_data_gas() (*gnosis.eth.EthereumClient static method*), 55
estimate_fee_eip1559() (*gnosis.eth.ethereum_client.EthereumClient method*), 44
estimate_fee_eip1559() (*gnosis.eth.EthereumClient method*), 55
estimate_gas() (*gnosis.eth.ethereum_client.EthereumClient method*), 44
estimate_gas() (*gnosis.eth.EthereumClient method*), 55
estimate_tx_base_gas() (*gnosis.safe.Safe method*), 122
estimate_tx_base_gas() (*gnosis.safe.safe.Safe method*), 105
estimate_tx_gas() (*gnosis.safe.Safe method*), 122
estimate_tx_gas() (*gnosis.safe.safe.Safe method*), 106
estimate_tx_gas_by_trying() (*gnosis.safe.Safe method*), 123
estimate_tx_gas_by_trying() (*gnosis.safe.safe.Safe method*), 106
estimate_tx_gas_with_safe() (*gnosis.safe.Safe method*), 123
estimate_tx_gas_with_safe() (*gnosis.safe.safe.Safe method*), 106
estimate_tx_gas_with_safe() (*gnosis.safe.safe.SafeV141 method*), 111
estimate_tx_gas_with_web3() (*gnosis.safe.Safe method*), 123
estimate_tx_gas_with_web3() (*gnosis.safe.safe.Safe*

method), 106
ETH_SIGN (*gnosis.safe.safe_signature.SafeSignatureType attribute*), 116
ETH_TOKEN_ADDRESS (*gnosis.eth.oracles.KyberOracle attribute*), 30
ETHEREUM_CLASSIC (*gnosis.eth.EthereumNetwork attribute*), 70
ETHEREUM_FAIR (*gnosis.eth.EthereumNetwork attribute*), 70
ethereum_node_url (*gnosis.eth.EthereumClient attribute*), 56
EthereumAddressField (*class in gnosis.eth.django.serializers*), 22
EthereumClient (*class in gnosis.eth*), 53
EthereumClient (*class in gnosis.eth.ethereum_client*), 42
EthereumClientManager (*class in gnosis.eth.ethereum_client*), 47
EthereumClientProvider (*class in gnosis.eth*), 58
EthereumClientProvider (*class in gnosis.eth.ethereum_client*), 47
EthereumNetwork (*class in gnosis.eth*), 58
EthereumNetworkNotSupported, 97
EthereumTxSent (*class in gnosis.eth*), 97
EthereumTxSent (*class in gnosis.eth.ethereum_client*), 47
ETHERGEM (*gnosis.eth.EthereumNetwork attribute*), 70
ETHERINC (*gnosis.eth.EthereumNetwork attribute*), 70
ETHERLINK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
ETHERLITE_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 70
EtherscanClient (*class in gnosis.eth.clients*), 15
EtherscanClient (*class in gnosis.eth.clients.etherscan_client*), 9
EtherscanClientConfigurationProblem, 12, 18
EtherscanClientException, 12, 18
EtherscanRateLimitError, 12, 18
ETHERSOCIAL_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 70
ETHO_PROTOCOL (*gnosis.eth.EthereumNetwork attribute*), 70
ETICA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
ETND_CHAIN_MAINNETS (*gnosis.eth.EthereumNetwork attribute*), 70
EURUS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EURUS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVANESCO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVANESCO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVMOS (*gnosis.eth.EthereumNetwork attribute*), 70
EVMOS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVOKE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVOKE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
EVRICE_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 70
EXCELON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EXCOINCIAL_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EXCOINCIAL_CHAIN_VOLTA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70
execute() (*gnosis.safe.safe_tx.SafeTx method*), 117
execute() (*gnosis.safe.SafeTx method*), 126
EXOSAMA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 70
EXPANSE_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 70
export_signature() (*gnosis.safe.safe_signature.SafeSignature method*), 113
export_signature() (*gnosis.safe.safe_signature.SafeSignatureContract method*), 115
export_signatures() (*gnosis.safe.safe_signature.SafeSignature class method*), 113
EXZO_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EZCHAIN_C_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 70
EZCHAIN_C_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 70

F

F_XCORE_MAINNET_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
factory (*gnosis.eth.oracles.oracles.UniswapV2Oracle property*), 26
factory (*gnosis.eth.oracles.UniswapV2Oracle property*), 33
FACTORY_127_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
factory_address (*gnosis.eth.oracles.oracles.UniswapV2Oracle property*), 26
factory_address (*gnosis.eth.oracles.UniswapV2Oracle property*), 33
fallback_handler (*gnosis.safe.safe.SafeInfo attribute*), 109

fallback_handler (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
FALLBACK_HANDLER_STORAGE_SLOT (*gnosis.safe.Safe attribute*), 121
FALLBACK_HANDLER_STORAGE_SLOT (*gnosis.safe.safe.Safe attribute*), 104
FANTASIA_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FANTOM_OPERA (*gnosis.eth.EthereumNetwork attribute*), 71
FANTOM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FAST (*gnosis.eth.ethereum_client.TxSpeed attribute*), 50
FAST (*gnosis.eth.TxSpeed attribute*), 98
fast_bytes_to_checksum_address() (*in module gnosis.eth.utils*), 52
fast_is_checksum_address() (*in module gnosis.eth.utils*), 52
fast_keccak() (*in module gnosis.eth.utils*), 52
fast_keccak_hex() (*in module gnosis.eth.utils*), 52
fast_to_checksum_address() (*in module gnosis.eth.utils*), 52
FASTEST (*gnosis.eth.ethereum_client.TxSpeed attribute*), 50
FASTEST (*gnosis.eth.TxSpeed attribute*), 98
FASTEX_CHAIN_BAHAMUT_OASIS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FASTEX_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FERRUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FIBONACCI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_BUTTERFLY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_CALIBRATION_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_HYPERSPACE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_LOCAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FILECOIN_WALLABY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
filter_out_errored_traces() (*gnosis.eth.ethereum_client.TracingManager method*), 48
FINDORA_FORGE (*gnosis.eth.EthereumNetwork attribute*), 71
FINDORA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FINDORA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FIRECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FIRECHAIN_MAINNET_OLD (*gnosis.eth.EthereumNetwork attribute*), 71
FIRECHAIN_ZKEVM (*gnosis.eth.EthereumNetwork attribute*), 71
FIRECHAIN_ZKEVM_GHOSTRIDER (*gnosis.eth.EthereumNetwork attribute*), 71
FIRENZE_TEST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 71
fixed_creation_cost (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
FLACHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FLANA (*gnosis.eth.EthereumNetwork attribute*), 71
FLANA_MIXNET (*gnosis.eth.EthereumNetwork attribute*), 71
FLANA_PRIVNET (*gnosis.eth.EthereumNetwork attribute*), 71
FLANA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 71
FLARE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 71
FLARE_TESTNET_COSTON (*gnosis.eth.EthereumNetwork attribute*), 71
FLARE_TESTNET_COSTON2 (*gnosis.eth.EthereumNetwork attribute*), 71
FNCY (*gnosis.eth.EthereumNetwork attribute*), 71
FNCY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
FOUNDRY_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
FOX_TESTNET_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
FRAME_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
FRAXTAL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
FRAXTAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
FREIGHT_TRUST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
FRENCHAIN (*gnosis.eth.EthereumNetwork attribute*), 72
from_bytes() (*gnosis.safe.multi_send.MultiSend class method*), 101
from_bytes() (*gnosis.safe.multi_send.MultiSendTx class method*), 101
from_transaction_data() (*gnosis.safe.multi_send.MultiSend class method*), 101
from_v() (*gnosis.safe.safe_signature.SafeSignatureType static method*), 116

from_values() (*gnosis.safe.safe_signature.SafeSignatureContract class method*), 115
FromAddressNotFound, 98
FRONTIER_OF_DREAMS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
FUSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
FUSE_SPARKNET (*gnosis.eth.EthereumNetwork attribute*), 72
FUSION_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
FUSION_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72

G

G8CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
G8CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
GANACHE (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_STAGE0 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_STAGE1 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_STAGE2 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_STAGE3 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_TESTNET_STAGE0 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_TESTNET_STAGE1 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_TESTNET_STAGE2 (*gnosis.eth.EthereumNetwork attribute*), 72
GARIZON_TESTNET_STAGE3 (*gnosis.eth.EthereumNetwork attribute*), 72
gas (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
gas_price (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
GasLimitExceeded, 98
GATECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
GATECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 72
GATHER_DEVNET_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
GATHER_MAINNET_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
GATHER_TESTNET_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 72
GAUSS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 72
GEAR_ZERO_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GEAR_ZERO_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GENECHAIN (*gnosis.eth.EthereumNetwork attribute*), 73
generate_contract_fn() (*in module gnosis.eth.contracts*), 19
GENESIS_COIN (*gnosis.eth.EthereumNetwork attribute*), 73
GENESIS_L1 (*gnosis.eth.EthereumNetwork attribute*), 73
GENESIS_L1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GENESYS_CODE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GENESYS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GESO_VERSE (*gnosis.eth.EthereumNetwork attribute*), 73
GESOTEN_VERSE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
get_balance() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_balance() (*gnosis.eth.ethereum_client.Erc721Manager method*), 42
get_balance() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_balance() (*gnosis.eth.EthereumClient method*), 56
get_balances() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_balances() (*gnosis.eth.ethereum_client.Erc721Manager method*), 42
get_block() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_block() (*gnosis.eth.EthereumClient method*), 56
get_blocks() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_blocks() (*gnosis.eth.EthereumClient method*), 56
get_chain_id() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_chain_id() (*gnosis.eth.EthereumClient method*), 56
get_chains() (*gnosis.eth.clients.sourcify_client.SourcifyClient method*), 12
get_chains() (*gnosis.eth.clients.SourcifyClient method*), 18
get_client_version() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_client_version() (*gnosis.eth.EthereumClient method*), 56
get_compatibility_fallback_handler_contract() (*in module gnosis.eth.contracts*), 19

get_compatibility_fallback_handler_V1_3_0_contract() (*in module gnosis.eth.contracts*), 19
get_compatibility_fallback_handler_V1_4_1_contract() (*in module gnosis.eth.contracts*), 19
get_contract() (*gnosis.safe.multi_send.MultiSend method*), 101
get_contract_abi() (*gnosis.eth.clients.etherscan_client.EtherscanClient method*), 11
get_contract_abi() (*gnosis.eth.clients.EtherscanClient method*), 17
get_contract_fn() (*gnosis.safe.proxy_factory.ProxyFactoryV100 method*), 103
get_contract_fn() (*gnosis.safe.proxy_factory.ProxyFactoryV111 method*), 103
get_contract_fn() (*gnosis.safe.proxy_factory.ProxyFactoryV130 method*), 103
get_contract_fn() (*gnosis.safe.proxy_factory.ProxyFactoryV141 method*), 104
get_contract_fn() (*gnosis.safe.safe.SafeV001 method*), 110
get_contract_fn() (*gnosis.safe.safe.SafeV100 method*), 110
get_contract_fn() (*gnosis.safe.safe.SafeV111 method*), 110
get_contract_fn() (*gnosis.safe.safe.SafeV120 method*), 110
get_contract_fn() (*gnosis.safe.safe.SafeV130 method*), 111
get_contract_fn() (*gnosis.safe.safe.SafeV141 method*), 111
get_contract_metadata() (*gnosis.eth.clients.blockscout_client.BlockscoutClient method*), 9
get_contract_metadata() (*gnosis.eth.clients.BlockscoutClient method*), 15
get_contract_metadata() (*gnosis.eth.clients.etherscan_client.EtherscanClient method*), 11
get_contract_metadata() (*gnosis.eth.clients.EtherscanClient method*), 17
get_contract_metadata() (*gnosis.eth.clients.sourcify_client.SourcifyClient method*), 12
get_contract_metadata() (*gnosis.eth.clients.SourcifyClient method*), 18
get_contract_source_code() (*gnosis.eth.clients.etherscan_client.EtherscanClient method*), 11
get_contract_source_code() (*gnosis.eth.clients.EtherscanClient method*), 17
get_cpk_factory_contract() (*in module gnosis.eth.contracts*), 19
get_decimals() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_delegate_constructor_proxy_contract() (*in module gnosis.eth.contracts*), 19
get_deploy_function() (*gnosis.safe.proxy_factory.ProxyFactory method*), 103
get_deploy_function() (*gnosis.safe.proxy_factory.ProxyFactoryV141 method*), 104
get_deploy_function() (*gnosis.safe.ProxyFactory method*), 120
get_empty_tx_params() (*in module gnosis.eth.utils*), 52
get_erc1155_contract() (*in module gnosis.eth.contracts*), 19
get_erc20_contract() (*in module gnosis.eth.contracts*), 19
get_erc721_contract() (*in module gnosis.eth.contracts*), 20
get_eth_address_with_invalid_checksum() (*in module gnosis.eth.utils*), 53
get_example_erc20_contract() (*in module gnosis.eth.contracts*), 20
get_factory() (*gnosis.eth.oracles.UniswapV3Oracle method*), 34
get_fields() (*gnosis.eth.django.serializers.TransactionResponseSerialize method*), 22
get_fields() (*gnosis.eth.django.serializers.TransactionSerializer method*), 22
get_info() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_info() (*gnosis.eth.ethereum_client.Erc721Manager method*), 42
get_kyber_network_proxy_contract() (*in module gnosis.eth.contracts*), 20
get_message_hash() (*gnosis.safe.Safe method*), 123
get_message_hash() (*gnosis.safe.safe.Safe method*), 107
get_multi_send_contract() (*in module gnosis.eth.contracts*), 20
get_multicall_v3_contract() (*in module gnosis.eth.contracts*), 20
get_name() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_network() (*gnosis.eth.ethereum_client.EthereumClient*

method), 45
get_network() (*gnosis.eth.EthereumClient method*), 56
get_next_traces() (*gnosis.eth.ethereum_client.TracingManager method*), 48
get_nonce_for_account() (*gnosis.eth.ethereum_client.EthereumClient method*), 45
get_nonce_for_account() (*gnosis.eth.EthereumClient method*), 56
get_owners() (*gnosis.eth.ethereum_client.Erc721Manager method*), 42
get_pair_address() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 26
get_pair_address() (*gnosis.eth.oracles.UniswapV2Oracle method*), 33
get_paying_proxy_contract() (*in module gnosis.eth.contracts*), 20
get_paying_proxy_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_pool_address() (*gnosis.eth.oracles.UniswapV3Oracle method*), 34
get_pool_token_price() (*gnosis.eth.oracles.BalancerOracle method*), 29
get_pool_token_price() (*gnosis.eth.oracles.MooniswapOracle method*), 30
get_pool_token_price() (*gnosis.eth.oracles.oracles.BalancerOracle method*), 23
get_pool_token_price() (*gnosis.eth.oracles.oracles.MooniswapOracle method*), 25
get_pool_token_price() (*gnosis.eth.oracles.oracles.PricePoolOracle method*), 25
get_pool_token_price() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 27
get_pool_token_price() (*gnosis.eth.oracles.PricePoolOracle method*), 31
get_pool_token_price() (*gnosis.eth.oracles.UniswapV2Oracle method*), 33
get_previous_trace() (*gnosis.eth.ethereum_client.TracingManager method*), 48
get_price() (*gnosis.eth.oracles.AaveOracle method*), 28
get_price() (*gnosis.eth.oracles.CowswapOracle method*), 29
get_price() (*gnosis.eth.oracles.CreamOracle method*), 29
get_price() (*gnosis.eth.oracles.KyberOracle method*), 30
get_price() (*gnosis.eth.oracles.oracles.AaveOracle method*), 23
get_price() (*gnosis.eth.oracles.oracles.CreamOracle method*), 24
get_price() (*gnosis.eth.oracles.oracles.PriceOracle method*), 25
get_price() (*gnosis.eth.oracles.oracles.UniswapOracle method*), 26
get_price() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 27
get_price() (*gnosis.eth.oracles.PriceOracle method*), 31
get_price() (*gnosis.eth.oracles.SuperfluidOracle method*), 31
get_price() (*gnosis.eth.oracles.UniswapOracle method*), 32
get_price() (*gnosis.eth.oracles.UniswapV2Oracle method*), 33
get_price() (*gnosis.eth.oracles.UniswapV3Oracle method*), 34
get_price_without_exception() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 27
get_price_without_exception() (*gnosis.eth.oracles.UniswapV2Oracle method*), 33
get_proxy_1_0_0_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_proxy_1_1_1_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_proxy_1_1_1_mainnet_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_proxy_1_3_0_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_proxy_1_4_1_deployed_bytecode() (*in module gnosis.eth.contracts*), 20
get_proxy_contract() (*in module gnosis.eth.contracts*), 20
get_proxy_creation_code() (*gnosis.safe.proxy_factory.ProxyFactory method*), 103
get_proxy_creation_code() (*gnosis.safe.ProxyFactory method*), 120
get_proxy_factory_contract() (*in module gnosis.eth.contracts*), 20
get_proxy_factory_V1_0_0_contract() (*in module gnosis.eth.contracts*), 20
get_proxy_factory_V1_1_1_contract() (*in module*

gnosis.eth.contracts), 20
get_proxy_factory_V1_3_0_contract() (*in module gnosis.eth.contracts*), 20
get_proxy_factory_V1_4_1_contract() (*in module gnosis.eth.contracts*), 20
get_proxy_runtime_code() (*gnosis.safe.proxy_factory.ProxyFactory method*), 103
get_proxy_runtime_code() (*gnosis.safe.proxy_factory.ProxyFactoryV141 method*), 104
get_proxy_runtime_code() (*gnosis.safe.ProxyFactory method*), 120
get_reserves() (*gnosis.eth.oracles.oracles.UniswapV2Oracle method*), 27
get_reserves() (*gnosis.eth.oracles.UniswapV2Oracle method*), 34
get_safe_contract() (*in module gnosis.eth.contracts*), 21
get_safe_V0_0_1_contract() (*in module gnosis.eth.contracts*), 20
get_safe_V1_0_0_contract() (*in module gnosis.eth.contracts*), 21
get_safe_V1_1_1_contract() (*in module gnosis.eth.contracts*), 21
get_safe_V1_3_0_contract() (*in module gnosis.eth.contracts*), 21
get_safe_V1_4_1_contract() (*in module gnosis.eth.contracts*), 21
get_sign_message_lib_contract() (*in module gnosis.eth.contracts*), 21
get_signing_address() (*in module gnosis.safe.signatures*), 118
get_simulate_tx_accessor_V1_4_1_contract() (*in module gnosis.eth.contracts*), 21
get_symbol() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_token_uris() (*gnosis.eth.ethereum_client.Erc721Manager method*), 42
get_total_transfer_history() (*gnosis.eth.ethereum_client.Erc20Manager method*), 38
get_transaction() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
get_transaction() (*gnosis.eth.EthereumClient method*), 56
get_transaction_receipt() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
get_transaction_receipt() (*gnosis.eth.EthereumClient method*), 56
get_transaction_receipts() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
get_transaction_receipts() (*gnosis.eth.EthereumClient method*), 57
get_transactions() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
get_transactions() (*gnosis.eth.EthereumClient method*), 57
get_transfer_history() (*gnosis.eth.ethereum_client.Erc20Manager method*), 40
get_underlying_tokens() (*gnosis.eth.oracles.ComposedPriceOracle method*), 29
get_underlying_tokens() (*gnosis.eth.oracles.CurveOracle method*), 30
get_underlying_tokens() (*gnosis.eth.oracles.oracles.ComposedPriceOracle method*), 24
get_underlying_tokens() (*gnosis.eth.oracles.oracles.CurveOracle method*), 24
get_underlying_tokens() (*gnosis.eth.oracles.oracles.YearnOracle method*), 27
get_underlying_tokens() (*gnosis.eth.oracles.oracles.ZerionComposedOracle method*), 28
get_underlying_tokens() (*gnosis.eth.oracles.YearnOracle method*), 35
get_underlying_tokens() (*gnosis.eth.oracles.ZerionComposedOracle method*), 36
get_uniswap_exchange() (*gnosis.eth.oracles.oracles.UniswapOracle method*), 26
get_uniswap_exchange() (*gnosis.eth.oracles.UniswapOracle method*), 32
get_uniswap_exchange_contract() (*in module gnosis.eth.contracts*), 21
get_uniswap_factory_contract() (*in module gnosis.eth.contracts*), 21
get_uniswap_v2_factory_contract() (*in module gnosis.eth.contracts*), 21
get_uniswap_v2_pair_contract() (*in module gnosis.eth.contracts*), 21
get_uniswap_v2_router_contract() (*in module gnosis.eth.contracts*), 21
get_version() (*gnosis.safe.Safe method*), 123
get_version() (*gnosis.safe.safe.Safe method*), 107
get_version() (*gnosis.safe.safe.SafeV001 method*), 110

get_version() (*gnosis.safe.safe.SafeV100 method*), 110
get_version() (*gnosis.safe.safe.SafeV111 method*), 110
get_version() (*gnosis.safe.safe.SafeV120 method*), 110
get_version() (*gnosis.safe.safe.SafeV130 method*), 111
get_version() (*gnosis.safe.safe.SafeV141 method*), 111
GIANT_MAMMOTH_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GIL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GITSHOCK_CARTENZ_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GLOBEL_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 73
gnosis
    module, 127
GNOSIS (*gnosis.eth.EthereumNetwork attribute*), 73
gnosis.eth
    module, 53
gnosis.eth.clients
    module, 13
gnosis.eth.clients.blockscout_client
    module, 7
gnosis.eth.clients.contract_metadata
    module, 9
gnosis.eth.clients.etherscan_client
    module, 9
gnosis.eth.clients.sourcify_client
    module, 12
gnosis.eth.constants
    module, 36
gnosis.eth.contracts
    module, 18
gnosis.eth.django
    module, 23
gnosis.eth.django.serializers
    module, 22
gnosis.eth.django.validators
    module, 23
gnosis.eth.ethereum_client
    module, 36
gnosis.eth.oracles
    module, 28
gnosis.eth.oracles.abis
    module, 23
gnosis.eth.oracles.abis.aave_abis
    module, 23
gnosis.eth.oracles.abis.balancer_abis
    module, 23
gnosis.eth.oracles.abis.curve_abis
    module, 23
gnosis.eth.oracles.abis.mooniswap_abis
    module, 23
gnosis.eth.oracles.abis.yearn_abis
    module, 23
gnosis.eth.oracles.oracles
    module, 23
gnosis.eth.typing
    module, 51
gnosis.eth.utils
    module, 52
gnosis.safe
    module, 119
gnosis.safe.exceptions
    module, 99
gnosis.safe.multi_send
    module, 100
gnosis.safe.proxy_factory
    module, 101
gnosis.safe.safe
    module, 104
gnosis.safe.safe_create2_tx
    module, 112
gnosis.safe.safe_signature
    module, 113
gnosis.safe.safe_tx
    module, 116
gnosis.safe.serializers
    module, 118
gnosis.safe.signatures
    module, 118
GNOSIS_CHIADO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GOCHAIN (*gnosis.eth.EthereumNetwork attribute*), 73
GOCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GODWOKEN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GODWOKEN_TESTNET_V1 (*gnosis.eth.EthereumNetwork attribute*), 73
GOERLI (*gnosis.eth.EthereumNetwork attribute*), 73
GOLD_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GOLD_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GOLDXCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GOLDXCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GON_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 73
GOODDATA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GOODDATA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 73
GRAPHLINQ_BLOCKCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GROK_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73

GTON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 73
GTON_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
GUAPCOINX (*gnosis.eth.EthereumNetwork attribute*), 74
guard (*gnosis.safe.safe.SafeInfo attribute*), 109
GUARD_STORAGE_SLOT (*gnosis.safe.Safe attribute*), 121
GUARD_STORAGE_SLOT (*gnosis.safe.safe.Safe attribute*), 104

H

HAIC (*gnosis.eth.EthereumNetwork attribute*), 74
HALO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAMMER_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAPCHAIN (*gnosis.eth.EthereumNetwork attribute*), 74
HAPCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAQQ_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAQQ_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 74
HARADEV_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_DEVNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_DEVNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_MAINNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_MAINNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_MAINNET_SHARD_2 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_MAINNET_SHARD_3 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_TESTNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 74
HARMONY_TESTNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 74
HASHBIT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HashHasNotBeenApproved, 99
HASHKEY_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAVEN1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAYMO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HAZLOR_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HEDERA_LOCALNET (*gnosis.eth.EthereumNetwork attribute*), 74
HEDERA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HEDERA_PREVIEWNET (*gnosis.eth.EthereumNetwork attribute*), 74
HEDERA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HELA_OFFICIAL_RUNTIME_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HELA_OFFICIAL_RUNTIME_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 74
HELP_THE_HOMELESS (*gnosis.eth.EthereumNetwork attribute*), 74
HERTZ_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 74
HexadecimalField (*class in gnosis.eth.django.serializers*), 22
HIGH_PERFORMANCE_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 75
HIGHBURY (*gnosis.eth.EthereumNetwork attribute*), 75
HIKA_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HOKUM (*gnosis.eth.EthereumNetwork attribute*), 75
HOKUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HOLESKY (*gnosis.eth.EthereumNetwork attribute*), 75
HOME_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HOO_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 75
HOO_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HORIZEN_EON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HORIZEN_GOBI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HTMLCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HTTP_HEADERS (*gnosis.eth.clients.etherscan_client.EtherscanClient attribute*), 9
HTTP_HEADERS (*gnosis.eth.clients.EtherscanClient attribute*), 15
HUMAN_PROTOCOL (*gnosis.eth.EthereumNetwork attribute*), 75
HUMANODE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HUMANODE_TESTNET_5_ISRAFEL (*gnosis.eth.EthereumNetwork attribute*), 75
HUMANS_AI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HUMANS_AI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HUOBI_ECO_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HUOBI_ECO_CHAIN_TESTNET (*gno-*

sis.eth.EthereumNetwork attribute), 75
HYBRID_CHAIN_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
HYBRID_CHAIN_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HYPERONCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
HYPRA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75

I

ICHAIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 75
ICPLAZA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
IDCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 75
IEXEC_SIDECHAIN (*gnosis.eth.EthereumNetwork attribute*), 75
IMMU3_EVM (*gnosis.eth.EthereumNetwork attribute*), 75
IMMUTABLE_ZKEVM (*gnosis.eth.EthereumNetwork attribute*), 75
IMMUTABLE_ZKEVM_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 75
IMMUTABLE_ZKEVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 75
IMPERIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
IMPERIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
IMVERSED_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
IMVERSED_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
InsufficientFunds, 98
InvalidChecksumAddress, 99, 119
InvalidContractSignatureLocation, 99
InvalidERC20Info, 98
InvalidERC20Token, 112
InvalidERC721Info, 98
InvalidInternalTx, 99, 119
InvalidMultisigTx, 99, 119
InvalidNonce, 98
InvalidOwnerProvided, 99
InvalidPaymentToken, 99, 119
InvalidSignaturesProvided, 99, 119
IOLITE (*gnosis.eth.EthereumNetwork attribute*), 76
IORA_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 76
IOTEX_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
IOTEX_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
IPOS_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 76
IRISHUB (*gnosis.eth.EthereumNetwork attribute*), 76
IRISHUB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
is_available() (*gnosis.eth.oracles.AaveOracle class method*), 28
is_available() (*gnosis.eth.oracles.BalancerOracle class method*), 29
is_available() (*gnosis.eth.oracles.CowswapOracle class method*), 29
is_available() (*gnosis.eth.oracles.CreamOracle class method*), 29
is_available() (*gnosis.eth.oracles.KyberOracle class method*), 30
is_available() (*gnosis.eth.oracles.oracles.AaveOracle class method*), 23
is_available() (*gnosis.eth.oracles.oracles.BalancerOracle class method*), 24
is_available() (*gnosis.eth.oracles.oracles.BaseOracle class method*), 24
is_available() (*gnosis.eth.oracles.oracles.CreamOracle class method*), 24
is_available() (*gnosis.eth.oracles.oracles.UniswapOracle class method*), 26
is_available() (*gnosis.eth.oracles.oracles.UniswapV2Oracle class method*), 27
is_available() (*gnosis.eth.oracles.oracles.YearnOracle class method*), 28
is_available() (*gnosis.eth.oracles.oracles.ZerionComposedOracle class method*), 28
is_available() (*gnosis.eth.oracles.SuperfluidOracle class method*), 31
is_available() (*gnosis.eth.oracles.UniswapOracle class method*), 32
is_available() (*gnosis.eth.oracles.UniswapV2Oracle class method*), 34
is_available() (*gnosis.eth.oracles.UniswapV3Oracle class method*), 35
is_available() (*gnosis.eth.oracles.YearnOracle class method*), 35
is_available() (*gnosis.eth.oracles.ZerionComposedOracle class method*), 36
is_chain_supported() (*gnosis.eth.clients.sourcify_client.SourcifyClient method*), 12
is_chain_supported() (*gno-*

sis.eth.clients.SourcifyClient method), 18
is_contract() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
is_contract() (*gnosis.eth.EthereumClient method*), 57
is_eip1559_supported() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
is_eip1559_supported() (*gnosis.eth.EthereumClient method*), 57
is_valid() (*gnosis.safe.safe_signature.SafeSignature method*), 113
is_valid() (*gnosis.safe.safe_signature.SafeSignatureApprovedHash method*), 114
is_valid() (*gnosis.safe.safe_signature.SafeSignatureContract method*), 115
is_valid() (*gnosis.safe.safe_signature.SafeSignatureEOA method*), 115
is_valid() (*gnosis.safe.safe_signature.SafeSignatureEthSign method*), 115
IVAR_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
IVAR_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76

J

J2O_TARO (*gnosis.eth.EthereumNetwork attribute*), 76
JANUS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
JAPAN_OPEN_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
JAPAN_OPEN_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
JELLIE (*gnosis.eth.EthereumNetwork attribute*), 76
JFIN_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 76
JIBCHAIN_L1 (*gnosis.eth.EthereumNetwork attribute*), 76
JOSEON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
JOULEVERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
JOYS_DIGITAL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
JOYS_DIGITAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
JUNCACHAIN (*gnosis.eth.EthereumNetwork attribute*), 76
JUNCACHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76

K

K_LAOS (*gnosis.eth.EthereumNetwork attribute*), 77
KAIBA_LIGHTNING_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
KALAR_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 76
KALICHAIN (*gnosis.eth.EthereumNetwork attribute*), 76
KALICHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 76
KALYCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 76
KALYCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KANAZAWA (*gnosis.eth.EthereumNetwork attribute*), 77
KARDIACHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 77
KARURA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 77
KARURA_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KAVA (*gnosis.eth.EthereumNetwork attribute*), 77
KAVA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KCC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 77
KCC_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KEKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 77
KEKCHAIN_KEKTEST (*gnosis.eth.EthereumNetwork attribute*), 77
KERLEANO (*gnosis.eth.EthereumNetwork attribute*), 77
KILN (*gnosis.eth.EthereumNetwork attribute*), 77
KINTO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KINTSUGI (*gnosis.eth.EthereumNetwork attribute*), 77
KIWI_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 77
KLAYTN_MAINNET_CYPRESS (*gnosis.eth.EthereumNetwork attribute*), 77
KLAYTN_TESTNET_BAOBAB (*gnosis.eth.EthereumNetwork attribute*), 77
KLYNTAR (*gnosis.eth.EthereumNetwork attribute*), 77
KORTHO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 77
KORTHOTEST (*gnosis.eth.EthereumNetwork attribute*), 77
KOTTI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
KREST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 77
KROMA (*gnosis.eth.EthereumNetwork attribute*), 77
KROMA_SEPOLIA (*gnosis.eth.EthereumNetwork attribute*), 77
kyber_network_proxy_address (*gnosis.eth.oracles.KyberOracle property*), 30
kyber_network_proxy_contract (*gnosis.eth.oracles.KyberOracle property*), 30
KyberOracle (*class in gnosis.eth.oracles*), 30
KYOTO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77

L

LACHAIN (*gnosis.eth.EthereumNetwork attribute*), 77
LACHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 77
LACHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
LAMBDA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 77
LAOS_ARRAKIS (*gnosis.eth.EthereumNetwork attribute*), 78
LATAM_BLOCKCHAIN_RESIL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LATEST_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LATESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIGHTLINK_PEGASUS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIGHTLINK_PHOENIX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIGHTSTREAMS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIGHTSTREAMS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LINEA (*gnosis.eth.EthereumNetwork attribute*), 78
LINEA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LINQTO_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIQUICHAIN (*gnosis.eth.EthereumNetwork attribute*), 78
LISINSKI (*gnosis.eth.EthereumNetwork attribute*), 78
LISK_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LIVEPLEX_ORACLEEVM (*gnosis.eth.EthereumNetwork attribute*), 78
LIVING_ASSETS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
load_contract_interface() (*in module gnosis.eth.contracts*), 21
logIndex (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
LogReceiptDecoded (*class in gnosis.eth.typing*), 51
LOOPNETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
LUCID_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 78
LUCKY_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 78
LUDAN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
LUKSO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
LUKSO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
LUMOZ_TESTNET_ALPHA (*gnosis.eth.EthereumNetwork attribute*), 78
LYCAN_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 78
LYRA_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 78

M

MAAL_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 78
MAALCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
MAINNETZ_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 78
MAINNETZ_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 78
MAISTESTSUBNET (*gnosis.eth.EthereumNetwork attribute*), 78
MAMMOTH_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MANTA_PACIFIC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MANTA_PACIFIC_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
MANTIS_TESTNET_HEXAPOD (*gnosis.eth.EthereumNetwork attribute*), 79
MANTLE (*gnosis.eth.EthereumNetwork attribute*), 79
MANTLE_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
MANTLE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
MAP_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MAP_MAKALU (*gnosis.eth.EthereumNetwork attribute*), 79
MARKR_GO (*gnosis.eth.EthereumNetwork attribute*), 79
MARO_BLOCKCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MAS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
master_copy (*gnosis.safe.safe.SafeInfo attribute*), 109
master_copy_address (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
MATHCHAIN (*gnosis.eth.EthereumNetwork attribute*), 79
MATHCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
MAXXCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MCH_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
MDGL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
MELD (*gnosis.eth.EthereumNetwork attribute*), 79
MEMO_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79

MERKLE_SCAN (*gnosis.eth.EthereumNetwork attribute*), 79
MESHNYAN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
METACHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
METACHAIN_ONE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
METADIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
METADIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
METADOT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
METADOT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
METAL_C_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 79
METAL_TAHOE_C_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 79
METAPLAYERONE_DUBAI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 79
METAPLAYERONE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 79
METER_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
METER_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MethodCanOnlyBeCalledFromThisContract, 99
METIS_ANDROMEDA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
METIS_GOERLI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
METIS_STARDUST_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MEVERSE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MEVERSE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MIEXS_SMARTCHAIN (*gnosis.eth.EthereumNetwork attribute*), 80
MILKOMEDA_A1_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MILKOMEDA_A1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MILKOMEDA_C1_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MILKOMEDA_C1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MILVINE (*gnosis.eth.EthereumNetwork attribute*), 80
MIND_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MIND_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MINTARA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MINTARA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MINTME_COM_COIN (*gnosis.eth.EthereumNetwork attribute*), 80
MIX (*gnosis.eth.EthereumNetwork attribute*), 80
MIXIN_VIRTUAL_MACHINE (*gnosis.eth.EthereumNetwork attribute*), 80
MIZANA (*gnosis.eth.EthereumNetwork attribute*), 80
MIZANA_MIXNET (*gnosis.eth.EthereumNetwork attribute*), 80
MIZANA_PRIVNET (*gnosis.eth.EthereumNetwork attribute*), 80
MIZANA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
mk_contract_address() (*in module gnosis.eth.utils*), 53
mk_contract_address_2() (*in module gnosis.eth.utils*), 53
MOAC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 80
MOAC_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MODE (*gnosis.eth.EthereumNetwork attribute*), 80
MODE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 80
MODULARIUM (*gnosis.eth.EthereumNetwork attribute*), 80
module
    gnosis, 127
    gnosis.eth, 53
    gnosis.eth.clients, 13
    gnosis.eth.clients.blockscout_client, 7
    gnosis.eth.clients.contract_metadata, 9
    gnosis.eth.clients.etherscan_client, 9
    gnosis.eth.clients.sourcify_client, 12
    gnosis.eth.constants, 36
    gnosis.eth.contracts, 18
    gnosis.eth.django, 23
    gnosis.eth.django.serializers, 22
    gnosis.eth.django.validators, 23
    gnosis.eth.ethereum_client, 36
    gnosis.eth.oracles, 28
    gnosis.eth.oracles.abis, 23
    gnosis.eth.oracles.abis.aave_abis, 23
    gnosis.eth.oracles.abis.balancer_abis, 23
    gnosis.eth.oracles.abis.curve_abis, 23
    gnosis.eth.oracles.abis.mooniswap_abis, 23
    gnosis.eth.oracles.abis.yearn_abis, 23
    gnosis.eth.oracles.oracles, 23
    gnosis.eth.typing, 51
    gnosis.eth.utils, 52
    gnosis.safe, 119

gnosis.safe.exceptions, 99
gnosis.safe.multi_send, 100
gnosis.safe.proxy_factory, 101
gnosis.safe.safe, 104
gnosis.safe.safe_create2_tx, 112
gnosis.safe.safe_signature, 113
gnosis.safe.safe_tx, 116
gnosis.safe.serializers, 118
gnosis.safe.signatures, 118
ModuleManagerException, 100
modules (*gnosis.safe.safe.SafeInfo attribute*), 109
MOLEREUM_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 80
MOONBASE_ALPHA (*gnosis.eth.EthereumNetwork attribute*), 80
MOONBEAM (*gnosis.eth.EthereumNetwork attribute*), 81
MooniswapOracle (*class in gnosis.eth.oracles*), 30
MooniswapOracle (*class in gnosis.eth.oracles.oracles*), 25
MOONRIVER (*gnosis.eth.EthereumNetwork attribute*), 81
MOONROCK (*gnosis.eth.EthereumNetwork attribute*), 81
MOONROCK_OLD (*gnosis.eth.EthereumNetwork attribute*), 81
MOONSAMA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 81
MORDEN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MORDOR_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MORPH_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MOVO_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
multicall (*gnosis.eth.ethereum_client.EthereumClient property*), 46
multicall (*gnosis.eth.EthereumClient property*), 57
MultiSend (*class in gnosis.safe.multi_send*), 100
MULTISEND_ADDRESSES (*gnosis.safe.multi_send.MultiSend attribute*), 100
MULTISEND_CALL_ONLY_ADDRESSES (*gnosis.safe.multi_send.MultiSend attribute*), 100
MultiSendOperation (*class in gnosis.safe.multi_send*), 101
MultiSendTx (*class in gnosis.safe.multi_send*), 101
MULTIVAC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
MUMBAI (*gnosis.eth.EthereumNetwork attribute*), 81
MUNODE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MUSICOIN (*gnosis.eth.EthereumNetwork attribute*), 81
MUSTER_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
MXC_WANNSEE_ZKEVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MXC_ZKEVM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
MYOWN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
MYTHICAL_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 81

N

NAHMII_3_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAHMII_3_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAHMII_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAHMII_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
name (*gnosis.eth.clients.contract_metadata.ContractMetadata attribute*), 9
name (*gnosis.eth.clients.ContractMetadata attribute*), 15
name (*gnosis.eth.ethereum_client.Erc20Info attribute*), 37
name (*gnosis.eth.ethereum_client.Erc721Info attribute*), 41
NATIV3_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
NATIV3_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAUTILUS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAUTILUS_PROTEUS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
NAUTILUS_TRITION_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 81
NEBULA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
NEON_EVM_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 81
NEON_EVM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEON_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEONLINK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 81
NEONLINK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 81
NEPAL_BLOCKCHAIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 82
NETWORK_WITH_API_URL (*gnosis.eth.clients.etherscan_client.EtherscanClient attribute*), 9
NETWORK_WITH_API_URL (*gnosis.eth.clients.EtherscanClient attribute*), 15

NETWORK_WITH_URL (*gnosis.eth.clients.blockscout_client.BlockscoutClient attribute*), 7
NETWORK_WITH_URL (*gnosis.eth.clients.BlockscoutClient attribute*), 13
NETWORK_WITH_URL (*gnosis.eth.clients.etherscan_client.EtherscanClient attribute*), 10
NETWORK_WITH_URL (*gnosis.eth.clients.EtherscanClient attribute*), 16
NEUROCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEUROCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEUTRINOS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEWTON (*gnosis.eth.EthereumNetwork attribute*), 82
NEWTON_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEXI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NEXI_V2_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
nonce (*gnosis.safe.safe.SafeInfo attribute*), 109
NonceTooHigh, 98
NonceTooLow, 98
NORDEK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NORMAL (*gnosis.eth.ethereum_client.TxSpeed attribute*), 51
NORMAL (*gnosis.eth.TxSpeed attribute*), 98
NotEnoughSafeTransactionGas, 100
NOVA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 82
NTITY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NULL_ADDRESS (*gnosis.eth.ethereum_client.EthereumClient attribute*), 43
NULL_ADDRESS (*gnosis.eth.EthereumClient attribute*), 53
NUMBERS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
NUMBERS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
NUME (*gnosis.eth.EthereumNetwork attribute*), 82

O

OASIS_EMERALD (*gnosis.eth.EthereumNetwork attribute*), 82
OASIS_EMERALD_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
OASIS_SAPPHIRE (*gnosis.eth.EthereumNetwork attribute*), 82
OASIS_SAPPHIRE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
OASISCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OASYS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OCTASPACE (*gnosis.eth.EthereumNetwork attribute*), 82
OEBLOCK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
OHO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OKEXCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
OKXCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OLYMPIC (*gnosis.eth.EthereumNetwork attribute*), 82
OM_PLATFORM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
OMAX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OMCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 82
OMNI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 82
ONELEDGER_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
ONELEDGER_TESTNET_FRANKENSTEIN (*gnosis.eth.EthereumNetwork attribute*), 83
OnlyOwnersCanApproveAHash, 100
ONTOLOGY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
ONTOLOGY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
ONUS_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
ONUS_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OONE_CHAIN_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 83
OONE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OORT_ASCRAEUS (*gnosis.eth.EthereumNetwork attribute*), 83
OORT_HUYGENS (*gnosis.eth.EthereumNetwork attribute*), 83
OORT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
OORT_MAINNETDEV (*gnosis.eth.EthereumNetwork attribute*), 83
OP_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPAL_TESTNET_BY_UNIQUE (*gnosis.eth.EthereumNetwork attribute*), 83
OPBNB_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPBNB_TESTNET (*gnosis.eth.EthereumNetwork attribute*),

83
OPENCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPENCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPENPIECE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPENPIECE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPENVESSEL (*gnosis.eth.EthereumNetwork attribute*), 83
OPSIDE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPTIMISM (*gnosis.eth.EthereumNetwork attribute*), 83
OPTIMISM_BEDROCK_GOERLI_ALPHA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPTIMISM_GOERLI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
OPTIMISM_KOVAN (*gnosis.eth.EthereumNetwork attribute*), 83
OPULENT_X_BETA (*gnosis.eth.EthereumNetwork attribute*), 83
ORDERLY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 83
ORDERLY_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 83
ORIGIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
ORIGINTRAIL_PARACHAIN (*gnosis.eth.EthereumNetwork attribute*), 83
ORLANDO_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 84
owner (*gnosis.safe.safe_signature.SafeSignature property*), 114
owner (*gnosis.safe.safe_signature.SafeSignatureApprovedHash property*), 114
owner (*gnosis.safe.safe_signature.SafeSignatureContract property*), 115
owner (*gnosis.safe.safe_signature.SafeSignatureEOA property*), 115
owner (*gnosis.safe.safe_signature.SafeSignatureEthSign property*), 116
OwnerManagerException, 100
owners (*gnosis.safe.safe.SafeInfo attribute*), 109
owners (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
OYCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
OYCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
OZONE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
OZONE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84

P

P12_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 84
PAIR_INIT_CODE (*gnosis.eth.oracles.oracles.UniswapV2Oracle attribute*), 26
PAIR_INIT_CODE (*gnosis.eth.oracles.SushiswapOracle attribute*), 31
PAIR_INIT_CODE (*gnosis.eth.oracles.UniswapV2Oracle attribute*), 32
PALETTE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PALETTE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PALM (*gnosis.eth.EthereumNetwork attribute*), 84
PALM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PANDOPROJECT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PANDOPROJECT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PARIBU_NET_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PARIBU_NET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
parse_signature() (*gnosis.safe.safe_signature.SafeSignature class method*), 114
partial_match (*gnosis.eth.clients.contract_metadata.ContractMetadata attribute*), 9
partial_match (*gnosis.eth.clients.ContractMetadata attribute*), 15
PARTYCHAIN (*gnosis.eth.EthereumNetwork attribute*), 84
PATEX (*gnosis.eth.EthereumNetwork attribute*), 84
PATEX_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PAWCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PAXB_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
payment (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
payment_ether (*gnosis.safe.safe_create2_tx.SafeCreate2Tx property*), 112
payment_receiver (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
payment_token (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
payment_token_eth_value (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
PDC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PEERPAY (*gnosis.eth.EthereumNetwork attribute*), 84

PEGGLECOIN (*gnosis.eth.EthereumNetwork attribute*), 84
PEGO_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 84
PEPCHAIN_CHURCHILL (*gnosis.eth.EthereumNetwork attribute*), 84
PEPE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PEPENETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 84
PEPERIUM_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 84
PERMISSION (*gnosis.eth.EthereumNetwork attribute*), 84
PGN_PUBLIC_GOODS_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 84
PHALA_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 84
PHI_NETWORK_V1 (*gnosis.eth.EthereumNetwork attribute*), 85
PHI_NETWORK_V2 (*gnosis.eth.EthereumNetwork attribute*), 85
PHOENIX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
PIECE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 85
PIRL (*gnosis.eth.EthereumNetwork attribute*), 85
PIXIE_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
PIXIE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 85
PLANQ_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
PLATON_DEV_TESTNET2 (*gnosis.eth.EthereumNetwork attribute*), 85
PLATON_DEV_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 85
PLATON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
PLAYA3ULL_GAMES (*gnosis.eth.EthereumNetwork attribute*), 85
PLIAN_MAINNET_MAIN (*gnosis.eth.EthereumNetwork attribute*), 85
PLIAN_MAINNET_SUBCHAIN_1 (*gnosis.eth.EthereumNetwork attribute*), 85
PLIAN_TESTNET_MAIN (*gnosis.eth.EthereumNetwork attribute*), 85
PLIAN_TESTNET_SUBCHAIN_1 (*gnosis.eth.EthereumNetwork attribute*), 85
PLINGA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
POA_NETWORK_CORE (*gnosis.eth.EthereumNetwork attribute*), 85
POA_NETWORK_SOKOL (*gnosis.eth.EthereumNetwork attribute*), 85
POCRNET (*gnosis.eth.EthereumNetwork attribute*), 85
POLIS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
POLIS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON_SUPERNET_ARIANEE (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON_ZKEVM (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON_ZKEVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON_ZKEVM_TESTNET_OLD (*gnosis.eth.EthereumNetwork attribute*), 85
POLYGON_ZKEVM_TESTNET_PRE_AUDIT_UPGRADED (*gnosis.eth.EthereumNetwork attribute*), 85
POLYJUICE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 85
POLYSMARTCHAIN (*gnosis.eth.EthereumNetwork attribute*), 85
PoolTogetherOracle (*class in gnosis.eth.oracles*), 31
PoolTogetherOracle (*class in gnosis.eth.oracles.oracles*), 25
POPCATEUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 85
PORTAL_FANTASY_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 86
PORTAL_FANTASY_CHAIN_TEST (*gnosis.eth.EthereumNetwork attribute*), 86
POSICHAIN_DEVNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 86
POSICHAIN_DEVNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 86
POSICHAIN_MAINNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 86
POSICHAIN_TESTNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 86
PRICE_CONVERSION_CONSTANT (*gnosis.eth.oracles.UniswapV3Oracle attribute*), 34
PriceOracle (*class in gnosis.eth.oracles*), 31
PriceOracle (*class in gnosis.eth.oracles.oracles*), 25
PricePoolOracle (*class in gnosis.eth.oracles*), 31
PricePoolOracle (*class in gnosis.eth.oracles.oracles*), 25
PRIMUSCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 86
private_key_to_address() (*gnosis.eth.ethereum_client.EthereumClient static method*), 46
private_key_to_address() (*gnosis.eth.EthereumClient static method*), 57
PROOF_OF_MEMES (*gnosis.eth.EthereumNetwork attribute*), 86
PROTOJUMBO_TESTNET (*gnosis.eth.EthereumNetwork at-*

tribute), 86
PROTON_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
proxy_factory_address (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
PROXY_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
ProxyFactory (*class in gnosis.safe*), 119
ProxyFactory (*class in gnosis.safe.proxy_factory*), 101
ProxyFactoryV100 (*class in gnosis.safe.proxy_factory*), 103
ProxyFactoryV111 (*class in gnosis.safe.proxy_factory*), 103
ProxyFactoryV130 (*class in gnosis.safe.proxy_factory*), 103
ProxyFactoryV141 (*class in gnosis.safe.proxy_factory*), 103
PUBLICMINT_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 86
PUBLICMINT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 86
PUBLICMINT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
PULSECHAIN (*gnosis.eth.EthereumNetwork attribute*), 86
PULSECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
PULSECHAIN_TESTNET_V2B (*gnosis.eth.EthereumNetwork attribute*), 86
PULSECHAIN_TESTNET_V3 (*gnosis.eth.EthereumNetwork attribute*), 86
PULSECHAIN_TESTNET_V4 (*gnosis.eth.EthereumNetwork attribute*), 86

Q

Q_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 87
Q_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 87
QEASYWEB3_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
QITMEER (*gnosis.eth.EthereumNetwork attribute*), 86
QITMEER_NETWORK_MIXNET (*gnosis.eth.EthereumNetwork attribute*), 86
QITMEER_NETWORK_PRIVNET (*gnosis.eth.EthereumNetwork attribute*), 86
QITMEER_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
QL1 (*gnosis.eth.EthereumNetwork attribute*), 86
QL1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
QUADRANS_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 86
QUADRANS_BLOCKCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
quantity (*gnosis.eth.oracles.oracles.UnderlyingToken attribute*), 25
quantity (*gnosis.eth.oracles.UnderlyingToken attribute*), 32
QUANTUM_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 86
QUANTUM_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 86
QUARIX (*gnosis.eth.EthereumNetwork attribute*), 86
QUARIX_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKBLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_ROOT (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_2 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_3 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_4 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_5 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_6 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_DEVNET_SHARD_7 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_ROOT (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_0 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_1 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_2 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_3 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_4 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_5 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_6 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARKCHAIN_MAINNET_SHARD_7 (*gnosis.eth.EthereumNetwork attribute*), 87
QUARTZ_BY_UNIQUE (*gnosis.eth.EthereumNetwork attribute*), 87
QUOKKACOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 87

R

RABA_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 87
RABBIT_ANALOG_TESTNET_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 87
RANGERS_PROTOCOL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 87
RANGERS_PROTOCOL_TESTNET_ROBIN (*gnosis.eth.EthereumNetwork attribute*), 87
RAPTORCHAIN (*gnosis.eth.EthereumNetwork attribute*), 87
raw_batch_request() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
raw_batch_request() (*gnosis.eth.EthereumClient method*), 57
RAZOR_SKALE_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 87
REALCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 87
REAPCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
REAPCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
recommended_gas() (*gnosis.safe.safe_tx.SafeTx method*), 117
recommended_gas() (*gnosis.safe.SafeTx method*), 126
REDBELLY_NETWORK_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 88
REDBELLY_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
REDBELLY_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
REDBELLY_NETWORK_TGE (*gnosis.eth.EthereumNetwork attribute*), 88
REDECOIN (*gnosis.eth.EthereumNetwork attribute*), 88
REDLIGHT_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
REDSTONE_HOLESKY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
REI_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
REI_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
REI_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 88
remove_swarm_metadata() (*in module gnosis.eth.utils*), 53
removed (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
ReplacementTransactionUnderpriced, 98
RESINCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
retrieve_all_info() (*gnosis.safe.Safe method*), 124
retrieve_all_info() (*gnosis.safe.safe.Safe method*), 107
retrieve_code() (*gnosis.safe.Safe method*), 124
retrieve_code() (*gnosis.safe.safe.Safe method*), 107
retrieve_domain_separator() (*gnosis.safe.Safe method*), 124
retrieve_domain_separator() (*gnosis.safe.safe.Safe method*), 107
retrieve_fallback_handler() (*gnosis.safe.Safe method*), 124
retrieve_fallback_handler() (*gnosis.safe.safe.Safe method*), 107
retrieve_guard() (*gnosis.safe.Safe method*), 124
retrieve_guard() (*gnosis.safe.safe.Safe method*), 107
retrieve_is_hash_approved() (*gnosis.safe.Safe method*), 124
retrieve_is_hash_approved() (*gnosis.safe.safe.Safe method*), 107
retrieve_is_message_signed() (*gnosis.safe.Safe method*), 124
retrieve_is_message_signed() (*gnosis.safe.safe.Safe method*), 107
retrieve_is_owner() (*gnosis.safe.Safe method*), 124
retrieve_is_owner() (*gnosis.safe.safe.Safe method*), 107
retrieve_master_copy_address() (*gnosis.safe.Safe method*), 124
retrieve_master_copy_address() (*gnosis.safe.safe.Safe method*), 108
retrieve_modules() (*gnosis.safe.Safe method*), 124
retrieve_modules() (*gnosis.safe.safe.Safe method*), 108
retrieve_nonce() (*gnosis.safe.Safe method*), 125
retrieve_nonce() (*gnosis.safe.safe.Safe method*), 108
retrieve_owners() (*gnosis.safe.Safe method*), 125
retrieve_owners() (*gnosis.safe.safe.Safe method*), 108
retrieve_threshold() (*gnosis.safe.Safe method*), 125
retrieve_threshold() (*gnosis.safe.safe.Safe method*), 108
retrieve_version() (*gnosis.safe.Safe method*), 125
retrieve_version() (*gnosis.safe.safe.Safe method*), 108
RIKEZA_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
RIKEZA_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
RINIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
RINIA_TESTNET_OLD (*gnosis.eth.EthereumNetwork attribute*), 88
RINKEBY (*gnosis.eth.EthereumNetwork attribute*), 88
RISE_OF_THE_WARBOTS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88

ROLLUX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
ROLLUX_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
ROOTSTOCK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
ROOTSTOCK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
ROPSTEN (*gnosis.eth.EthereumNetwork attribute*), 88
router (*gnosis.eth.oracles.UniswapV3Oracle property*), 35
router_address (*gnosis.eth.oracles.UniswapV2Oracle attribute*), 34
ROUTER_ADDRESSES (*gnosis.eth.oracles.oracles.UniswapV2Oracle attribute*), 26
ROUTER_ADDRESSES (*gnosis.eth.oracles.SushiswapOracle attribute*), 31
ROUTER_ADDRESSES (*gnosis.eth.oracles.UniswapV2Oracle attribute*), 33
ROUTER_ADDRESSES (*gnosis.eth.oracles.UniswapV3Oracle attribute*), 34
RUBY_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
RUBY_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
RUPAYA (*gnosis.eth.EthereumNetwork attribute*), 88

S

SAAKURU_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 88
SAAKURU_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 88
Safe (*class in gnosis.safe*), 121
Safe (*class in gnosis.safe.safe*), 104
safe_address (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
SAFE_MESSAGE_TYPEHASH (*gnosis.safe.Safe attribute*), 121
SAFE_MESSAGE_TYPEHASH (*gnosis.safe.safe.Safe attribute*), 104
safe_nonce (*gnosis.safe.safe_tx.SafeTx property*), 117
safe_nonce (*gnosis.safe.SafeTx property*), 127
safe_setup_data (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
safe_tx_hash (*gnosis.safe.safe_tx.SafeTx property*), 117
safe_tx_hash (*gnosis.safe.SafeTx property*), 127
safe_tx_hash_preimage (*gnosis.safe.safe_tx.SafeTx property*), 117
safe_tx_hash_preimage (*gnosis.safe.SafeTx property*), 127
safe_version (*gnosis.safe.safe_tx.SafeTx property*), 117
safe_version (*gnosis.safe.SafeTx property*), 127
SafeCreate2Tx (*class in gnosis.safe.safe_create2_tx*), 112
SafeCreate2TxBuilder (*class in gnosis.safe.safe_create2_tx*), 112
SafeInfo (*class in gnosis.safe.safe*), 109
SafeMultisigEstimateTxSerializer (*class in gnosis.safe.serializers*), 118
SafeMultisigTxSerializer (*class in gnosis.safe.serializers*), 118
SafeMultisigTxSerializerV1 (*class in gnosis.safe.serializers*), 118
SafeOperationEnum (*class in gnosis.safe*), 125
SafeServiceException, 100, 126
SafeSignature (*class in gnosis.safe.safe_signature*), 113
SafeSignatureApprovedHash (*class in gnosis.safe.safe_signature*), 114
SafeSignatureContract (*class in gnosis.safe.safe_signature*), 114
SafeSignatureEOA (*class in gnosis.safe.safe_signature*), 115
SafeSignatureEthSign (*class in gnosis.safe.safe_signature*), 115
SafeSignatureException, 116
SafeSignatureSerializer (*class in gnosis.safe.serializers*), 118
SafeSignatureType (*class in gnosis.safe.safe_signature*), 116
SafeTransactionFailedWhenGasPriceAndSafeTxGasEmpty, 100
SafeTx (*class in gnosis.safe*), 126
SafeTx (*class in gnosis.safe.safe_tx*), 116
SafeV001 (*class in gnosis.safe.safe*), 109
SafeV100 (*class in gnosis.safe.safe*), 110
SafeV111 (*class in gnosis.safe.safe*), 110
SafeV120 (*class in gnosis.safe.safe*), 110
SafeV130 (*class in gnosis.safe.safe*), 111
SafeV141 (*class in gnosis.safe.safe*), 111
SAKURA (*gnosis.eth.EthereumNetwork attribute*), 88
salt_nonce (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
SANR_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 88
SAPPHIRE_BY_UNIQUE (*gnosis.eth.EthereumNetwork attribute*), 89
SARDIS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SARDIS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SATOSHICHAIN_MAINNET (*gnosis.eth.EthereumNetwork*

attribute), 89
SATOSHICHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SATOSHIE (*gnosis.eth.EthereumNetwork attribute*), 89
SATOSHIE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCALIND (*gnosis.eth.EthereumNetwork attribute*), 89
SCALIND_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCOLCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCOLCOIN_WEICHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCRIPT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCROLL (*gnosis.eth.EthereumNetwork attribute*), 89
SCROLL_ALPHA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCROLL_PRE_ALPHA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SCROLL_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SECURECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SECURECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SEELE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SEI_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 89
send_eth_to() (*gnosis.eth.ethereum_client.EthereumClient method*), 46
send_eth_to() (*gnosis.eth.EthereumClient method*), 57
send_multisig_tx() (*gnosis.safe.Safe method*), 125
send_multisig_tx() (*gnosis.safe.safe.Safe method*), 108
send_raw_transaction() (*gnosis.eth.ethereum_client.EthereumClient method*), 47
send_raw_transaction() (*gnosis.eth.EthereumClient method*), 57
send_tokens() (*gnosis.eth.ethereum_client.Erc20Manager method*), 41
send_transaction() (*gnosis.eth.ethereum_client.EthereumClient method*), 47
send_transaction() (*gnosis.eth.EthereumClient method*), 57
send_unsigned_transaction() (*gnosis.eth.ethereum_client.EthereumClient method*), 47
send_unsigned_transaction() (*gnosis.eth.EthereumClient method*), 57
SenderAccountNotFoundInNode, 98
SENJEPOWERS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SENJEPOWERS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SEPOLIA (*gnosis.eth.EthereumNetwork attribute*), 89
SEPOLIA_PGN_PUBLIC_GOODS_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 89
set_eip1559_fees() (*gnosis.eth.ethereum_client.EthereumClient method*), 47
set_eip1559_fees() (*gnosis.eth.EthereumClient method*), 58
SETHEUM (*gnosis.eth.EthereumNetwork attribute*), 89
Sha3HashField (*class in gnosis.eth.django.serializers*), 22
SHARDEUM_LIBERTY_1_X (*gnosis.eth.EthereumNetwork attribute*), 89
SHARDEUM_LIBERTY_2_X (*gnosis.eth.EthereumNetwork attribute*), 89
SHARDEUM_SPHINX_1_X (*gnosis.eth.EthereumNetwork attribute*), 89
SHERPAX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 89
SHERPAX_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 89
SHIBACHAIN (*gnosis.eth.EthereumNetwork attribute*), 89
SHIBARIUM (*gnosis.eth.EthereumNetwork attribute*), 90
SHIBARIUM_BETA (*gnosis.eth.EthereumNetwork attribute*), 90
SHIDEN (*gnosis.eth.EthereumNetwork attribute*), 90
SHIMMEREVM (*gnosis.eth.EthereumNetwork attribute*), 90
SHIMMEREVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SHIMMEREVM_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 90
SHIMMEREVM_TESTNET_DEPRECATED_1072 (*gnosis.eth.EthereumNetwork attribute*), 90
SHINARIUM_BETA (*gnosis.eth.EthereumNetwork attribute*), 90
SHINARIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 90
SHRAPNEL_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 90
SHRAPNEL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SHYFT_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 90
SHYFT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SIBERIUM_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 90
SIBERIUM_TEST_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 90
sign() (*gnosis.safe.safe_tx.SafeTx method*), 117
sign() (*gnosis.safe.SafeTx method*), 127

signature_split() (*in module gnosis.safe.signatures*), 118
signature_to_bytes() (*in module gnosis.safe.signatures*), 118
signature_type (*gnosis.safe.safe_signature.SafeSignature property*), 114
signature_type (*gnosis.safe.safe_signature.SafeSignatureApprovedHash property*), 114
signature_type (*gnosis.safe.safe_signature.SafeSignatureContract property*), 115
signature_type (*gnosis.safe.safe_signature.SafeSignatureEOA property*), 115
signature_type (*gnosis.safe.safe_signature.SafeSignatureEthSign property*), 116
SignatureNotProvidedByOwner, 100, 127
signatures_to_bytes() (*in module gnosis.safe.signatures*), 118
SignaturesDataTooShort, 100
SignatureSerializer (*class in gnosis.eth.django.serializers*), 22
signers (*gnosis.safe.safe_tx.SafeTx property*), 117
signers (*gnosis.safe.SafeTx property*), 127
simulate_tx_accessor_address (*gnosis.safe.Safe property*), 125
simulate_tx_accessor_address (*gnosis.safe.safe.Safe property*), 109
SINGULARITY_ZERO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 90
SINGULARITY_ZERO_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SIRIUSNET (*gnosis.eth.EthereumNetwork attribute*), 90
SIRIUSNET_V2 (*gnosis.eth.EthereumNetwork attribute*), 90
SIX_PROTOCOL (*gnosis.eth.EthereumNetwork attribute*), 90
SIX_PROTOCOL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SJATSH (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_CALYPSO_HUB (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_CALYPSO_HUB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_EUROPA_HUB (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_EUROPA_HUB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_NEBULA_HUB (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_NEBULA_HUB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_TITAN_HUB (*gnosis.eth.EthereumNetwork attribute*), 90
SKALE_TITAN_HUB_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 90
SLOW (*gnosis.eth.ethereum_client.TxSpeed attribute*), 51
SLOW (*gnosis.eth.TxSpeed attribute*), 98
slow_w3 (*gnosis.eth.EthereumClient attribute*), 58
SLOWEST (*gnosis.eth.ethereum_client.TxSpeed attribute*), 51
SLOWEST (*gnosis.eth.TxSpeed attribute*), 98
SMART_BITCOIN_CASH (*gnosis.eth.EthereumNetwork attribute*), 91
SMART_BITCOIN_CASH_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SMART_HOST_TEKNOLOJI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SMART_LAYER_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 91
SMART_LAYER_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SMART_TRADE_NETWORKS (*gnosis.eth.EthereumNetwork attribute*), 91
SMARTMESH_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 90
SOCIAL_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SOMA_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SOMA_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SONGBIRD_CANARY_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 91
sorted_signers (*gnosis.safe.safe_tx.SafeTx property*), 117
sorted_signers (*gnosis.safe.SafeTx property*), 127
SOTERONE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SOTERONE_MAINNET_OLD (*gnosis.eth.EthereumNetwork attribute*), 91
SourcifyClient (*class in gnosis.eth.clients*), 18
SourcifyClient (*class in gnosis.eth.clients.sourcify_client*), 12
SourcifyClientConfigurationProblem, 12, 18
SourcifyClientException, 12, 18
SOVERUN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SOVERUN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SPORTS_CHAIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 91
SPS (*gnosis.eth.EthereumNetwork attribute*), 91
SPS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91

STAR_SOCIAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
STEP_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 91
STEP_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
STORAGECHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
STORAGECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
STRATOS (*gnosis.eth.EthereumNetwork attribute*), 91
STRATOS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
STREAMUX_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 91
STRUCTX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SUPER_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 91
SUPER_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 91
SuperfluidOracle (*class in gnosis.eth.oracles*), 31
SUR_BLOCKCHAIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 91
SushiswapOracle (*class in gnosis.eth.oracles*), 31
SUSONO (*gnosis.eth.EthereumNetwork attribute*), 91
SWAPDEX (*gnosis.eth.EthereumNetwork attribute*), 91
SWISSDLT (*gnosis.eth.EthereumNetwork attribute*), 92
SWISSTRONIK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
SX_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
SX_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
symbol (*gnosis.eth.ethereum_client.Erc20Info attribute*), 37
symbol (*gnosis.eth.ethereum_client.Erc721Info attribute*), 41
SYMPLEXIA_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 92
SYNAPSE_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
SYSCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
SYSCOIN_TANENBAUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92

T

T_E_A_M_BLOCKCHAIN (*gnosis.eth.EthereumNetwork attribute*), 94
T_EKTA (*gnosis.eth.EthereumNetwork attribute*), 94
TAF_ECO_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TAIKO_ALPHA_2_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TAIKO_ELDFELL_L3 (*gnosis.eth.EthereumNetwork attribute*), 92
TAIKO_GRIMSVOTN_L2 (*gnosis.eth.EthereumNetwork attribute*), 92
TAIKO_JOLNIR_L2 (*gnosis.eth.EthereumNetwork attribute*), 92
TAIKO_KATLA_L2 (*gnosis.eth.EthereumNetwork attribute*), 92
TANGLE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TANSSI_EVM_CONTAINERCHAIN (*gnosis.eth.EthereumNetwork attribute*), 92
TAO_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 92
TARAXA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TARAXA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TAYCAN (*gnosis.eth.EthereumNetwork attribute*), 92
TAYCAN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TBSI_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TBSI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TBWG_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 92
TCG_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TECHPAY_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TECTUM_EMISSION_TOKEN (*gnosis.eth.EthereumNetwork attribute*), 92
TELEPORT (*gnosis.eth.EthereumNetwork attribute*), 92
TELEPORT_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TELOS_EVM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 92
TELOS_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 92
TEN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
TENET (*gnosis.eth.EthereumNetwork attribute*), 93
TENET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
TESLAFUNDS (*gnosis.eth.EthereumNetwork attribute*), 93
TESTNET_BEONE_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 93
THAICHAIN (*gnosis.eth.EthereumNetwork attribute*), 93
THAICHAIN_2_0_THAIFI (*gnosis.eth.EthereumNetwork attribute*), 93
THE_ROOT_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 93

THE_ROOT_NETWORK_PORCINI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
THETA_AMBER_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
THETA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 93
THETA_SAPPHIRE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
THETA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_MAINNET_CHAIN_0 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_MAINNET_CHAIN_1 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_MAINNET_CHAIN_103 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_MAINNET_CHAIN_2 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_TESTNET_CHAIN_0 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_TESTNET_CHAIN_1 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_TESTNET_CHAIN_103 (*gnosis.eth.EthereumNetwork attribute*), 93
THINKIUM_TESTNET_CHAIN_2 (*gnosis.eth.EthereumNetwork attribute*), 93
threshold (*gnosis.safe.safe.SafeInfo attribute*), 109
threshold (*gnosis.safe.safe_create2_tx.SafeCreate2Tx attribute*), 112
ThresholdNeedsToBeDefined, 100
THUNDERCORE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 93
THUNDERCORE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
TILTYARD_SUBNET (*gnosis.eth.EthereumNetwork attribute*), 93
TIPBOXCOIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 93
TIPBOXCOIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
TITAN (*gnosis.eth.EthereumNetwork attribute*), 93
TLCHAIN_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 93
TMY_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 93
to_internal_value() (*gnosis.eth.django.serializers.EthereumAddressField method*), 22
to_internal_value() (*gnosis.eth.django.serializers.HexadecimalField method*), 22
to_representation() (*gnosis.eth.django.serializers.EthereumAddressField method*), 22
to_representation() (*gnosis.eth.django.serializers.HexadecimalField method*), 22
token_address (*gnosis.eth.ethereum_client.TokenBalance attribute*), 47
token_address (*gnosis.eth.typing.BalanceDict attribute*), 51
TokenBalance (*class in gnosis.eth.ethereum_client*), 47
TOKI_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 93
TOKI_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 93
TOMB_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TOMOCHAIN (*gnosis.eth.EthereumNetwork attribute*), 94
TOMOCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TOOL_GLOBAL_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TOOL_GLOBAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TOP_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TOP_MAINNET_EVM (*gnosis.eth.EthereumNetwork attribute*), 94
topics (*gnosis.eth.typing.LogReceiptDecoded attribute*), 51
TORONET_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TORONET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TORUS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TORUS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
trace_block() (*gnosis.eth.ethereum_client.TracingManager method*), 48
trace_blocks() (*gnosis.eth.ethereum_client.TracingManager method*), 48
trace_filter() (*gnosis.eth.ethereum_client.TracingManager method*), 48
trace_transaction() (*gnosis.eth.ethereum_client.TracingManager method*), 50
trace_transactions() (*gnosis.eth.ethereum_client.TracingManager method*), 50
tracing (*gnosis.eth.EthereumClient attribute*), 58
TracingManager (*class in gnosis.eth.ethereum_client*), 48
TransactionAlreadyImported, 98
transactionHash (*gnosis.eth.typing.LogReceiptDecoded attribute*),

51
transactionIndex (*gnosis.eth.typing.LogReceiptDecoded attribute*), 52
TransactionQueueLimitReached, 98
TransactionResponseSerializer (*class in gnosis.eth.django.serializers*), 22
TransactionSerializer (*class in gnosis.eth.django.serializers*), 22
TRANSFER_TOPIC (*gnosis.eth.ethereum_client.Erc20Manager attribute*), 38
TRANSFER_TOPIC (*gnosis.eth.ethereum_client.Erc721Manager attribute*), 41
TREASURENET_MAINNET_ALPHA (*gnosis.eth.EthereumNetwork attribute*), 94
TREASURENET_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TRES_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TRES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TRITANIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TRUST_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
TTCOIN_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
TURKEY_DEMO_DEV (*gnosis.eth.EthereumNetwork attribute*), 94
tx (*gnosis.eth.ethereum_client.EthereumTxSent attribute*), 47
tx (*gnosis.eth.EthereumTxSent attribute*), 98
tx_hash (*gnosis.eth.ethereum_client.EthereumTxSent attribute*), 47
tx_hash (*gnosis.eth.EthereumTxSent attribute*), 98
tx_with_exception_handling() (*in module gnosis.eth.ethereum_client*), 51
TxSpeed (*class in gnosis.eth*), 98
TxSpeed (*class in gnosis.eth.ethereum_client*), 50

U

U2U_SOLARIS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
UB_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 94
UB_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
UBIQ (*gnosis.eth.EthereumNetwork attribute*), 94
UBIQ_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
uint_to_address() (*in module gnosis.safe.safe_signature*), 116
ULTRA_PRO_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
ULTRON_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 94
ULTRON_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
UnderlyingToken (*class in gnosis.eth.oracles*), 32
UnderlyingToken (*class in gnosis.eth.oracles.oracles*), 25
UNICORN_ULTRA_NEBULAS_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 94
UNIQUE (*gnosis.eth.EthereumNetwork attribute*), 94
uniswap_factory (*gnosis.eth.oracles.oracles.UniswapOracle property*), 26
uniswap_factory (*gnosis.eth.oracles.UniswapOracle property*), 32
uniswap_factory_address (*gnosis.eth.oracles.oracles.UniswapOracle property*), 26
uniswap_factory_address (*gnosis.eth.oracles.UniswapOracle property*), 32
UniswapOracle (*class in gnosis.eth.oracles*), 32
UniswapOracle (*class in gnosis.eth.oracles.oracles*), 25
UniswapV2Oracle (*class in gnosis.eth.oracles*), 32
UniswapV2Oracle (*class in gnosis.eth.oracles.oracles*), 26
UniswapV3Oracle (*class in gnosis.eth.oracles*), 34
UNKNOWN (*gnosis.eth.EthereumNetwork attribute*), 95
UnknownAccount, 99
UNREAL_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
unsign() (*gnosis.safe.safe_tx.SafeTx method*), 117
unsign() (*gnosis.safe.SafeTx method*), 127
UPTICK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
UPTN (*gnosis.eth.EthereumNetwork attribute*), 95
UPTN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
UZMI_NETWORK_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95

V

validate() (*gnosis.safe.serializers.SafeMultisigEstimateTxSerializer method*), 118
validate() (*gnosis.safe.serializers.SafeSignatureSerializer method*), 118
validate_checksumed_address() (*in module gnosis.eth.django.validators*), 23
validate_operation() (*gnosis.safe.serializers.SafeMultisigEstimateTxSerializer method*), 118

validate_v() (*gnosis.safe.serializers.SafeSignatureSerializer method*), 118
VALORBIT (*gnosis.eth.EthereumNetwork attribute*), 95
VCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VECHAIN (*gnosis.eth.EthereumNetwork attribute*), 95
VECHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
VELA1_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VELAS_EVM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VELO_LABS_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VENIDIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VENIDIUM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
VENTION_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VENTION_SMART_CHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
version (*gnosis.safe.safe.SafeInfo attribute*), 109
VERY_FAST (*gnosis.eth.ethereum_client.TxSpeed attribute*), 51
VERY_FAST (*gnosis.eth.TxSpeed attribute*), 98
VERY_SLOW (*gnosis.eth.ethereum_client.TxSpeed attribute*), 51
VERY_SLOW (*gnosis.eth.TxSpeed attribute*), 99
VEX_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
VINE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
VINUCHAIN_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 95
VINUCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
VISION_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 95
VISION_VPIONEER_TEST_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 95
VULTURE_EVM_BETA (*gnosis.eth.EthereumNetwork attribute*), 95
VYVO_SMART_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 95

W

w3 (*gnosis.eth.EthereumClient attribute*), 58
w3 (*gnosis.safe.multi_send.MultiSend property*), 101
w3 (*gnosis.safe.safe_tx.SafeTx property*), 117
w3 (*gnosis.safe.SafeTx property*), 127
w3_tx (*gnosis.safe.safe_tx.SafeTx property*), 117
w3_tx (*gnosis.safe.SafeTx property*), 127
W3GAMEZ_HOLESKY_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
WAGMI (*gnosis.eth.EthereumNetwork attribute*), 95
WANCHAIN (*gnosis.eth.EthereumNetwork attribute*), 95
WANCHAIN_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
WEB3GAMES_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 95
WEB3GAMES_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 95
WEB3Q_GALILEO (*gnosis.eth.EthereumNetwork attribute*), 96
WEB3Q_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WEB3Q_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
WEBCHAIN (*gnosis.eth.EthereumNetwork attribute*), 96
WEELINK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
WEGOCHAIN_RUBIDIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WEMIX3_0_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WEMIX3_0_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
weth_address (*gnosis.eth.oracles.oracles.UniswapV2Oracle property*), 27
weth_address (*gnosis.eth.oracles.UniswapV2Oracle property*), 34
weth_address (*gnosis.eth.oracles.UniswapV3Oracle property*), 35
WHITEBIT_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 96
WHITEBIT_NETWORK_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
WIRESHAPE_FLORIPA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
WOOPCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WORLD_TRADE_TECHNICAL_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WORLDLAND_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
WORLDLAND_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
WORLDS_CALDERA (*gnosis.eth.EthereumNetwork attribute*), 96
WYZTH_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96

X

X1_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 96
X1_FASTNET (*gnosis.eth.EthereumNetwork attribute*), 96
X1_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96

X1_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 96
X1_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
XANACHAIN (*gnosis.eth.EthereumNetwork attribute*), 96
XCAP (*gnosis.eth.EthereumNetwork attribute*), 96
XDC_APOTHEM_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 96
XDC_NETWORK (*gnosis.eth.EthereumNetwork attribute*), 96
XEROM (*gnosis.eth.EthereumNetwork attribute*), 96
XODEX (*gnosis.eth.EthereumNetwork attribute*), 96
XPLA_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96
XPLA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 96
XT_SMART_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 96

Y

YearnOracle (*class in gnosis.eth.oracles*), 35
YearnOracle (*class in gnosis.eth.oracles.oracles*), 27
YIDARK_CHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
YOOLDO_VERSE_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
YOOLDO_VERSE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
YUANCHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97

Z

ZAFIRIUM_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZCORE_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZEETH_CHAIN (*gnosis.eth.EthereumNetwork attribute*), 97
ZEETH_CHAIN_DEV (*gnosis.eth.EthereumNetwork attribute*), 97
ZENIQ (*gnosis.eth.EthereumNetwork attribute*), 97
ZENITH_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.CurveOracle attribute*), 30
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.EnzymeOracle attribute*), 30
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.oracles.CurveOracle attribute*), 24
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.oracles.EnzymeOracle attribute*), 25
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.oracles.PoolTogetherOracle attribute*), 25
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.oracles.ZerionComposedOracle attribute*), 28
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.PoolTogetherOracle attribute*), 31
ZERION_ADAPTER_ADDRESS (*gnosis.eth.oracles.ZerionComposedOracle attribute*), 36
zerion_adapter_contract (*gnosis.eth.oracles.oracles.ZerionComposedOracle property*), 28
zerion_adapter_contract (*gnosis.eth.oracles.ZerionComposedOracle property*), 36
ZerionComposedOracle (*class in gnosis.eth.oracles*), 35
ZerionComposedOracle (*class in gnosis.eth.oracles.oracles*), 28
ZETACHAIN_ATHENS_3_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZETACHAIN_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZHEJIANG (*gnosis.eth.EthereumNetwork attribute*), 97
ZILLIQA_2_EVM_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZILLIQA_EVM (*gnosis.eth.EthereumNetwork attribute*), 97
ZILLIQA_EVM_DEVNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZILLIQA_EVM_ISOLATED_SERVER (*gnosis.eth.EthereumNetwork attribute*), 97
ZILLIQA_EVM_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZKATANA (*gnosis.eth.EthereumNetwork attribute*), 97
ZKFAIR_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZKFAIR_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZKSYNC_ERA_GOERLI_TESTNET_DEPRECATED (*gnosis.eth.EthereumNetwork attribute*), 97
ZKSYNC_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZKSYNC_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZORA (*gnosis.eth.EthereumNetwork attribute*), 97
ZORA_SEPOLIA_TESTNET (*gnosis.eth.EthereumNetwork attribute*), 97
ZYX_MAINNET (*gnosis.eth.EthereumNetwork attribute*), 97