py-stellar-sdk Documentation

Release 2.2.3

Stellar Community

Mar 12, 2020

Contents

Note: This branch is about v2, click here for the v1 branch. Thank you for your patience as we transition!

py-stellar-sdk is a Python library for communicating with a Stellar Horizon server. It is used for building Stellar apps on Python. It supports **Python 3.6+** as well as PyPy 3.6+.

It provides:

- a networking layer API for Horizon endpoints.
- facilities for building and signing transactions, for communicating with a Stellar Horizon instance, and for submitting transactions or querying network history.

CHAPTER 1

Quickstart

At the absolute basics, you'll want to read up on Stellar's Documentation Guides, as it contains a lot of information on the concepts used below (Transactions, Payments, Operations, KeyPairs, etc.).

1.1 Installation

1.1.1 Via pipenv or pip

To install py-stellar-sdk, use pipenv to install the module:

```
pipenv install stellar-sdk==2.2.3
```

If you're not using pipenv, you should. Otherwise, you can install it via plain old pip. More on installing Python and dependencies can be found over in the Hitchhiker's Guide to Python.

1.1.2 Via Source Code

Please use the code on pypi whenever possible. The latest code may be unstable.

You can clone the repository directly, and install it locally:

```
git clone https://github.com/StellarCN/py-stellar-base.git
cd py-stellar-base
git checkout 2.2.3
pip install .
```

1.2 Generate Keypair

`Keypair` object represents key pair used to sign transactions in Stellar network. `Keypair` object can contain both a public and private key, or only a public key.

If `Keypair` object does not contain private key it can't be used to sign transactions. The most convenient method of creating new keypair is by passing the account's secret seed:

```
from stellar_sdk import Keypair

keypair = Keypair.from_secret("SBK2VIYYSVG76E7VC3QHYARNFLY2EAQXDHRC7BMXBBGIFG74ARPRMNQM")
public_key = keypair.public_key  # GDHMW6QZOL73SHKG2JA3YHXFDHM46SS5ZRWEYF5BCYHX2C5TVO6KZBYL
can_sign = keypair.can_sign()  # True
```

You can create a keypair from public key, but its function is limited:

```
from stellar_sdk import Keypair

keypair = Keypair.from_public_key("GDHMW6QZOL73SHKG2JA3YHXFDHM46SS5ZRWEYF5BCYHX2C5TVO6KZBYL")
can_sign = keypair.can_sign()  # False
```

You can also create a randomly generated keypair:

```
from stellar_sdk import Keypair

keypair = Keypair.random()
print("Public Key: " + keypair.public_key)
print("Secret Seed: " + keypair.secret)
```

1.3 Create Account

Now, in order to create an account, you need to run a `CreateAccount` operation with your new account ID. Due to Stellar's minimum account balance, you'll need to transfer the minimum account balance from another account with the create account operation. As of this writing, minimum balance is **1 XLM (2 x 0.5 Base Reserve)**, and is subject to change.

1.3.1 Using The SDF Testnet

If you want to play in the Stellar test network, you can ask our Friendbot to create an account for you as shown below:

```
import requests

from stellar_sdk import Keypair

keypair = Keypair.random()

print("Public Key: " + keypair.public_key)
print("Secret Seed: " + keypair.secret)

url = 'https://friendbot.stellar.org'
response = requests.get(url, params={'addr': keypair.public_key})
print(response)
```

1.3.2 Using The Stellar Live Network

On the other hand, if you would like to create an account on the live network, you should buy some Stellar Lumens from an exchange. When you withdraw the Lumens into your new account, the exchange will automatically create the account for you. However, if you want to create an account from another account of your own, here's an example of how to do so:

```
from stellar_sdk import TransactionBuilder, Server, Network, Keypair

server = Server(horizon_url="https://horizon-testnet.stellar.org")
source = Keypair.from_secret("SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD")
destination = Keypair.random()

source_account = server.load_account(account_id=source.public_key)
transaction = TransactionBuilder(
    source_account=source_account,
    network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
    base_fee=100) \
    .append_create_account_op(destination=destination.public_key, starting_balance="12.25") \
    .build()
transaction.sign(source)
response = server.submit_transaction(transaction)
print("Transaction hash: {}".format(response["hash"]))
print("New Keypair: \n\taccount id: {account_id}\n\tsecret seed: {secret_seed}".format(
    account_id=destination.public_key, secret_seed=destination.secret))
```

1.4 Querying Horizon

py-stellar-sdk gives you access to all the endpoints exposed by Horizon.

1.4.1 Building requests

py-stellar-sdk uses the Builder pattern to create the requests to send to Horizon. Starting with a `Server` object, you can chain methods together to generate a query. (See the Horizon reference documentation for what methods are possible.)

```
from stellar_sdk import Server

server = Server(horizon_url="https://horizon-testnet.stellar.org")

# get a list of transactions that occurred in ledger 1400
transactions = server.transactions().for_ledger(1400).call()
print(transactions)

# get a list of transactions submitted by a particular account
transactions = server.transactions() \
    .for_account(account_id="GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW") \
    .call()
print(transactions)
```

Once the request is built, it can be invoked with `call()` or with `stream()`. `call()` will return the response given by Horizon.

1.4.2 Streaming requests

Many requests can be invoked with `stream()`. Instead of returning a result like `call()` does, `stream()` will return an EventSource. Horizon will start sending responses from either the beginning of time or from the point specified with `cursor()`. (See the Horizon reference documentation to learn which endpoints support streaming.)

For example, to log instances of transactions from a particular account:

```
from stellar_sdk import Server

server = Server(horizon_url="https://horizon-testnet.stellar.org")
account_id = "GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW"
last_cursor = 'now'  # or load where you left off


def tx_handler(tx_response):
    print(tx_response)


for tx in server.transactions().for_account(account_id).cursor(last_cursor).stream():
    tx_handler(tx)
```

1.5 Assets

Object of the `Asset` class represents an asset in the Stellar network. Right now there are 3 possible types of assets in the Stellar network:

- native **XLM** asset (**ASSET_TYPE_NATIVE**),
- issued assets with asset code of maximum 4 characters (**ASSET_TYPE_CREDIT_ALPHANUM4**),
- issued assets with asset code of maximum 12 characters (**ASSET_TYPE_CREDIT_ALPHANUM12**).

To create a new native asset representation use static `native()` method:

```
from stellar_sdk import Asset
native = Asset.native()
```

To represent an issued asset you need to create a new object of type `Asset` with an asset code and issuer:

```
from stellar_sdk import Asset
# Creates TEST asset issued by␣
↪GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB
test_asset = Asset("TEST", "GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB")
is_native = test_asset.is_native()  # False
# Creates Google stock asset issued by␣
↪GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB
google_stock_asset = Asset('US38259P7069',
↪'GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB')
google_stock_asset_type = google_stock_asset.type  # credit_alphanum12
```

1.6 Building Transactions

Transactions are the commands that modify the state of the ledger. They include sending payments, creating offers, making account configuration changes, etc.

Every transaction has a source account. This is the account that pays the fee and uses up a sequence number for the transaction.

Transactions are made up of one or more operations. Each operation also has a source account, which defaults to the transaction's source account.

1.6.1 TransactionBuilder

The `TransactionBuilder` class is used to construct new transactions. TransactionBuilder is given an account that is used as transaction's **source account**. The transaction will use the current sequence number of the given `Account` object as its sequence number and increments the given account's sequence number when `build()` is called on the TransactionBuilder.

Operations can be added to the transaction calling `append_operation` for each operation you wish to add to the transaction. See *Operation* for a list of possible operations you can add. `append_operation` returns the current `TransactionBuilder` object so you can chain multiple calls. But you don't need to call it directly, we have prepared a lot of easy-to-use functions for you, such as `append_payment_op` can add a payment operation to the `TransactionBuilder`.

After adding the desired operations, call the `build()` method on the `TransactionBuilder`. This will return a fully constructed `TransactionEnvelope`. The transaction object is wrapped in an object called a `TransactionEnvelope`, the returned transaction will contain the sequence number of the source account. This transaction is unsigned. You must sign it before it will be accepted by the Stellar network.

```
from stellar_sdk import TransactionBuilder, Network, Keypair, Account

root_keypair = Keypair.from_secret("SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC")
# Create an Account object from an address and sequence number.
root_account = Account(account_id=root_keypair.public_key, sequence=1)

transaction = TransactionBuilder(
    source_account=root_account,
    # If you want to submit to pubnet, you need to change `network_passphrase` to `Network.PUBLIC_NETWORK_PASSPHRASE`
    network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
    base_fee=100) \
    .append_payment_op(  # add a payment operation to the transaction
    destination="GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW",
    asset_code="XLM",
    amount="125.5") \
    .append_set_options_op(  # add a set options operation to the transaction
    home_domain="overcat.me") \
    .set_timeout(30) \
    .build()  # mark this transaction as valid only for the next 30 seconds
```

1.6.2 Sequence Numbers

The sequence number of a transaction has to match the sequence number stored by the source account or else the transaction is invalid. After the transaction is submitted and applied to the ledger, the source account's sequence

number increases by 1.

There are two ways to ensure correct sequence numbers:

1. Read the source account's sequence number before submitting a transaction
2. Manage the sequence number locally

During periods of high transaction throughput, fetching a source account's sequence number from the network may not return the correct value. So, if you're submitting many transactions quickly, you will want to keep track of the sequence number locally.

1.6.3 Adding Memos

Transactions can contain a **memo** field you can use to attach additional information to the transaction. You can do this by passing a *Memo* object when you construct the TransactionBuilder.

There are 5 types of memos:

- *stellar_sdk.memo.NoneMemo* - empty memo,
- `stellar_sdk.memo.TextMemo` - 28-byte ascii encoded string memo,
- *stellar_sdk.memo.IdMemo*- 64-bit number memo,
- *stellar_sdk.memo.HashMemo* - 32-byte hash - ex. hash of an item in a content server,
- *stellar_sdk.memo.ReturnHashMemo* - 32-byte hash used for returning payments - contains hash of the transaction being rejected.

```
from stellar_sdk import TransactionBuilder, Network, Keypair, Account

root_keypair = Keypair.from_secret("SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC")
# Create an Account object from an address and sequence number.
root_account = Account(account_id=root_keypair.public_key, sequence=1)

transaction = TransactionBuilder(source_account=root_account, network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
                                 base_fee=100).add_text_memo("Happy birthday!").append_payment_op(
    destination="GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW", amount="2000",
    asset_code="XLM").set_timeout(30).build()
```

1.6.4 Transaction and TransactionEnvelope

These two concepts may make the novices unclear, but the official has given a good explanation.

Transactions are commands that modify the ledger state. Among other things, Transactions are used to send payments, enter orders into the distributed exchange, change settings on accounts, and authorize another account to hold your currency. If you think of the ledger as a database, then transactions are SQL commands.

Once a transaction is ready to be signed, the transaction object is wrapped in an object called a Transaction Envelope, which contains the transaction as well as a set of signatures. Most transaction envelopes only contain a single signature along with the transaction, but in multi-signature setups it can contain many signatures. Ultimately, transaction envelopes are passed around the network and are included in transaction sets, as opposed to raw Transaction objects.

1.7 Creating a payment transaction

1.7.1 Payment

In this example, the destination account must exist. We use synchronous methods to submit transactions here, if you want, you can also use asynchronous methods.

```python
"""
Create, sign, and submit a transaction using Python Stellar SDK.

Assumes that you have the following items:
1. Secret key of a funded account to be the source account
2. Public key of an existing account as a recipient
    These two keys can be created and funded by the friendbot at
    https://www.stellar.org/laboratory/ under the heading "Quick Start: Test Account"
3. Access to Python Stellar SDK (https://github.com/StellarCN/py-stellar-base) through Python shell.
"""
from stellar_sdk import Server, Keypair, TransactionBuilder, Network

# The source account is the account we will be signing and sending from.
source_secret_key = "SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD"

# Derive Keypair object and public key (that starts with a G) from the secret
source_keypair = Keypair.from_secret(source_secret_key)
source_public_key = source_keypair.public_key

receiver_public_key = "GA7YNBW5CBTJZ3ZZOWX3ZNBKD6OE7A7IHUQVWMY62W2ZBG2SGZVOOPVH"

# Configure StellarSdk to talk to the horizon instance hosted by Stellar.org
# To use the live network, set the hostname to 'horizon.stellar.org'
server = Server(horizon_url="https://horizon-testnet.stellar.org")

# Transactions require a valid sequence number that is specific to this account.
# We can fetch the current sequence number for the source account from Horizon.
source_account = server.load_account(source_public_key)

base_fee = server.fetch_base_fee()
# we are going to submit the transaction to the test network,
# so network_passphrase is `Network.TESTNET_NETWORK_PASSPHRASE`,
# if you want to submit to the public network, please use `Network.PUBLIC_NETWORK_PASSPHRASE`.
transaction = (
    TransactionBuilder(
        source_account=source_account,
        network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
        base_fee=base_fee,
    )
        .add_text_memo("Hello, Stellar!")  # Add a memo
        # Add a payment operation to the transaction
        # Send 350.1234567 XLM to receiver
        # Specify 350.1234567 lumens. Lumens are divisible to seven digits past the decimal.
        .append_payment_op(receiver_public_key, "350.1234567", "XLM")
        .set_timeout(30)  # Make this transaction valid for the next 30 seconds only
        .build()
)
```

(continues on next page)

(continued from previous page)

```

# Sign this transaction with the secret key
# NOTE: signing is transaction is network specific. Test network transactions
# won't work in the public network. To switch networks, use the Network object
# as explained above (look for stellar_sdk.network.Network).
transaction.sign(source_keypair)

# Let's see the XDR (encoded in base64) of the transaction we just built
print(transaction.to_xdr())

# Submit the transaction to the Horizon server.
# The Horizon server will then submit the transaction into the network for us.
response = server.submit_transaction(transaction)
print(response)
```

1.7.2 Path Payment

In the example below we're sending 1000 XLM (at max) from *GABJLI6IVBKJ7HIC5NN7HHDCIEW3CMWQ2DWYHREQQUFWSWZ2CDAMZZX4* to *GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB*. Destination Asset will be *GBP* issued by *GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW*. Assets will be exchanged using the following path:

- *USD* issued by *GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB*
- *EUR* issued by *GDTNXRLOJD2YEBPKK7KCMR7J33AAG5VZXHAJTHIG736D6LVEFLLLKPDL*

The path payment will cause the destination address to get 5.5 GBP. It will cost the sender no more than 1000 XLM. In this example there will be 3 exchanges, XLM->USD, USD->EUR, EUR->GBP.

```
from stellar_sdk import Keypair, Server, TransactionBuilder, Network, Asset

server = Server(horizon_url="https://horizon-testnet.stellar.org")
source_keypair = Keypair.from_secret("SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC")

source_account = server.load_account(account_id=source_keypair.public_key)

path = [
    Asset("USD", "GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB"),
    Asset("EUR", "GDTNXRLOJD2YEBPKK7KCMR7J33AAG5VZXHAJTHIG736D6LVEFLLLKPDL")
]
transaction = TransactionBuilder(
    source_account=source_account, network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE, base_fee=100) \
    .append_path_payment_op(destination="GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB",
                            send_code="XLM", send_issuer=None, send_max="1000", dest_code="GBP",
                            dest_issuer="GASOCNHNNLYFNMDJYQ3XFMI7BYHIOCFW3GJEOWRPEGK2TDPGTG2E5EDW",
                            dest_amount="5.50", path=path) \
    .set_timeout(30) \
    .build()
transaction.sign(source_keypair)
response = server.submit_transaction(transaction)
```

1.8 Asynchronous

Now we have supported the use of asynchronous methods to submit transactions, py-stellar-sdk gives you the choice, rather than forcing you into always writing async; sync code is easier to write, generally safer, and has many more libraries to choose from.

`Server` has one parameter is **client**, here we need to talk about the **client** parameter, if you do not specify the client, we will use the `RequestsClient` instance by default, it is a synchronous HTTPClient, you can also specify an asynchronous HTTP Client, for example: `AiohttpClient`. If you use a synchronous client, then all requests are synchronous, if you use an asynchronous client, then all requests are asynchronous.

The following is an example of send a payment by an asynchronous method, the same example of using the synchronization method can be found here:

```
"""
The effect of this example is the same as `payment.py`, but this example is asynchronous.

Create, sign, and submit a transaction using Python Stellar SDK.

Assumes that you have the following items:
1. Secret key of a funded account to be the source account
2. Public key of an existing account as a recipient
    These two keys can be created and funded by the friendbot at
    https://www.stellar.org/laboratory/ under the heading "Quick Start: Test Account"
3. Access to Python Stellar SDK (https://github.com/StellarCN/py-stellar-base) through Python shell.
"""
import asyncio

from stellar_sdk import Server, Keypair, TransactionBuilder, Network, AiohttpClient

# The source account is the account we will be signing and sending from.
source_secret_key = "SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD"

# Derive Keypair object and public key (that starts with a G) from the secret
source_keypair = Keypair.from_secret(source_secret_key)
source_public_key = source_keypair.public_key

receiver_public_key = "GA7YNBW5CBTJZ3ZZOWX3ZNBKD6OE7A7IHUQVWMY62W2ZBG2SGZVOOPVH"


async def main():
    # Configure StellarSdk to talk to the horizon instance hosted by Stellar.org
    # To use the live network, set the hostname to 'horizon.stellar.org'
    # When we use the `with` syntax, it automatically releases the resources it occupies.
    async with Server(
            horizon_url="https://horizon-testnet.stellar.org", client=AiohttpClient()
    ) as server:
        # Transactions require a valid sequence number that is specific to this account.
        # We can fetch the current sequence number for the source account from Horizon.
        source_account = await server.load_account(source_public_key)

        base_fee = await server.fetch_base_fee()
```

(continues on next page)

(continued from previous page)

```
        # we are going to submit the transaction to the test network,
        # so network_passphrase is `Network.TESTNET_NETWORK_PASSPHRASE`,
        # if you want to submit to the public network, please use `Network.PUBLIC_
↪NETWORK_PASSPHRASE`.
        transaction = (
            TransactionBuilder(
                source_account=source_account,
                network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
                base_fee=base_fee,
            )
                .add_text_memo("Hello, Stellar!")  # Add a memo
                # Add a payment operation to the transaction
                # Send 350.1234567 XLM to receiver
                # Specify 350.1234567 lumens. Lumens are divisible to seven digits␣
↪past the decimal.
                .append_payment_op(receiver_public_key, "350.1234567", "XLM")
                .set_timeout(30)  # Make this transaction valid for the next 30␣
↪seconds only
                .build()
        )

        # Sign this transaction with the secret key
        # NOTE: signing is transaction is network specific. Test network transactions
        # won't work in the public network. To switch networks, use the Network object
        # as explained above (look for stellar_sdk.network.Network).
        transaction.sign(source_keypair)

        # Let's see the XDR (encoded in base64) of the transaction we just built
        print(transaction.to_xdr())

        # Submit the transaction to the Horizon server.
        # The Horizon server will then submit the transaction into the network for us.
        response = await server.submit_transaction(transaction)
        print(response)


if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
    loop.close()
    # asyncio.run(main())  # Python 3.7+
```

The following example helps you listen to multiple endpoints asynchronously.

```
import asyncio

from stellar_sdk import AiohttpClient, Server

HORIZON_URL = "https://horizon.stellar.org"


async def payments():
    async with Server(HORIZON_URL, AiohttpClient()) as server:
        async for payment in server.payments().cursor(cursor="now").stream():
            print(f"Payment: {payment}")


```

(continues on next page)

(continued from previous page)

```
async def effects():
    async with Server(HORIZON_URL, AiohttpClient()) as server:
        async for effect in server.effects().cursor(cursor="now").stream():
            print(f"Effect: {effect}")


async def operations():
    async with Server(HORIZON_URL, AiohttpClient()) as server:
        async for operation in server.operations().cursor(cursor="now").stream():
            print(f"Operation: {operation}")


async def transactions():
    async with Server(HORIZON_URL, AiohttpClient()) as server:
        async for transaction in server.transactions().cursor(cursor="now").stream():
            print(f"Transaction: {transaction}")


async def listen():
    await asyncio.gather(
        payments(),
        effects(),
        operations(),
        transactions()
    )


if __name__ == '__main__':
    asyncio.run(listen())
```

1.9 Multi-signature account

Multi-signature accounts can be used to require that transactions require multiple public keys to sign before they are considered valid. This is done by first configuring your account's **threshold** levels. Each operation has a threshold level of either low, medium, or high. You give each threshold level a number between 1-255 in your account. Then, for each key in your account, you assign it a weight (1-255, setting a 0 weight deletes the key). Any transaction must be signed with enough keys to meet the threshold.

For example, let's say you set your threshold levels; low = 1, medium = 2, high = 3. You want to send a payment operation, which is a medium threshold operation. Your master key has weight 1. Additionally, you have a secondary key associated with your account which has a weight of 1. Now, the transaction you submit for this payment must include both signatures of your master key and secondary key since their combined weight is 2 which is enough to authorize the payment operation.

In this example, we will:

- Add a second signer to the account
- Set our account's masterkey weight and threshold levels
- Create a multi signature transaction that sends a payment

```
from stellar_sdk import Server, TransactionBuilder, Signer, Network, Keypair

server = Server(horizon_url="https://horizon-testnet.stellar.org")
```

(continues on next page)

(continued from previous page)

```
root_keypair = Keypair.from_secret("SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC")
root_account = server.load_account(account_id=root_keypair.public_key)
secondary_keypair = Keypair.from_secret("SAMZUAAPLRUH62HH3XE7NVD6ZSMTWPWGM6DS4X47HLVRHEBKP4U2H5E7")

secondary_signer = Signer.ed25519_public_key(account_id=secondary_keypair.public_key, weight=1)
transaction = TransactionBuilder(
    source_account=root_account,
    network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
    base_fee=100) \
    .append_set_options_op(
    master_weight=1,  # set master key weight
    low_threshold=1,
    med_threshold=2,  # a payment is medium threshold
    high_threshold=2,  # make sure to have enough weight to add up to the high threshold!
    signer=secondary_signer) \
    .set_timeout(30) \
    .build()

# only need to sign with the root signer as the 2nd signer won't
# be added to the account till after this transaction completes
transaction.sign(root_keypair)
response = server.submit_transaction(transaction)
print(response)

# now create a payment with the account that has two signers
destination = "GBA5SMM5OYAOOPL6R773MV7O3CCLUDVLCWHIVVL3W4XTD3DA5FJ4JSEZ"
transaction = TransactionBuilder(
    source_account=root_account,
    network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
    base_fee=100) \
    .append_payment_op(
    destination=destination,
    amount="2000",
    asset_code="XLM") \
    .set_timeout(30) \
    .build()

# now we need to sign the transaction with both the root and the secondary_keypair
transaction.sign(root_keypair)
transaction.sign(secondary_keypair)
response = server.submit_transaction(transaction)
print(response)
```

CHAPTER 2

API Documentation

Here you'll find detailed documentation on specific functions, classes, and methods.

2.1 API Documentation

2.1.1 Account

class stellar_sdk.account.**Account**(*account_id*, *sequence*)

The *Account* object, which represents represents a single account in Stellar network and its sequence number.

Account tracks the sequence number as it is used by *TransactionBuilder*

See Accounts For more information about the formats used for asset codes and how issuers work on Stellar's network,

Parameters

- **account_id** (str) – Account ID of the account (ex. *GB3KJPLFUYN5VL6R3GU3EGCGVCKFDSD7BEDX42HWG5BWFKB3KQGJJRMA*)
- **sequence** (int) – sequence current sequence number of the account

Raises *Ed25519PublicKeyInvalidError*: if account_id is not a valid ed25519 public key.

increment_sequence_number()

Increments sequence number in this object by one.

Return type None

load_ed25519_public_key_signers()

Load ed25519 public key signers, may change in 3.0.

Return type List[Ed25519PublicKeySigner]

2.1.2 Asset

class `stellar_sdk.asset.`**`Asset`**(*code*, *issuer=None*)

The `Asset` object, which represents an asset and its corresponding issuer on the Stellar network.

For more information about the formats used for asset codes and how issuers work on Stellar's network, see Stellar's guide on assets.

Parameters

- **code** (`str`) – The asset code, in the formats specified in Stellar's guide on assets.
- **issuer** (`Optional[str]`) – The account ID of the issuer. Note if the currency is the native currency (XLM (Lumens)), no issuer is necessary.

Raises

`AssetCodeInvalidError`: if `code` is invalid.

`AssetIssuerInvalidError`: if `issuer` is not a valid ed25519 public key.

classmethod from_xdr_object(*asset_xdr_object*)

Create a `Asset` from an XDR Asset object.

Parameters **asset_xdr_object** (`Asset`) – The XDR Asset object.

Return type `Asset`

Returns A new `Asset` object from the given XDR Asset object.

guess_asset_type()

Return the type of the asset, Can be one of following types: *native*, *credit_alphanum4* or *credit_alphanum12*

Return type `str`

Returns The type of the asset.

is_native()

Return true if the `Asset` is the native asset.

Return type `bool`

Returns True if the Asset is native, False otherwise.

static native()

Returns an asset object for the native asset.

Return type `Asset`

Returns An asset object for the native asset.

to_dict()

Generate a dict for this object's attributes.

Return type `dict`

Returns A dict representing an `Asset`

to_xdr_object()

Returns the xdr object for this asset.

Return type `Asset`

Returns XDR Asset object

type

Return the type of the asset, Can be one of following types: *native*, *credit_alphanum4* or *credit_alphanum12*

Return type `str`

Returns The type of the asset.

2.1.3 Call Builder

BaseCallBuilder

class `stellar_sdk.call_builder.`**`BaseCallBuilder`**(*horizon_url*, *client*)

Creates a new `BaseCallBuilder` pointed to server defined by horizon_url.

This is an **abstract** class. Do not create this object directly, use `stellar_sdk.server.Server` class.

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union`[`BaseAsyncClient`, `BaseSyncClient`]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union`[`Dict`[`str`, `Any`], `Coroutine`[`Any`, `Any`, `Dict`[`str`, `Any`]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (`bool`) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

AccountsCallBuilder

class stellar_sdk.call_builder.**AccountsCallBuilder**(*horizon_url*, *client*)

Creates a new `AccountsCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.accounts()`.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.

account_id(*account_id*)

Returns information and links relating to a single account. The balances section in the returned JSON will also list all the trust lines this account has set up.

See Account Details

Parameters **account_id** (`str`) – account id, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*

Return type `AccountsCallBuilder`

Returns current AccountCallBuilder instance

asset(*asset*)

Filtering accounts who have a trustline to an asset. The result is a list of accounts.

See Account Details

Parameters **asset** (`Asset`) – an issued asset

Return type `AccountsCallBuilder`

Returns current AccountCallBuilder instance

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

for_asset(*asset*)

Filtering accounts who have a trustline to an asset. The result is a list of accounts.

See Account Details

Parameters **asset** (*Asset*) – an issued asset

Return type `AccountsCallBuilder`

Returns current AccountCallBuilder instance

for_signer(*signer*)

Filtering accounts who have a given signer. The result is a list of accounts.

See Account Details

Parameters **signer** (`str`) – signer's account id, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*

Return type `AccountsCallBuilder`

Returns current AccountCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

signer(*signer*)

Filtering accounts who have a given signer. The result is a list of accounts.

See Account Details

Parameters **signer** (str) – signer's account id, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*

Return type AccountsCallBuilder

Returns current AccountCallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

AssetsCallBuilder

class stellar_sdk.call_builder.**AssetsCallBuilder**(*horizon_url*, *client*)

Creates a new *AssetsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.assets()*.

See All Assets

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

for_code(*asset_code*)

This endpoint filters all assets by the asset code.

Parameters **asset_code** (`str`) – asset code, for example: *USD*

Return type `AssetsCallBuilder`

Returns current AssetCallBuilder instance

for_issuer(*asset_issuer*)

This endpoint filters all assets by the asset issuer.

Parameters **asset_issuer** (`str`) – asset issuer, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*

Return type `AssetsCallBuilder`

Returns current AssetCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (`bool`) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

DataCallBuilder

class stellar_sdk.call_builder.**DataCallBuilder**(*horizon_url*, *client*, *account_id*, *data_name*)

Creates a new `DataCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.data()`.

See Data for Account

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.
- **account_id** (`str`) – account id, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*
- **data_name** (`str`) – Key name

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (`bool`) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

EffectsCallBuilder

class stellar_sdk.call_builder.**EffectsCallBuilder**(*horizon_url*, *client*)

Creates a new *`EffectsCallBuilder`* pointed to server defined by horizon_url. Do not create this object directly, use *`stellar_sdk.server.Server.effects()`*.

See All Effects

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

for_account(*account_id*)

This endpoint represents all effects that changed a given account. It will return relevant effects from the creation of the account to the current ledger.

See Effects for Account

Parameters **account_id** (`str`) – account id, for example: *GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD*

Return type `EffectsCallBuilder`

Returns this EffectCallBuilder instance

for_ledger(*sequence*)

Effects are the specific ways that the ledger was changed by any operation. This endpoint represents all effects that occurred in the given ledger.

See Effects for Ledger

Parameters **sequence** (`Union[int, str]`) – ledger sequence

Return type `EffectsCallBuilder`

Returns this EffectCallBuilder instance

for_operation(*operation_id*)

This endpoint represents all effects that occurred as a result of a given operation.

See Effects for Operation

Parameters **operation_id** (`Union[int, str]`) – operation ID

Return type `EffectsCallBuilder`

Returns this EffectCallBuilder instance

for_transaction(*transaction_hash*)

This endpoint represents all effects that occurred as a result of a given transaction.

See Effects for Transaction

Parameters **transaction_hash** (`str`) – transaction hash

Return type `EffectsCallBuilder`

Returns this EffectCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

FeeStatsCallBuilder

class stellar_sdk.call_builder.**FeeStatsCallBuilder**(*horizon_url*, *client*)

Creates a new *FeeStatsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.fee_stats()*.

See Fee Stats

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit (*limit*)
Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order (*desc=True*)
Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (`bool`) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream ()
Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

LedgersCallBuilder

class stellar_sdk.call_builder.**LedgersCallBuilder** (*horizon_url*, *client*)
Creates a new *`LedgersCallBuilder`* pointed to server defined by horizon_url. Do not create this object directly, use *`stellar_sdk.server.Server.ledgers()`*.

See All Ledgers

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[`*`BaseAsyncClient`*`, `*`BaseSyncClient`*`]`) – The client instance used to send request.

call ()
Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404

`BadResponseError`: if 500 <= status_code < 600

`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

ledger(*sequence*)

Provides information on a single ledger.

See Ledger Details

Parameters **sequence** (`Union[int, str]`) – Ledger sequence

Return type `LedgersCallBuilder`

Returns current LedgerCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (`bool`) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

OffersCallBuilder

class stellar_sdk.call_builder.**OffersCallBuilder**(*horizon_url*, *client*)

Creates a new *OffersCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.offers()*.

See Offer Details See Offers

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

account(*account_id*)

Returns all offers where the given account is the seller.

See Offers for Account

Parameters **account_id** – Account ID

Returns this OffersCallBuilder instance

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

for_buying(*buying*)

Returns all offers buying an asset.

People on the Stellar network can make offers to buy or sell assets. This endpoint represents all the current offers, allowing filtering by *seller*, *selling_asset* or *buying_asset*.

See Offers

Parameters **buying** (*Asset*) – The asset being bought.

Returns this OffersCallBuilder instance

for_seller(*seller*)

Returns all offers where the given account is the seller.

People on the Stellar network can make offers to buy or sell assets. This endpoint represents all the current offers, allowing filtering by *seller*, *selling_asset* or *buying_asset*.

See Offers

Parameters **seller** (str) – Account ID of the offer creator

Returns this OffersCallBuilder instance

for_selling(*selling*)

Returns all offers selling an asset.

People on the Stellar network can make offers to buy or sell assets. This endpoint represents all the current offers, allowing filtering by *seller*, *selling_asset* or *buying_asset*.

See Offers

Parameters **selling** (*Asset*) – The asset being sold.

Returns this OffersCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

offer(*offer_id*)

Returns information and links relating to a single offer.

See Offer Details

Parameters **offer_id** (Union[str, int]) – Offer ID.

Returns this OffersCallBuilder instance

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, `True` to get desc sort direction, the default setting is `True`.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

OperationsCallBuilder

class `stellar_sdk.call_builder.`**`OperationsCallBuilder`**(*horizon_url*, *client*)

Creates a new `OperationsCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.operations()`.

See All Operations

Parameters

- **horizon_url** – Horizon server URL.
- **client** (`Union`[`BaseAsyncClient`, `BaseSyncClient`]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union`[`Dict`[`str`, `Any`], `Coroutine`[`Any`, `Any`, `Dict`[`str`, `Any`]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

for_account(*account_id*)

This endpoint represents all operations that were included in valid transactions that affected a particular account.

See Operations for Account

Parameters **account_id** (`str`) – Account ID

Return type `OperationsCallBuilder`

Returns this OperationCallBuilder instance

for_ledger(*sequence*)

This endpoint returns all operations that occurred in a given ledger.

See Operations for Ledger

Parameters **sequence** (Union[int, str]) – Sequence ID

Return type OperationsCallBuilder

Returns this OperationCallBuilder instance

for_transaction(*transaction_hash*)

This endpoint represents all operations that are part of a given transaction.

See Operations for Transaction

Parameters **transaction_hash** (str) –

Return type OperationsCallBuilder

Returns this OperationCallBuilder instance

include_failed(*include_failed*)

Adds a parameter defining whether to include failed transactions. By default only operations of successful transactions are returned.

Parameters **include_failed** (bool) – Set to *True* to include operations of failed transactions.

Return type OperationsCallBuilder

Returns current OperationsCallBuilder instance

join(*join*)

join represents *join* param in queries, currently only supports *transactions*

Parameters **join** (str) – join represents *join* param in queries, currently only supports *transactions*

Return type OperationsCallBuilder

Returns current OperationsCallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

operation(*operation_id*)

The operation details endpoint provides information on a single operation. The operation ID provided in the id argument specifies which operation to load.

See Operation Details

Parameters **operation_id** (Union[int, str]) – Operation ID

Return type OperationsCallBuilder

Returns this OperationCallBuilder instance

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

OrderbookCallBuilder

class stellar_sdk.call_builder.**OrderbookCallBuilder**(*horizon_url*, *client*, *selling*, *buying*)

Creates a new *OrderbookCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.orderbook()*.

See Orderbook Details

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.
- **selling** (*Asset*) – Asset being sold
- **buying** (*Asset*) – Asset being bought

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

PathsCallBuilder

class stellar_sdk.call_builder.**PathsCallBuilder**(*horizon_url*, *client*, *source_account*, *destination_account*, *destination_asset*, *destination_amount*)

Creates a new *PathsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.paths()*.

The Stellar Network allows payments to be made across assets through path payments. A path payment specifies a series of assets to route a payment through, from source asset (the asset debited from the payer) to destination asset (the asset credited to the payee).

See Find Payment Paths

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

- **source_account** (str) – The sender's account ID. Any returned path must use a source that the sender can hold.
- **destination_account** (str) – The destination account ID that any returned path should use.
- **destination_asset** (Asset) – The destination asset.
- **destination_amount** (str) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

PaymentsCallBuilder

class stellar_sdk.call_builder.**PaymentsCallBuilder**(*horizon_url*, *client*)

Creates a new *PaymentsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.payments()*.

See All Payments

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

for_account(*account_id*)

This endpoint responds with a collection of Payment operations where the given account was either the sender or receiver.

See Payments for Account

Parameters **account_id** (str) – Account ID

Return type PaymentsCallBuilder

Returns current PaymentsCallBuilder instance

for_ledger(*sequence*)

This endpoint represents all payment operations that are part of a valid transactions in a given ledger.

See Payments for Ledger

Parameters **sequence** (Union[int, str]) – Ledger sequence

Return type PaymentsCallBuilder

Returns current PaymentsCallBuilder instance

for_transaction(*transaction_hash*)

This endpoint represents all payment operations that are part of a given transaction.

See Payments for Transaction

Parameters **transaction_hash** (str) – Transaction hash

Return type PaymentsCallBuilder

Returns current PaymentsCallBuilder instance

include_failed(*include_failed*)

Adds a parameter defining whether to include failed transactions. By default only payments of successful transactions are returned.

Parameters **include_failed** (bool) – Set to True to include payments of failed transactions.

Return type PaymentsCallBuilder

Returns current PaymentsCallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type `Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]`

Returns If it is called synchronous, it will return `Generator`, If it is called asynchronously, it will return `AsyncGenerator`.

RootCallBuilder

class `stellar_sdk.call_builder.`**`RootCallBuilder`**(*horizon_url*, *client*)

Creates a new `RootCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.root()`.

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

limit(*limit*)

Sets `limit` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (`int`) – Number of records the server should return.

Return type `BaseCallBuilder`

Returns

order(*desc=True*)

Sets `order` parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

StrictReceivePathsCallBuilder

class stellar_sdk.call_builder.**StrictReceivePathsCallBuilder**(*horizon_url*, *client*, *source*, *destination_asset*, *destination_amount*)

Creates a new *StrictReceivePathsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.strict_receive_paths()*.

The Stellar Network allows payments to be made across assets through path payments. A path payment specifies a series of assets to route a payment through, from source asset (the asset debited from the payer) to destination asset (the asset credited to the payee).

A path search is specified using:

- The source address or source assets.
- The asset and amount that the destination account should receive.

As part of the search, horizon will load a list of assets available to the source address and will find any payment paths from those source assets to the desired destination asset. The search's amount parameter will be used to determine if there a given path can satisfy a payment of the desired amount.

If a list of assets is passed as the source, horizon will find any payment paths from those source assets to the desired destination asset.

See Find Payment Paths

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.
- **source** (Union[str, List[*Asset*]]) – The sender's account ID or a list of Assets. Any returned path must use a source that the sender can hold.
- **destination_asset** (*Asset*) – The destination asset.
- **destination_amount** (str) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

StrictSendPathsCallBuilder

class stellar_sdk.call_builder.**StrictSendPathsCallBuilder**(*horizon_url*, *client*, *source_asset*, *source_amount*, *destination*)

Creates a new `StrictSendPathsCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.strict_send_paths()`.

The Stellar Network allows payments to be made across assets through path payments. A strict send path payment specifies a series of assets to route a payment through, from source asset (the asset debited from the payer) to destination asset (the asset credited to the payee).

A strict send path search is specified using:

- The source asset
- The source amount
- The destination assets or destination account.

As part of the search, horizon will load a list of assets available to the source address and will find any payment paths from those source assets to the desired destination asset. The search's source_amount parameter will be used to determine if there a given path can satisfy a payment of the desired amount.

See Find Payment Paths

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.
- **source_asset** (`Asset`) – The asset to be sent.
- **source_amount** (`str`) – The amount, denominated in the source asset, that any returned path should be able to satisfy.
- **destination** (`Union[str, List[Asset]]`) – The destination account or the destination assets.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

TradeAggregationsCallBuilder

class stellar_sdk.call_builder.**TradeAggregationsCallBuilder**(*horizon_url, client, base, counter, resolution, start_time=None, end_time=None, offset=None*)

Creates a new *TradeAggregationsCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.trade_aggregations()*.

Trade Aggregations facilitate efficient gathering of historical trade data.

See Trade Aggregations

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

- **base** (*Asset*) – base asset
- **counter** (*Asset*) – counter asset
- **resolution** (int) – segment duration as millis since epoch. *Supported values are 1 minute (60000), 5 minutes (300000), 15 minutes (900000), 1 hour (3600000), 1 day (86400000) and 1 week (604800000).*
- **start_time** (Optional[int]) – lower time boundary represented as millis since epoch
- **end_time** (Optional[int]) – upper time boundary represented as millis since epoch
- **offset** (Optional[int]) – segments can be offset using this parameter. Expressed in milliseconds. *Can only be used if the resolution is greater than 1 hour. Value must be in whole hours, less than the provided resolution, and less than 24 hours.*

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

TradesCallBuilder

class stellar_sdk.call_builder.**TradesCallBuilder**(*horizon_url*, *client*)

Creates a new *TradesCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use *stellar_sdk.server.Server.trades()*.

See Trades

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*]) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

ConnectionError: if you have not successfully connected to the server.
NotFoundError: if status_code == 404
BadRequestError: if 400 <= status_code < 500 and status_code != 404
BadResponseError: if 500 <= status_code < 600
UnknownRequestError: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets cursor parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (Union) – A cursor is a value that points to a specific location in a collection of resources.

Return type BaseCallBuilder

Returns current CallBuilder instance

for_account(*account_id*)

Filter trades for a specific account

See Trades for Account

Parameters **account_id** (str) – account id

Return type TradesCallBuilder

Returns current TradesCallBuilder instance

for_asset_pair(*base*, *counter*)

Filter trades for a specific asset pair (orderbook)

Parameters

- **base** (*Asset*) – base asset
- **counter** (*Asset*) – counter asset

Return type TradesCallBuilder

Returns current TradesCallBuilder instance

for_offer(*offer_id*)

Filter trades for a specific offer

See Trades for Offer

Parameters **offer_id** (Union[int, str]) – offer id

Return type TradesCallBuilder

Returns current TradesCallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

TransactionsCallBuilder

class stellar_sdk.call_builder.**TransactionsCallBuilder**(*horizon_url*, *client*)

Creates a new `TransactionsCallBuilder` pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.server.Server.transactions()`.

See All Transactions

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (`Union[BaseAsyncClient, BaseSyncClient]`) – The client instance used to send request.

call()

Triggers a HTTP request using this builder's current configuration.

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns If it is called synchronous, the response will be returned. If it is called asynchronously, it will return Coroutine.

Raises

`ConnectionError`: if you have not successfully connected to the server.
`NotFoundError`: if status_code == 404
`BadRequestError`: if 400 <= status_code < 500 and status_code != 404
`BadResponseError`: if 500 <= status_code < 600
`UnknownRequestError`: if an unknown error occurs, please submit an issue

cursor(*cursor*)

Sets `cursor` parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **cursor** (`Union`) – A cursor is a value that points to a specific location in a collection of resources.

Return type `BaseCallBuilder`

Returns current CallBuilder instance

for_account(*account_id*)

This endpoint represents all transactions that affected a given account.

See Transactions for Account

Parameters **account_id** (`str`) – account id

Return type `TransactionsCallBuilder`

Returns current TransactionsCallBuilder instance

for_ledger(*sequence*)

This endpoint represents all transactions in a given ledger.

See Transactions for Ledger

Parameters **sequence** (`Union[str, int]`) – ledger sequence

Return type `TransactionsCallBuilder`

Returns current TransactionsCallBuilder instance

include_failed(*include_failed*)

Adds a parameter defining whether to include failed transactions. By default only transactions of successful transactions are returned.

Parameters **include_failed** (bool) – Set to *True* to include failed transactions.

Return type TransactionsCallBuilder

Returns current TransactionsCallBuilder instance

limit(*limit*)

Sets limit parameter for the current call. Returns the CallBuilder object on which this method has been called.

See Paging

Parameters **limit** (int) – Number of records the server should return.

Return type BaseCallBuilder

Returns

order(*desc=True*)

Sets order parameter for the current call. Returns the CallBuilder object on which this method has been called.

Parameters **desc** (bool) – Sort direction, True to get desc sort direction, the default setting is True.

Return type BaseCallBuilder

Returns current CallBuilder instance

stream()

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Return type Union[AsyncGenerator[Dict[str, Any], None], Generator[Dict[str, Any], None, None]]

Returns If it is called synchronous, it will return Generator, If it is called asynchronously, it will return AsyncGenerator.

transaction(*transaction_hash*)

The transaction details endpoint provides information on a single transaction. The transaction hash provided in the hash argument specifies which transaction to load.

See Transaction Details

Parameters **transaction_hash** (str) – transaction hash

Return type TransactionsCallBuilder

Returns current TransactionsCallBuilder instance

2.1.4 Client

BaseAsyncClient

class `stellar_sdk.client.base_async_client.`**`BaseAsyncClient`**

This is an abstract class, and if you want to implement your own asynchronous client, you **must** implement this class.

get(*url*, *params=None*)

Perform HTTP GET request.

Parameters

- **url** (`str`) – the request url
- **params** (`Optional[Dict[str, str]]`) – the request params

Return type *`Response`*

Returns the response from server

Raise *`ConnectionError`*

post(*url*, *data*)

Perform HTTP POST request.

Parameters

- **url** (`str`) – the request url
- **data** (`Dict[str, str]`) – the data send to server

Return type *`Response`*

Returns the response from server

Raise *`ConnectionError`*

stream(*url*, *params=None*)

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Parameters

- **url** (`str`) – the request url
- **params** (`Optional[Dict[str, str]]`) – the request params

Return type `AsyncGenerator[Dict[str, Any], None]`

Returns a dict AsyncGenerator for server response

Raise *`ConnectionError`*

BaseSyncClient

class `stellar_sdk.client.base_sync_client.`**`BaseSyncClient`**

This is an abstract class, and if you want to implement your own synchronous client, you **must** implement this class.

get(*url*, *params=None*)

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type *Response*

Returns the response from server

Raise *ConnectionError*

post (*url*, *data*)

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (Dict[str, str]) – the data send to server

Return type *Response*

Returns the response from server

Raise *ConnectionError*

stream (*url*, *params=None*)

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type Generator[Dict[str, Any], None, None]

Returns a dict Generator for server response

Raise *ConnectionError*

AiohttpClient

class stellar_sdk.client.aiohttp_client.**AiohttpClient** (*pool_size=None*, *request_timeout=20*, *backoff_factor=0.5*, *user_agent=None*, ***kwargs*)

The *AiohttpClient* object is a asynchronous http client, which represents the interface for making requests to a server instance.

Parameters

- **pool_size** (Optional[int]) – persistent connection to Horizon and connection pool
- **request_timeout** (float) – the timeout for all requests
- **backoff_factor** (Optional[float]) – a backoff factor to apply between attempts after the second try
- **user_agent** (Optional[str]) – the server can use it to identify you

close()
Close underlying connector.

Release all acquired resources.

Return type None

get(*url*, *params=None*)
Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the requested params

Return type *Response*

Returns the response from server

Raise *ConnectionError*

post(*url*, *data=None*)
Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (Optional[Dict[str, str]]) – the data send to server

Return type *Response*

Returns the response from server

Raise *ConnectionError*

stream(*url*, *params=None*)
Init the sse session

Return type AsyncGenerator[Dict[str, Any], None]

RequestsClient

class stellar_sdk.client.requests_client.**RequestsClient**(*pool_size=10*, *num_retries=3*, *request_timeout=20*, *backoff_factor=0.5*, *session=None*, *stream_session=None*)
The *RequestsClient* object is a synchronous http client, which represents the interface for making requests to a server instance.

Parameters

- **pool_size** (int) – persistent connection to Horizon and connection pool
- **num_retries** (int) – configurable request retry functionality
- **request_timeout** (int) – the timeout for all requests
- **backoff_factor** (float) – a backoff factor to apply between attempts after the second try
- **session** (Optional[Session]) – the request session

- **stream_session** (Optional[Session]) – the stream request session

close()

Close underlying connector.

Release all acquired resources.

Return type None

get(*url*, *params=None*)

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type *Response*

Returns the response from server

Raise *ConnectionError*

post(*url*, *data=None*)

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (Optional[Dict[str, str]]) – the data send to server

Return type *Response*

Returns the response from server

Raise *ConnectionError*

stream(*url*, *params=None*)

Creates an EventSource that listens for incoming messages from the server.

See Horizon Response Format

See MDN EventSource

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type Generator[Dict[str, Any], None, None]

Returns a Generator for server response

Raise *ConnectionError*

SimpleRequestsClient

class stellar_sdk.client.simple_requests_client.**SimpleRequestsClient**

The *SimpleRequestsClient* object is a synchronous http client, which represents the interface for making requests to a server instance.

This client is to guide you in writing a client that suits your needs. I don't recommend that you actually use it.

get (*url*, *params=None*)
Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type *Response*

Returns the response from server

Raise *ConnectionError*

post (*url*, *data*)
Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (Dict[str, str]) – the data send to server

Return type *Response*

Returns the response from server

Raise *ConnectionError*

stream (*url*, *params=None*)
Not Implemented

Parameters

- **url** (str) – the request url
- **params** (Optional[Dict[str, str]]) – the request params

Return type None

Returns None

Response

class stellar_sdk.client.response.**Response** (*status_code*, *text*, *headers*, *url*)
The *Response* object, which contains a server's response to an HTTP request.

Parameters

- **status_code** (int) – response status code
- **text** (str) – response content
- **headers** (dict) – response headers
- **url** (str) – request url

json ()
convert the content to dict

Return type dict

Returns the content from server

2.1.5 Exceptions

SdkError

class stellar_sdk.exceptions.**SdkError**
Base exception for all stellar sdk related errors

ValueError

class stellar_sdk.exceptions.**ValueError**
exception for all values related errors

TypeError

class stellar_sdk.exceptions.**TypeError**
exception for all type related errors

BadSignatureError

class stellar_sdk.exceptions.**BadSignatureError**
Raised when the signature was forged or otherwise corrupt.

Ed25519PublicKeyInvalidError

class stellar_sdk.exceptions.**Ed25519PublicKeyInvalidError**
Ed25519 public key is incorrect.

Ed25519SecretSeedInvalidError

class stellar_sdk.exceptions.**Ed25519SecretSeedInvalidError**
Ed25519 secret seed is incorrect.

MissingEd25519SecretSeedError

class stellar_sdk.exceptions.**MissingEd25519SecretSeedError**
Missing Ed25519 secret seed in the keypair

MemoInvalidException

class stellar_sdk.exceptions.**MemoInvalidException**
Memo is incorrect.

AssetCodeInvalidError

class stellar_sdk.exceptions.**AssetCodeInvalidError**
Asset Code is incorrect.

AssetIssuerInvalidError

class stellar_sdk.exceptions.**AssetIssuerInvalidError**
Asset issuer is incorrect.

NoApproximationError

class stellar_sdk.exceptions.**NoApproximationError**
Approximation cannot be found

SignatureExistError

class stellar_sdk.exceptions.**SignatureExistError**
A keypair can only sign a transaction once.

BaseRequestError

class stellar_sdk.exceptions.**BaseRequestError**
Base class for requests errors.

ConnectionError

class stellar_sdk.exceptions.**ConnectionError**
Base class for client connection errors.

BaseHorizonError

class stellar_sdk.exceptions.**BaseHorizonError**(*response*)
Base class for horizon request errors.

Parameters **response** (*Response*) – client response

NotFoundError

class stellar_sdk.exceptions.**NotFoundError**(*response*)
This exception is thrown when the requested resource does not exist. status_code == 400

BadRequestError

class stellar_sdk.exceptions.**BadRequestError**(*response*)
The request from the client has an error. 400 <= status_code < 500 and status_code != 404

BadResponseError

class stellar_sdk.exceptions.**BadResponseError**(*response*)
The response from the server has an error. 500 <= status_code < 600

2.1.6 Keypair

class stellar_sdk.keypair.**Keypair**(*verify_key*, *signing_key=None*)

The `Keypair` object, which represents a signing and verifying key for use with the Stellar network.

Instead of instantiating the class directly, we recommend using one of several class methods:

- `Keypair.random()`
- `Keypair.from_secret()`
- `Keypair.from_public_key()`

Parameters

- **verify_key** (`VerifyKey`) – The verifying (public) Ed25519 key in the keypair.
- **signing_key** (`Optional[SigningKey]`) – The signing (private) Ed25519 key in the keypair.

can_sign()

Returns *True* if this `Keypair` object contains secret key and can sign.

Return type `bool`

Returns *True* if this `Keypair` object contains secret key and can sign

classmethod from_mnemonic_phrase(*mnemonic_phrase*, *passphrase=''*, *index=0*)

Generate a `Keypair` object via a mnemonic phrase.

Parameters

- **mnemonic_phrase** (`str`) – A unique string used to deterministically generate keypairs.
- **passphrase** (`str`) – An optional passphrase used as part of the salt during PBKDF2 rounds when generating the seed from the mnemonic.
- **index** (`int`) – The index of the keypair generated by the mnemonic. This allows for multiple Keypairs to be derived from the same mnemonic, such as:

```
>>> from stellar_sdk.keypair import Keypair
>>> mnemonic = 'update hello cry airport drive chunk elite boat␣
↪shaft sea describe number'  # Don't use this mnemonic in␣
↪practice.
>>> kp1 = Keypair.from_mnemonic_phrase(mnemonic, index=0)
>>> kp2 = Keypair.from_mnemonic_phrase(mnemonic, index=1)
>>> kp3 = Keypair.from_mnemonic_phrase(mnemonic, index=2)
```

Returns A new `Keypair` instance derived from the mnemonic.

classmethod from_public_key(*public_key*)

Generate a `Keypair` object from a public key.

Parameters **public_key** (`str`) – strkey ed25519 public key, for example: *GATPGGOIE6VWADVKD3ER3IFO2IH6DTOA5G535ITB3TT66FZFSIZEAU2B*

Return type `Keypair`

Returns A new `Keypair` instance derived by the public key.

Raise `Ed25519PublicKeyInvalidError`: if `public_key` is not a valid ed25519 public key.

classmethod from_raw_ed25519_public_key(*raw_public_key*)
Generate a `Keypair` object from ed25519 public key raw bytes.

Parameters **raw_public_key** (`bytes`) – ed25519 public key raw bytes

Return type `Keypair`

Returns A new `Keypair` instance derived by the ed25519 public key raw bytes

classmethod from_raw_ed25519_seed(*raw_seed*)
Generate a `Keypair` object from ed25519 secret key seed raw bytes.

Parameters **raw_seed** (`bytes`) – ed25519 secret key seed raw bytes

Return type `Keypair`

Returns A new `Keypair` instance derived by the ed25519 secret key seed raw bytes

classmethod from_secret(*secret*)
Generate a `Keypair` object from a secret seed.

Parameters **secret** (`str`) – strkey ed25519 seed, for example: *SB2LHKBL24ITV2Y346BU46XPEL45BDAFOOJLZ6SESCJZ6V5JMP7D6G5X*

Return type `Keypair`

Returns A new `Keypair` instance derived by the secret.

Raise `Ed25519SecretSeedInvalidError`: if `secret` is not a valid ed25519 secret seed.

static generate_mnemonic_phrase(*language=<Language.ENGLISH: 'english'>, strength=128*)
Generate a mnemonic phrase.

Parameters

- **language** (`Union[Language, str]`) – The language of the mnemonic phrase, defaults to english.
- **strength** (`int`) – The complexity of the mnemonic.

Returns A mnemonic phrase.

public_key
Returns public key associated with this `Keypair` instance

Return type `str`

Returns public key

classmethod random()
Generate a `Keypair` object from a randomly generated seed.

Return type `Keypair`

Returns A new `Keypair` instance derived by the randomly seed.

raw_public_key()
Returns raw public key.

Return type `bytes`

Returns raw public key

raw_secret_key()
Returns raw secret key.

Return type `bytes`

Returns raw secret key

secret

Returns secret key associated with this *Keypair* instance

Return type `str`

Returns secret key

Raise *MissingEd25519SecretSeedError* The *Keypair* does not contain secret seed

sign(*data*)

Sign the provided data with the keypair's private key.

Parameters **data** (`bytes`) – The data to sign.

Return type `bytes`

Returns signed bytes

Raise *MissingEd25519SecretSeedError*: if *Keypair* does not contain secret seed.

sign_decorated(*data*)

Sign the provided data with the keypair's private key and returns DecoratedSignature.

Parameters **data** – signed bytes

Return type `DecoratedSignature`

Returns sign decorated

signature_hint()

Returns signature hint associated with this *Keypair* instance

Return type `bytes`

Returns signature hint

verify(*data*, *signature*)

Verify the provided data and signature match this keypair's public key.

Parameters

- **data** (`bytes`) – The data that was signed.
- **signature** (`bytes`) – The signature.

Raise *BadSignatureError*: if the verification failed and the signature was incorrect.

Return type `None`

xdr_public_key()

Return type `PublicKey`

Returns xdr public key

2.1.7 Memo

Memo

class `stellar_sdk.memo.`**Memo**

The *Memo* object, which represents the base class for memos for use with Stellar transactions.

The memo for a transaction contains optional extra information about the transaction taking place. It is the responsibility of the client to interpret this value.

See the following implementations that serve a more practical use with the library:

- *NoneMemo* - No memo.
- *TextMemo* - A string encoded using either ASCII or UTF-8, up to 28-bytes long.
- *IdMemo* - A 64 bit unsigned integer.
- *HashMemo* - A 32 byte hash.
- `RetHashMemo` - A 32 byte hash intended to be interpreted as the hash of the transaction the sender is refunding.

See Stellar's documentation on Transactions for more information on how memos are used within transactions, as well as information on the available types of memos.

static from_xdr_object(*xdr_obj*)
Returns an Memo object from XDR memo object.

Return type *Memo*

to_xdr_object()
Creates an XDR Memo object that represents this *Memo*.

Return type `Memo`

NoneMemo

class stellar_sdk.memo.**NoneMemo**
The *NoneMemo*, which represents no memo for a transaction.

classmethod from_xdr_object(*xdr_obj*)
Returns an *NoneMemo* object from XDR memo object.

Return type *NoneMemo*

to_xdr_object()
Creates an XDR Memo object that represents this *NoneMemo*.

Return type `Memo`

TextMemo

class stellar_sdk.memo.**TextMemo**(*text*)
The *TextMemo*, which represents MEMO_TEXT in a transaction.

Parameters **text** (*str, bytes*) – A string encoded using either ASCII or UTF-8, up to 28-bytes long.

Raises *MemoInvalidException*: if `text` is not a valid text memo.

classmethod from_xdr_object(*xdr_obj*)
Returns an *TextMemo* object from XDR memo object.

Return type *TextMemo*

to_xdr_object()
Creates an XDR Memo object that represents this *TextMemo*.

Return type `Memo`

IdMemo

class stellar_sdk.memo.**IdMemo**(*memo_id*)

The *IdMemo* which represents MEMO_ID in a transaction.

Parameters **memo_id** (*int*) – A 64 bit unsigned integer.

Raises *MemoInvalidException*: if id is not a valid id memo.

classmethod from_xdr_object(*xdr_obj*)

Returns an *IdMemo* object from XDR memo object.

Return type *IdMemo*

to_xdr_object()

Creates an XDR Memo object that represents this *IdMemo*.

Return type Memo

HashMemo

class stellar_sdk.memo.**HashMemo**(*memo_hash*)

The *HashMemo* which represents MEMO_HASH in a transaction.

Parameters **memo_hash** (Union[bytes, str]) – A 32 byte hash hex encoded string.

Raises *MemoInvalidException*: if memo_hash is not a valid hash memo.

classmethod from_xdr_object(*xdr_obj*)

Returns an *HashMemo* object from XDR memo object.

Return type *HashMemo*

to_xdr_object()

Creates an XDR Memo object that represents this *HashMemo*.

Return type Memo

ReturnHashMemo

class stellar_sdk.memo.**ReturnHashMemo**(*memo_return*)

The *ReturnHashMemo* which represents MEMO_RETURN in a transaction.

MEMO_RETURN is typically used with refunds/returns over the network - it is a 32 byte hash intended to be interpreted as the hash of the transaction the sender is refunding.

Parameters **memo_return** (bytes) – A 32 byte hash or hex encoded string intended to be interpreted as the hash of the transaction the sender is refunding.

Raises *MemoInvalidException*: if memo_return is not a valid return hash memo.

classmethod from_xdr_object(*xdr_obj*)

Returns an *ReturnHashMemo* object from XDR memo object.

Return type *ReturnHashMemo*

to_xdr_object()

Creates an XDR Memo object that represents this *ReturnHashMemo*.

Return type Memo

2.1.8 Network

class stellar_sdk.network.**Network**(*network_passphrase*)

The *Network* object, which represents a Stellar network.

This class represents such a stellar network such as the Public network and the Test network.

Parameters **network_passphrase** (*str*) – The passphrase for the network. (ex. 'Public Global Stellar Network ; September 2015')

PUBLIC_NETWORK_PASSPHRASE = 'Public Global Stellar Network ; September 2015'

Get the Public network passphrase.

TESTNET_NETWORK_PASSPHRASE = 'Test SDF Network ; September 2015'

Get the Test network passphrase.

network_id()

Get the network ID of the network, which is an XDR hash of the passphrase.

Return type bytes

Returns The network ID of the network.

classmethod public_network()

Get the *Network* object representing the PUBLIC Network.

Return type *Network*

Returns PUBLIC Network

classmethod testnet_network()

Get the *Network* object representing the TESTNET Network.

Return type *Network*

Returns TESTNET Network

2.1.9 Operation

Operation

class stellar_sdk.operation.**Operation**(*source=None*)

The *Operation* object, which represents an operation on Stellar's network.

An operation is an individual command that mutates Stellar's ledger. It is typically rolled up into a transaction (a transaction is a list of operations with additional metadata).

Operations are executed on behalf of the source account specified in the transaction, unless there is an override defined for the operation.

For more on operations, see Stellar's documentation on operations as well as Stellar's List of Operations, which includes information such as the security necessary for a given operation, as well as information about when validity checks occur on the network.

The *Operation* class is typically not used, but rather one of its subclasses is typically included in transactions.

Parameters **source** (Optional[str]) – The source account for the payment. Defaults to the transaction's source account.

static from_xdr_amount(*value*)

Converts an str amount from an XDR amount object

Parameters **value** (int) – The amount to convert to a string from an XDR int64 amount.

Return type `str`

classmethod from_xdr_object(*operation_xdr_object*)

Create the appropriate `Operation` subclass from the XDR object.

Parameters **operation_xdr_object** (`Operation`) – The XDR object to create an `Operation` (or subclass) instance from.

Return type `Operation`

static get_source_from_xdr_obj(*xdr_object*)

Get the source account from account the operation xdr object.

Parameters **xdr_object** (`Operation`) – the operation xdr object.

Return type `Optional[str]`

Returns The source account from account the operation xdr object.

static to_xdr_amount(*value*)

Converts an amount to the appropriate value to send over the network as a part of an XDR object.

Each asset amount is encoded as a signed 64-bit integer in the XDR structures. An asset amount unit (that which is seen by end users) is scaled down by a factor of ten million (10,000,000) to arrive at the native 64-bit integer representation. For example, the integer amount value 25,123,456 equals 2.5123456 units of the asset. This scaling allows for seven decimal places of precision in human-friendly amount units.

This static method correctly multiplies the value by the scaling factor in order to come to the integer value used in XDR structures.

See Stellar's documentation on Asset Precision for more information.

Parameters **value** (`Union[str, Decimal]`) – The amount to convert to an integer for XDR serialization.

Return type `int`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

AccountMerge

class stellar_sdk.operation.**AccountMerge**(*destination*, *source=None*)

The `AccountMerge` object, which represents a AccountMerge operation on Stellar's network.

Transfers the native balance (the amount of XLM an account holds) to another account and removes the source account from the ledger.

Threshold: High

Parameters

- **destination** (`str`) – Destination to merge the source account into.
- **source** (`Optional[str]`) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a `AccountMerge` object from an XDR Operation object.

Return type `AccountMerge`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

AllowTrust

class `stellar_sdk.operation.`**`AllowTrust`**(*trustor*, *asset_code*, *authorize*, *source=None*)

The `AllowTrust` object, which represents a AllowTrust operation on Stellar's network.

Updates the authorized flag of an existing trustline. This can only be called by the issuer of a trustline's asset.

The issuer can only clear the authorized flag if the issuer has the AUTH_REVOCABLE_FLAG set. Otherwise, the issuer can only set the authorized flag.

Threshold: Low

Parameters

- **trustor** (`str`) – The trusting account (the one being authorized).
- **asset_code** (`str`) – The asset code being authorized.
- **authorize** (`bool`) – *True* to authorize the line, *False* to deauthorize.
- **source** (`Optional[str]`) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a `AllowTrust` object from an XDR Operation object.

Return type `AllowTrust`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

BumpSequence

class `stellar_sdk.operation.`**`BumpSequence`**(*bump_to*, *source=None*)

The `BumpSequence` object, which represents a BumpSequence operation on Stellar's network.

Bump sequence allows to bump forward the sequence number of the source account of the operation, allowing to invalidate any transactions with a smaller sequence number. If the specified bumpTo sequence number is greater than the source account's sequence number, the account's sequence number is updated with that value, otherwise it's not modified.

Threshold: Low

Parameters

- **bump_to** (`int`) – Sequence number to bump to.
- **source** (`Optional[str]`) – The optional source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a `BumpSequence` object from an XDR Operation object.

Return type `BumpSequence`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

ChangeTrust

class stellar_sdk.operation.**ChangeTrust**(*asset*, *limit=None*, *source=None*)

The *ChangeTrust* object, which represents a ChangeTrust operation on Stellar's network.

Creates, updates, or deletes a trustline. For more on trustlines, please refer to the *assets documentation <https://www.stellar.org/developers/guides/concepts/assets.html>_*.

Threshold: Medium

Parameters

- **asset** (*Asset*) – The asset for the trust line.
- **limit** (Union[str, Decimal, None]) – The limit for the asset, defaults to max int64(922337203685.4775807). If the limit is set to "0" it deletes the trustline.
- **source** (Optional[str]) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *ChangeTrust* object from an XDR Operation object.

Return type ChangeTrust

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type Operation

CreateAccount

class stellar_sdk.operation.**CreateAccount**(*destination*, *starting_balance*, *source=None*)

The *CreateAccount* object, which represents a Create Account operation on Stellar's network.

This operation creates and funds a new account with the specified starting balance.

Threshold: Medium

Parameters

- **destination** (str) – Destination account ID to create an account for.
- **starting_balance** (Union[str, Decimal]) – Amount in XLM the account should be funded for. Must be greater than the reserve balance amount.
- **source** (Optional[str]) – The source account for the payment. Defaults to the transaction's source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *CreateAccount* object from an XDR Operation object.

Return type CreateAccount

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type Operation

CreatePassiveSellOffer

class stellar_sdk.operation.**CreatePassiveSellOffer**(*selling*, *buying*, *amount*, *price*, *source=None*)

The *CreatePassiveSellOffer* object, which represents a CreatePassiveSellOffer operation on Stellar's network.

A passive sell offer is an offer that does not act on and take a reverse offer of equal price. Instead, they only take offers of lesser price. For example, if an offer exists to buy 5 BTC for 30 XLM, and you make a passive sell offer to buy 30 XLM for 5 BTC, your passive sell offer does not take the first offer.

Note that regular offers made later than your passive sell offer can act on and take your passive sell offer, even if the regular offer is of the same price as your passive sell offer.

Passive sell offers allow market makers to have zero spread. If you want to trade EUR for USD at 1:1 price and USD for EUR also at 1:1, you can create two passive sell offers so the two offers don't immediately act on each other.

Once the passive sell offer is created, you can manage it like any other offer using the manage offer operation - see `ManageOffer` for more details.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (Union[str, Decimal]) – The total amount you're selling. If 0, deletes the offer.
- **price** (Union[*Price*, str, Decimal]) – Price of 1 unit of *selling* in terms of *buying*.
- **source** (Optional[str]) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *CreatePassiveSellOffer* object from an XDR Operation object.

Return type `CreatePassiveSellOffer`

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type `Operation`

Inflation

class stellar_sdk.operation.**Inflation**(*source=None*)

The *Inflation* object, which represents a Inflation operation on Stellar's network.

This operation runs inflation.

Threshold: Low

Parameters **source** (*str*) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *Inflation* object from an XDR Operation object.

Return type `Inflation`

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type `Operation`

ManageBuyOffer

class stellar_sdk.operation.**ManageBuyOffer**(*selling*, *buying*, *amount*, *price*, *offer_id=0*, *source=None*)

The *ManageBuyOffer* object, which represents a ManageBuyOffer operation on Stellar's network.

Creates, updates, or deletes an buy offer.

If you want to create a new offer set Offer ID to 0.

If you want to update an existing offer set Offer ID to existing offer ID.

If you want to delete an existing offer set Offer ID to existing offer ID and set Amount to 0.

Threshold: Medium

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (Union[str, Decimal]) – Amount being bought. if set to 0, delete the offer.
- **price** (Union[*Price*, str, Decimal]) – Price of thing being bought in terms of what you are selling.
- **offer_id** (int) – If *0*, will create a new offer (default). Otherwise, edits an existing offer.
- **source** (Optional[str]) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *ManageBuyOffer* object from an XDR Operation object.

Return type ManageBuyOffer

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type Operation

ManageData

class stellar_sdk.operation.**ManageData**(*data_name*, *data_value*, *source=None*)

The *ManageData* object, which represents a ManageData operation on Stellar's network.

Allows you to set, modify or delete a Data Entry (name/value pair) that is attached to a particular account. An account can have an arbitrary amount of DataEntries attached to it. Each DataEntry increases the minimum balance needed to be held by the account.

DataEntries can be used for application specific things. They are not used by the core Stellar protocol.

Threshold: Medium

Parameters

- **data_name** (str) – The name of the data entry.
- **data_value** (Union[str, bytes, None]) – The value of the data entry.
- **source** – The optional source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *ManageData* object from an XDR Operation object.

Return type ManageData

to_xdr_object()
Creates an XDR Operation object that represents this *Operation*.

Return type Operation

ManageSellOffer

class stellar_sdk.operation.**ManageSellOffer**(*selling*, *buying*, *amount*, *price*, *offer_id=0*, *source=None*)

The *ManageSellOffer* object, which represents a ManageSellOffer operation on Stellar's network.

Creates, updates, or deletes an sell offer.

If you want to create a new offer set Offer ID to 0.

If you want to update an existing offer set Offer ID to existing offer ID.

If you want to delete an existing offer set Offer ID to existing offer ID and set Amount to 0.

Threshold: Medium

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (Union[str, Decimal]) – The total amount you're selling. If *0*, deletes the offer.
- **price** (Union[*Price*, str, Decimal]) – Price of 1 unit of *selling* in terms of *buying*.
- **offer_id** (int) – If *0*, will create a new offer (default). Otherwise, edits an existing offer.
- **source** (Optional[str]) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)
Creates a *ManageSellOffer* object from an XDR Operation object.

Return type ManageSellOffer

to_xdr_object()
Creates an XDR Operation object that represents this *Operation*.

Return type Operation

PathPayment

class stellar_sdk.operation.**PathPayment**(*destination*, *send_asset*, *send_max*, *dest_asset*, *dest_amount*, *path*, *source=None*)

The *PathPayment* object, which represents a PathPayment operation on Stellar's network.

Sends an amount in a specific asset to a destination account through a path of offers. This allows the asset sent (e.g. 450 XLM) to be different from the asset received (e.g. 6 BTC).

Threshold: Medium

Parameters

- **destination** (str) – The destination account to send to.
- **send_asset** (*Asset*) – The asset to pay with.
- **send_max** (Union[str, Decimal]) – The maximum amount of send_asset to send.

- **dest_asset** (*Asset*) – The asset the destination will receive.
- **dest_amount** (Union[str, Decimal]) – The amount the destination receives.
- **path** (List[*Asset*]) – A list of Asset objects to use as the path.
- **source** (Optional[str]) – The source account for the payment. Defaults to the transaction's source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *PathPaymentStrictReceive* object from an XDR Operation object.

Return type PathPaymentStrictReceive

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type Operation

PathPaymentStrictReceive

class stellar_sdk.operation.**PathPaymentStrictReceive**(*destination*, *send_asset*, *send_max*, *dest_asset*, *dest_amount*, *path*, *source=None*)

The *PathPaymentStrictReceive* object, which represents a PathPaymentStrictReceive operation on Stellar's network.

Sends an amount in a specific asset to a destination account through a path of offers. This allows the asset sent (e.g. 450 XLM) to be different from the asset received (e.g. 6 BTC).

Threshold: Medium

Parameters

- **destination** (str) – The destination account to send to.
- **send_asset** (*Asset*) – The asset to pay with.
- **send_max** (Union[str, Decimal]) – The maximum amount of send_asset to send.
- **dest_asset** (*Asset*) – The asset the destination will receive.
- **dest_amount** (Union[str, Decimal]) – The amount the destination receives.
- **path** (List[*Asset*]) – A list of Asset objects to use as the path.
- **source** (Optional[str]) – The source account for the payment. Defaults to the transaction's source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a *PathPaymentStrictReceive* object from an XDR Operation object.

Return type PathPaymentStrictReceive

to_xdr_object()

Creates an XDR Operation object that represents this *Operation*.

Return type Operation

PathPaymentStrictSend

class stellar_sdk.operation.**PathPaymentStrictSend**(*destination*, *send_asset*, *send_amount*, *dest_asset*, *dest_min*, *path*, *source=None*)

The `PathPaymentStrictSend` object, which represents a PathPaymentStrictSend operation on Stellar's network.

Sends an amount in a specific asset to a destination account through a path of offers. This allows the asset sent (e.g, 450 XLM) to be different from the asset received (e.g, 6 BTC).

Threshold: Medium

Parameters

- **destination** (`str`) – The destination account to send to.
- **send_asset** (`Asset`) – The asset to pay with.
- **send_amount** (`Union[str, Decimal]`) – Amount of send_asset to send.
- **dest_asset** (`Asset`) – The asset the destination will receive.
- **dest_min** (`Union[str, Decimal]`) – The minimum amount of dest_asset to be received.
- **path** (`List[Asset]`) – A list of Asset objects to use as the path.
- **source** (`Optional[str]`) – The source account for the payment. Defaults to the transaction's source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a `PathPaymentStrictSend` object from an XDR Operation object.

Return type `PathPaymentStrictSend`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

Payment

class stellar_sdk.operation.**Payment**(*destination*, *asset*, *amount*, *source=None*)

The `Payment` object, which represents a Payment operation on Stellar's network.

Sends an amount in a specific asset to a destination account.

Threshold: Medium

Parameters

- **destination** (`str`) – The destination account ID.
- **asset** (`Asset`) – The asset to send.
- **amount** (`str`) – The amount to send.
- **source** (`str`) – The source account for the payment. Defaults to the transaction's source account.

classmethod from_xdr_object(*operation_xdr_object*)

Creates a `Payment` object from an XDR Operation object.

Return type `Payment`

to_xdr_object()
Creates an XDR Operation object that represents this *Operation*.

Return type Operation

SetOptions

class stellar_sdk.operation.**SetOptions**(*inflation_dest=None*, *clear_flags=None*, *set_flags=None*, *master_weight=None*, *low_threshold=None*, *med_threshold=None*, *high_threshold=None*, *signer=None*, *home_domain=None*, *source=None*)

The *SetOptions* object, which represents a SetOptions operation on Stellar's network.

This operation sets the options for an account.

For more information on the signing options, please refer to the multi-sig doc.

When updating signers or other thresholds, the threshold of this operation is high.

Threshold: Medium or High

Parameters

- **inflation_dest** (Optional[str]) – Account of the inflation destination.
- **clear_flags** (Union[int, *Flag*, None]) – Indicates which flags to clear. For details about the flags, please refer to the accounts doc. The bit mask integer subtracts from the existing flags of the account. This allows for setting specific bits without knowledge of existing flags, you can also use *stellar_sdk.operation.set_options.Flag* - AUTHORIZATION_REQUIRED = 1 - AUTHORIZATION_REVOCABLE = 2 - AUTHORIZATION_IMMUTABLE = 4
- **set_flags** (Union[int, *Flag*, None]) – Indicates which flags to set. For details about the flags, please refer to the accounts doc. The bit mask integer adds onto the existing flags of the account. This allows for setting specific bits without knowledge of existing flags, you can also use *stellar_sdk.operation.set_options.Flag* - AUTHORIZATION_REQUIRED = 1 - AUTHORIZATION_REVOCABLE = 2 - AUTHORIZATION_IMMUTABLE = 4
- **master_weight** (Optional[int]) – A number from 0-255 (inclusive) representing the weight of the master key. If the weight of the master key is updated to 0, it is effectively disabled.
- **low_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a low threshold.
- **med_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a medium threshold.
- **high_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a high threshold.
- **home_domain** (Optional[str]) – sets the home domain used for reverse federation lookup.
- **signer** (Optional[*Signer*]) – Add, update, or remove a signer from the account.
- **source** (Optional[str]) – The source account (defaults to transaction source).

classmethod from_xdr_object(*operation_xdr_object*)
Creates a *SetOptions* object from an XDR Operation object.

Return type `SetOptions`

to_xdr_object()

Creates an XDR Operation object that represents this `Operation`.

Return type `Operation`

class stellar_sdk.operation.set_options.**Flag**

Indicates which flags to set. For details about the flags, please refer to the accounts doc. The bit mask integer adds onto the existing flags of the account.

2.1.10 Price

class stellar_sdk.price.**Price**(*n*, *d*)

Create a new price. Price in Stellar is represented as a fraction.

Parameters

- **n** (`int`) – numerator
- **d** (`int`) – denominator

classmethod from_raw_price(*price*)

Create a `Price` from the given str price.

Parameters **price** (`str`) – the str price. (ex. *'0.125'*)

Return type `Price`

Returns A new `Price` object from the given str price.

Raises `NoApproximationError`: if the approximation could not not be found.

classmethod from_xdr_object(*price_xdr_object*)

Create a `Price` from an XDR Asset object.

Parameters **price_xdr_object** (`Price`) – The XDR Price object.

Return type `Price`

Returns A new `Price` object from the given XDR Price object.

to_xdr_object()

Returns the xdr object for this price object.

Return type `Price`

Returns XDR Price object

2.1.11 Server

class stellar_sdk.server.**Server**(*horizon_url='https://horizon-testnet.stellar.org/'*, *client=None*)

Server handles the network connection to a Horizon instance and exposes an interface for requests to that instance.

Here we need to talk about the **client** parameter, if you do not specify the client, we will use the `stellar_sdk.client.requests_client.RequestsClient` instance by default, it is a synchronous HTTPClient, you can also specify an asynchronous HTTP Client, for example: `stellar_sdk.client.aiohttp_client.AiohttpClient`. If you use a synchronous client, then all requests are synchronous. If you use an asynchronous client, then all requests are asynchronous. The choice is in your hands.

Parameters

- **horizon_url** (str) – Horizon Server URL (ex. *https://horizon-testnet.stellar.org*)
- **client** (Union[BaseAsyncClient, BaseSyncClient, None]) – Http Client used to send the request

Raises TypeError: if the client does not meet the standard.

accounts()

Return type AccountsCallBuilder

Returns New stellar_sdk.call_builder.AccountsCallBuilder object configured by a current Horizon server configuration.

assets()

Return type AssetsCallBuilder

Returns New stellar_sdk.call_builder.AssetsCallBuilder object configured by a current Horizon server configuration.

close()

Close underlying connector.

Release all acquired resources.

Return type Optional[Coroutine[Any, Any, None]]

data(*account_id*, *data_name*)

Returns New stellar_sdk.call_builder.DataCallBuilder object configured by a current Horizon server configuration.

effects()

Return type EffectsCallBuilder

Returns New stellar_sdk.call_builder.EffectsCallBuilder object configured by a current Horizon server configuration.

fee_stats()

Return type FeeStatsCallBuilder

Returns New stellar_sdk.call_builder.FeeStatsCallBuilder object configured by a current Horizon server configuration.

fetch_base_fee()

Fetch the base fee. Since this hits the server, if the server call fails, you might get an error. You should be prepared to use a default value if that happens.

Return type Union[int, Coroutine[Any, Any, int]]

Returns the base fee

Raises ConnectionError NotFoundError BadRequestError BadResponseError UnknownRequestError

ledgers()

Return type LedgersCallBuilder

Returns New stellar_sdk.call_builder.LedgersCallBuilder object configured by a current Horizon server configuration.

load_account(*account_id*)

Fetches an account's most current state in the ledger and then creates and returns an `stellar_sdk.account.Account` object.

Parameters **account_id** (`str`) – The account to load.

Return type `Union[Account, Coroutine[Any, Any, Account]]`

Returns an `stellar_sdk.account.Account` object.

Raises `ConnectionError` `NotFoundError` `BadRequestError` `BadResponseError` `UnknownRequestError`

offers()

Return type `OffersCallBuilder`

Returns New `stellar_sdk.call_builder.OffersCallBuilder` object configured by a current Horizon server configuration.

operations()

Return type `OperationsCallBuilder`

Returns New `stellar_sdk.call_builder.OperationsCallBuilder` object configured by a current Horizon server configuration.

orderbook(*selling*, *buying*)

Parameters

- **selling** (`Asset`) – Asset being sold
- **buying** (`Asset`) – Asset being bought

Return type `OrderbookCallBuilder`

Returns New `stellar_sdk.call_builder.OrderbookCallBuilder` object configured by a current Horizon server configuration.

paths(*source_account*, *destination_account*, *destination_asset*, *destination_amount*)

Parameters

- **source_account** (`str`) – The sender's account ID. Any returned path must use a source that the sender can hold.
- **destination_account** (`str`) – The destination account ID that any returned path should use.
- **destination_asset** (`Asset`) – The destination asset.
- **destination_amount** (`str`) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

Return type `PathsCallBuilder`

Returns New `stellar_sdk.call_builder.PathsCallBuilder` object configured by a current Horizon server configuration.

payments()

Return type `PaymentsCallBuilder`

Returns New `stellar_sdk.call_builder.PaymentsCallBuilder` object configured by a current Horizon server configuration.

root()

Return type `RootCallBuilder`

Returns New `stellar_sdk.call_builder.RootCallBuilder` object configured by a current Horizon server configuration.

strict_receive_paths(*source*, *destination_asset*, *destination_amount*)

Parameters

- **source** (`Union[str, List[Asset]]`) – The sender's account ID or a list of Assets. Any returned path must use a source that the sender can hold.
- **destination_asset** (`Asset`) – The destination asset.
- **destination_amount** (`str`) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

Returns New `stellar_sdk.call_builder.StrictReceivePathsCallBuilder` object configured by a current Horizon server configuration.

strict_send_paths(*source_asset*, *source_amount*, *destination*)

Parameters

- **source_asset** (`Asset`) – The asset to be sent.
- **source_amount** (`str`) – The amount, denominated in the source asset, that any returned path should be able to satisfy.
- **destination** (`Union[str, List[Asset]]`) – The destination account or the destination assets.

Returns New `stellar_sdk.call_builder.StrictReceivePathsCallBuilder` object configured by a current Horizon server configuration.

submit_transaction(*transaction_envelope*)

Submits a transaction to the network.

Parameters **transaction_envelope** (`Union[TransactionEnvelope, str]`) – `stellar_sdk.transaction_envelope.TransactionEnvelope` object or base64 encoded xdr

Return type `Union[Dict[str, Any], Coroutine[Any, Any, Dict[str, Any]]]`

Returns the response from horizon

trade_aggregations(*base*, *counter*, *resolution*, *start_time=None*, *end_time=None*, *offset=None*)

Parameters

- **base** (`Asset`) – base asset
- **counter** (`Asset`) – counter asset
- **resolution** (`int`) – segment duration as millis since epoch. *Supported values are 1 minute (60000), 5 minutes (300000), 15 minutes (900000), 1 hour (3600000), 1 day (86400000) and 1 week (604800000).*
- **start_time** (`Optional[int]`) – lower time boundary represented as millis since epoch
- **end_time** (`Optional[int]`) – upper time boundary represented as millis since epoch
- **offset** (`Optional[int]`) – segments can be offset using this parameter. Expressed in milliseconds. *Can only be used if the resolution is greater than 1 hour. Value must be in whole hours, less than the provided resolution, and less than 24 hours.*

Return type `TradeAggregationsCallBuilder`

Returns New `stellar_sdk.call_builder.TradeAggregationsCallBuilder` object configured by a current Horizon server configuration.

trades()

Return type `TradesCallBuilder`

Returns New `stellar_sdk.call_builder.TradesCallBuilder` object configured by a current Horizon server configuration.

transactions()

Return type `TransactionsCallBuilder`

Returns New `stellar_sdk.call_builder.TransactionsCallBuilder` object configured by a current Horizon server configuration.

2.1.12 Signer

class stellar_sdk.signer.**Signer**(*signer_key*, *weight*)

The `Signer` object, which represents an account signer on Stellar's network.

Parameters

- **signer_key** (`SignerKey`) – The XDR signer object
- **weight** –

classmethod ed25519_public_key(*account_id*, *weight*)

Create ED25519 PUBLIC KEY Signer from account id.

Parameters

- **account_id** (`str`) – account id
- **weight** (`int`) – The weight of the signer (0 to delete or 1-255)

Return type `Signer`

Returns ED25519 PUBLIC KEY Signer

Raises `Ed25519PublicKeyInvalidError`: if `account_id` is not a valid ed25519 public key.

classmethod from_xdr_object(*signer_xdr_object*)

Create a `Signer` from an XDR TimeBounds object.

Parameters **signer_xdr_object** (`Signer`) – The XDR Signer object.

Return type `Signer`

Returns A new `Signer` object from the given XDR Signer object.

classmethod pre_auth_tx(*pre_auth_tx_hash*, *weight*)

Create Pre AUTH TX Signer from the sha256 hash of a transaction, click here for more information.

Parameters

- **pre_auth_tx_hash** (`bytes`) – The sha256 hash of a transaction.
- **weight** (`int`) – The weight of the signer (0 to delete or 1-255)

Return type `Signer`

Returns Pre AUTH TX Signer

classmethod sha256_hash(*sha256_hash*, *weight*)
Create SHA256 HASH Signer from a sha256 hash of a preimage, click here for more information.

Parameters

- **sha256_hash** (bytes) – a sha256 hash of a preimage
- **weight** (int) – The weight of the signer (0 to delete or 1-255)

Return type *Signer*

Returns SHA256 HASH Signer

to_xdr_object()
Returns the xdr object for this Signer object.

Return type Signer

Returns XDR Signer object

2.1.13 TimeBounds

class stellar_sdk.time_bounds.**TimeBounds**(*min_time*, *max_time*)
TimeBounds represents the time interval that a transaction is valid.

The UNIX timestamp (in seconds), determined by ledger time, of a lower and upper bound of when this transaction will be valid. If a transaction is submitted too early or too late, it will fail to make it into the transaction set. **max_time** equal 0 means that it's not set.

See Stellar's documentation on Transactions for more information on how TimeBounds are used within transactions.

Parameters

- **min_time** (int) – the UNIX timestamp (in seconds)
- **max_time** (int) – the UNIX timestamp (in seconds)

Raises *ValueError*: if max_time less than min_time.

classmethod from_xdr_object(*time_bounds_xdr_object*)
Create a *TimeBounds* from an XDR TimeBounds object.

Parameters **time_bounds_xdr_object** (TimeBounds) – The XDR TimeBounds object.

Return type *TimeBounds*

Returns A new *TimeBounds* object from the given XDR TimeBounds object.

to_xdr_object()
Returns the xdr object for this TimeBounds object.

Return type TimeBounds

Returns XDR TimeBounds object

2.1.14 Transaction

class stellar_sdk.transaction.**Transaction**(*source*, *sequence*, *fee*, *operations*, *memo=None*, *time_bounds=None*)
The *Transaction* object, which represents a transaction on Stellar's network.

A transaction contains a list of operations, which are all executed in order as one ACID transaction, along with an associated source account, fee, account sequence number, list of signatures, both an optional memo and an optional TimeBounds. Typically a `Transaction` is placed in a `TransactionEnvelope` which is then signed before being sent over the network.

For more information on Transactions in Stellar, see Stellar's guide on transactions.

Parameters

- **source** (`Union[Keypair, str]`) – the source account for the transaction.
- **sequence** (`int`) – The sequence number for the transaction.
- **fee** (`int`) – The fee amount for the transaction, which should equal FEE (currently 100 stroops) multiplied by the number of operations in the transaction. See Stellar's latest documentation on fees for more information.
- **operations** (`List[Operation]`) – A list of operations objects (typically its subclasses as defined in `stellar_sdk.operation.Operation`.
- **time_bounds** (`Optional[TimeBounds]`) – The timebounds for the validity of this transaction.
- **memo** (`Optional[Memo]`) – The memo being sent with the transaction, being represented as one of the subclasses of the `Memo` object.

classmethod from_xdr_object(*tx_xdr_object*)

Create a new `Transaction` from an XDR object.

Parameters **tx_xdr_object** – The XDR object that represents a transaction.

Return type `Transaction`

Returns A new `Transaction` object from the given XDR Transaction object.

to_xdr_object()

Get an XDR object representation of this `Transaction`.

Return type `Transaction`

Returns XDR Transaction object

2.1.15 TransactionEnvelope

class stellar_sdk.transaction_envelope.**TransactionEnvelope**(*transaction*, *network_passphrase*, *signatures=None*)

The `TransactionEnvelope` object, which represents a transaction envelope ready to sign and submit to send over the network.

When a transaction is ready to be prepared for sending over the network, it must be put into a `TransactionEnvelope`, which includes additional metadata such as the signers for a given transaction. Ultimately, this class handles signing and conversion to and from XDR for usage on Stellar's network.

Parameters

- **transaction** (`Transaction`) – The transaction that is encapsulated in this envelope.
- **signatures** (`list`) – which contains a list of signatures that have already been created.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from.

classmethod from_xdr(*xdr*, *network_passphrase*)

Create a new `TransactionEnvelope` from an XDR string.

Parameters

- **xdr** (`str`) – The XDR string that represents a transaction envelope.
- **network** – which network this transaction envelope is associated with.

Return type `TransactionEnvelope`

Returns A new `TransactionEnvelope` object from the given XDR TransactionEnvelope base64 string object.

classmethod from_xdr_object(*te_xdr_object*, *network_passphrase*)

Create a new `TransactionEnvelope` from an XDR object.

Parameters

- **te_xdr_object** (`TransactionEnvelope`) – The XDR object that represents a transaction envelope.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from.

Return type `TransactionEnvelope`

Returns A new `TransactionEnvelope` object from the given XDR TransactionEnvelope object.

hash()

Get the XDR Hash of the signature base.

This hash is ultimately what is signed before transactions are sent over the network. See `signature_base()` for more details about this process.

Return type `bytes`

Returns The XDR Hash of this transaction envelope's signature base.

hash_hex()

Return a hex encoded hash for this transaction envelope.

Return type `str`

Returns A hex encoded hash for this transaction envelope.

sign(*signer*)

Sign this transaction envelope with a given keypair.

Note that the signature must not already be in this instance's list of signatures.

Parameters **signer** (`Union[Keypair, str]`) – The keypair or secret to use for signing this transaction envelope.

Raise `SignatureExistError`: if this signature already exists.

Return type `None`

sign_hashx(*preimage*)

Sign this transaction envelope with a Hash(x) signature.

See Stellar's documentation on Multi-Sig for more details on Hash(x) signatures.

Parameters **preimage** (`bytes`) – 32 byte hash or hex encoded string , the "x" value to be hashed and used as a signature.

Return type `None`

signature_base()

Get the signature base of this transaction envelope.

Return the "signature base" of this transaction, which is the value that, when hashed, should be signed to create a signature that validators on the Stellar Network will accept.

It is composed of a 4 prefix bytes followed by the xdr-encoded form of this transaction.

Return type `bytes`

Returns The signature base of this transaction envelope.

to_xdr()

Get the base64 encoded XDR string representing this `TransactionEnvelope`.

Return type `str`

Returns XDR TransactionEnvelope base64 string object

to_xdr_object()

Get an XDR object representation of this `TransactionEnvelope`.

Return type `TransactionEnvelope`

Returns XDR TransactionEnvelope object

2.1.16 TransactionBuilder

class stellar_sdk.transaction_builder.**TransactionBuilder**(*source_account, network_passphrase='Test SDF Network ; September 2015', base_fee=100*)

Transaction builder helps constructs a new `TransactionEnvelope` using the given `Account` as the transaction's "source account". The transaction will use the current sequence number of the given account as its sequence number and increment the given account's sequence number by one. The given source account must include a private key for signing the transaction or an error will be thrown.

Be careful about **unsubmitted transactions**! When you build a transaction, stellar-sdk automatically increments the source account's sequence number. If you end up not submitting this transaction and submitting another one instead, it'll fail due to the sequence number being wrong. So if you decide not to use a built transaction, make sure to update the source account's sequence number with `stellar_sdk.Server.load_account()` before creating another transaction.

Parameters

- **source_account** (`Account`) – The source account for this transaction.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from. Defaults to **Test SDF Network ; September 2015**.
- **base_fee** (`int`) – Base fee in stroops. The network base fee is obtained by default from the latest ledger. Transaction fee is equal to base fee times number of operations in this transaction.

add_hash_memo(*memo_hash*)

Set the memo for the transaction to a new `HashMemo`.

Parameters **memo_hash** (`Union[bytes, str]`) – A 32 byte hash or hex encoded string to use as the memo.

Return type *TransactionBuilder*

Returns This builder instance.

Raises *MemoInvalidException*: if `memo_hash` is not a valid hash memo.

add_id_memo(*memo_id*)

Set the memo for the transaction to a new *IdMemo*.

Parameters **memo_id** (`int`) – A 64 bit unsigned integer to set as the memo.

Return type *TransactionBuilder*

Returns This builder instance.

Raises *MemoInvalidException*: if `memo_id` is not a valid id memo.

add_memo(*memo*)

Set the memo for the transaction build by this `Builder`.

Parameters **memo** (*Memo*) – A memo to add to this transaction.

Return type *TransactionBuilder*

Returns This builder instance.

add_return_hash_memo(*memo_return*)

Set the memo for the transaction to a new *RetHashMemo*.

Parameters **memo_return** (`Union`[`bytes`, `str`]) – A 32 byte hash or hex encoded string intended to be interpreted as the hash of the transaction the sender is refunding.

Return type *TransactionBuilder*

Returns This builder instance.

Raises *MemoInvalidException*: if `memo_return` is not a valid return hash memo.

add_text_memo(*memo_text*)

Set the memo for the transaction to a new *TextMemo*.

Parameters **memo_text** (`Union`[`str`, `bytes`]) – The text for the memo to add.

Return type *TransactionBuilder*

Returns This builder instance.

Raises *MemoInvalidException*: if `memo_text` is not a valid text memo.

add_time_bounds(*min_time*, *max_time*)

Add a time bound to this transaction.

Add a UNIX timestamp, determined by ledger time, of a lower and upper bound of when this transaction will be valid. If a transaction is submitted too early or too late, it will fail to make it into the transaction set. maxTime equal 0 means that it's not set.

Parameters

- **min_time** (`int`) – the UNIX timestamp (in seconds)
- **max_time** (`int`) – the UNIX timestamp (in seconds)

Return type *TransactionBuilder*

Returns This builder instance.

append_account_merge_op(*destination*, *source=None*)

Append a *AccountMerge* operation to the list of operations.

Parameters

- **destination** (str) – The ID of the offer. 0 for new offer. Set to existing offer ID to update or delete.
- **source** (Optional[str]) – The source address that is being merged into the destination account.

Return type *TransactionBuilder*

Returns This builder instance.

append_allow_trust_op (*trustor*, *asset_code*, *authorize*, *source=None*)

Append an *AllowTrust* operation to the list of operations.

Parameters

- **trustor** (str) – The account of the recipient of the trustline.
- **asset_code** (str) – The asset of the trustline the source account is authorizing. For example, if an anchor wants to allow another account to hold its USD credit, the type is USD:anchor.
- **authorize** (bool) – Flag indicating whether the trustline is authorized.
- **source** (Optional[str]) – The source address that is establishing the trust in the allow trust operation.

Return type *TransactionBuilder*

Returns This builder instance.

append_bump_sequence_op (*bump_to*, *source=None*)

Append a *BumpSequence* operation to the list of operations.

Parameters

- **bump_to** (int) – Sequence number to bump to.
- **source** (Optional[str]) – The source address that is running the inflation operation.

Return type *TransactionBuilder*

Returns This builder instance.

append_change_trust_op (*asset_code*, *asset_issuer*, *limit=None*, *source=None*)

Append a *ChangeTrust* operation to the list of operations.

Parameters

- **asset_issuer** (str) – The issuer address for the asset.
- **asset_code** (str) – The asset code for the asset.
- **limit** (Union[str, Decimal, None]) – The limit of the new trustline.
- **source** (Optional[str]) – The source address to add the trustline to.

Return type *TransactionBuilder*

Returns This builder instance.

append_create_account_op (*destination*, *starting_balance*, *source=None*)

Append a *CreateAccount* operation to the list of operations.

Parameters

- **destination** (str) – Account address that is created and funded.

- **starting_balance** (Union[str, Decimal]) – Amount of XLM to send to the newly created account. This XLM comes from the source account.
- **source** (Optional[str]) – The source address to deduct funds from to fund the new account.

Return type *TransactionBuilder*

Returns This builder instance.

append_create_passive_sell_offer_op(*selling_code*, *selling_issuer*, *buying_code*, *buying_issuer*, *amount*, *price*, *source=None*)

Append a *CreatePassiveSellOffer* operation to the list of operations.

Parameters

- **selling_code** (str) – The asset code for the asset the offer creator is selling.
- **selling_issuer** (Optional[str]) – The issuing address for the asset the offer creator is selling.
- **buying_code** (str) – The asset code for the asset the offer creator is buying.
- **buying_issuer** (Optional[str]) – The issuing address for the asset the offer creator is buying.
- **amount** (Union[str, Decimal]) – Amount of the asset being sold. Set to 0 if you want to delete an existing offer.
- **price** (Union[str, *Price*, Decimal]) – Price of 1 unit of selling in terms of buying.
- **source** (Optional[str]) – The source address that is creating a passive offer on Stellar's distributed exchange.

Return type *TransactionBuilder*

Returns This builder instance.

append_ed25519_public_key_signer(*account_id*, *weight*, *source=None*)

Add a ed25519 public key signer to an account.

Add a ed25519 public key signer to an account via a SetOptions <stellar_sdk.operation.SetOptions operation. This is a helper function for *append_set_options_op()*.

Parameters

- **account_id** (str) – The account id of the new ed25519_public_key signer.
- **weight** (int) – The weight of the new signer.
- **source** – The source account that is adding a signer to its list of signers.

Return type *TransactionBuilder*

Returns This builder instance.

append_hashx_signer(*sha256_hash*, *weight*, *source=None*)

Add a sha256 hash(HashX) signer to an account.

Add a HashX signer to an account via a SetOptions <stellar_sdk.operation.SetOptions operation. This is a helper function for *append_set_options_op()*.

Parameters

- **sha256_hash** (*[<class 'bytes'>, <class 'str'>]*) – The address of the new sha256 hash(hashX) signer, a 32 byte hash or hex encoded string.
- **weight** (int) – The weight of the new signer.

- **source** – The source account that is adding a signer to its list of signers.

Return type `TransactionBuilder`

Returns This builder instance.

append_inflation_op(*source=None*)

Append a `Inflation` operation to the list of operations.

Parameters **source** (`Optional[str]`) – The source address that is running the inflation operation.

Return type `TransactionBuilder`

Returns This builder instance.

append_manage_buy_offer_op(*selling_code*, *selling_issuer*, *buying_code*, *buying_issuer*, *amount*, *price*, *offer_id=0*, *source=None*)

Append a `ManageBuyOffer` operation to the list of operations.

Parameters

- **selling_code** (`str`) – The asset code for the asset the offer creator is selling.
- **selling_issuer** (`Optional[str]`) – The issuing address for the asset the offer creator is selling.
- **buying_code** (`str`) – The asset code for the asset the offer creator is buying.
- **buying_issuer** (`Optional[str]`) – The issuing address for the asset the offer creator is buying.
- **amount** (`Union[str, Decimal]`) – Amount being bought. if set to. Set to 0 if you want to delete an existing offer.
- **price** (`Union[str, Decimal, Price]`) – Price of thing being bought in terms of what you are selling.
- **offer_id** (`int`) – The ID of the offer. 0 for new offer. Set to existing offer ID to update or delete.
- **source** (`Optional[str]`) – The source address that is managing a buying offer on Stellar's distributed exchange.

Return type `TransactionBuilder`

Returns This builder instance.

append_manage_data_op(*data_name*, *data_value*, *source=None*)

Append a `ManageData` operation to the list of operations.

Parameters

- **data_name** (`str`) – String up to 64 bytes long. If this is a new Name it will add the given name/value pair to the account. If this Name is already present then the associated value will be modified.
- **data_value** (`Union[str, bytes, None]`) – If not present then the existing Name will be deleted. If present then this value will be set in the DataEntry. Up to 64 bytes long.
- **source** (`Optional[str]`) – The source account on which data is being managed. operation.

Return type `TransactionBuilder`

Returns This builder instance.

append_manage_sell_offer_op(*selling_code*, *selling_issuer*, *buying_code*, *buying_issuer*, *amount*, *price*, *offer_id=0*, *source=None*)

Append a `ManageSellOffer` operation to the list of operations.

Parameters

- **selling_code** (`str`) – The asset code for the asset the offer creator is selling.
- **selling_issuer** (`Optional[str]`) – The issuing address for the asset the offer creator is selling.
- **buying_code** (`str`) – The asset code for the asset the offer creator is buying.
- **buying_issuer** (`Optional[str]`) – The issuing address for the asset the offer creator is buying.
- **amount** (`Union[str, Decimal]`) – Amount of the asset being sold. Set to 0 if you want to delete an existing offer.
- **price** (`Union[str, Price, Decimal]`) – Price of 1 unit of selling in terms of buying.
- **offer_id** (`int`) – The ID of the offer. 0 for new offer. Set to existing offer ID to update or delete.
- **source** (`Optional[str]`) – The source address that is managing an offer on Stellar's distributed exchange.

Return type `TransactionBuilder`

Returns This builder instance.

append_operation(*operation*)

Add an operation to the builder instance

Parameters **operation** (`Operation`) – an operation

Return type `TransactionBuilder`

Returns This builder instance.

append_path_payment_op(*destination*, *send_code*, *send_issuer*, *send_max*, *dest_code*, *dest_issuer*, *dest_amount*, *path*, *source=None*)

Append a `PathPayment` operation to the list of operations.

Parameters

- **destination** (`str`) – The destination address (Account ID) for the payment.
- **send_code** (`str`) – The asset code for the source asset deducted from the source account.
- **send_issuer** (`Optional[str]`) – The address of the issuer of the source asset.
- **send_max** (`Union[str, Decimal]`) – The maximum amount of send asset to deduct (excluding fees).
- **dest_code** (`str`) – The asset code for the final destination asset sent to the recipient.
- **dest_issuer** (`Optional[str]`) – Account address that receives the payment.
- **dest_amount** (`Union[str, Decimal]`) – The amount of destination asset the destination account receives.
- **path** (`List[Asset]`) – A list of Asset objects to use as the path.
- **source** – The source address of the path payment.

Return type `TransactionBuilder`

Returns This builder instance.

append_path_payment_strict_receive_op(*destination*, *send_code*, *send_issuer*, *send_max*, *dest_code*, *dest_issuer*, *dest_amount*, *path*, *source=None*)

Append a `PathPaymentStrictReceive` operation to the list of operations.

Parameters

- **destination** (`str`) – The destination address (Account ID) for the payment.
- **send_code** (`str`) – The asset code for the source asset deducted from the source account.
- **send_issuer** (`Optional[str]`) – The address of the issuer of the source asset.
- **send_max** (`Union[str, Decimal]`) – The maximum amount of send asset to deduct (excluding fees).
- **dest_code** (`str`) – The asset code for the final destination asset sent to the recipient.
- **dest_issuer** (`Optional[str]`) – Account address that receives the payment.
- **dest_amount** (`Union[str, Decimal]`) – The amount of destination asset the destination account receives.
- **path** (`List[Asset]`) – A list of Asset objects to use as the path.
- **source** – The source address of the path payment.

Return type `TransactionBuilder`

Returns This builder instance.

append_path_payment_strict_send_op(*destination*, *send_code*, *send_issuer*, *send_amount*, *dest_code*, *dest_issuer*, *dest_min*, *path*, *source=None*)

Append a `PathPaymentStrictSend` operation to the list of operations.

Parameters

- **destination** (`str`) – The destination address (Account ID) for the payment.
- **send_code** (`str`) – The asset code for the source asset deducted from the source account.
- **send_issuer** (`Optional[str]`) – The address of the issuer of the source asset.
- **send_amount** (`Union[str, Decimal]`) – Amount of send_asset to send.
- **dest_code** (`str`) – The asset code for the final destination asset sent to the recipient.
- **dest_issuer** (`Optional[str]`) – Account address that receives the payment.
- **dest_min** (`Union[str, Decimal]`) – The minimum amount of dest_asset to be received.
- **path** (`List[Asset]`) – A list of Asset objects to use as the path.
- **source** – The source address of the path payment.

Return type `TransactionBuilder`

Returns This builder instance.

append_payment_op(*destination*, *amount*, *asset_code='XLM'*, *asset_issuer=None*, *source=None*)

Append a `Payment` operation to the list of operations.

Parameters

- **destination** (str) – Account address that receives the payment.
- **amount** (Union[str, Decimal]) – The amount of the currency to send in the payment.
- **asset_code** (str) – The asset code for the asset to send.
- **asset_issuer** (Optional[str]) – The address of the issuer of the asset.
- **source** (Optional[str]) – The source address of the payment.

Return type *TransactionBuilder*

Returns This builder instance.

append_pre_auth_tx_signer(*pre_auth_tx_hash*, *weight*, *source=None*)

Add a PreAuthTx signer to an account.

Add a PreAuthTx signer to an account via a SetOptions <stellar_sdk.operation.SetOptions operation. This is a helper function for *append_set_options_op()*.

Parameters

- **pre_auth_tx_hash** (bytes) – The address of the new preAuthTx signer - obtained by calling hash on the *TransactionEnvelope*, a 32 byte hash or hex encoded string.
- **weight** (int) – The weight of the new signer.
- **source** – The source account that is adding a signer to its list of signers.

Return type *TransactionBuilder*

Returns This builder instance.

append_set_options_op(*inflation_dest=None*, *clear_flags=None*, *set_flags=None*, *master_weight=None*, *low_threshold=None*, *med_threshold=None*, *high_threshold=None*, *home_domain=None*, *signer=None*, *source=None*)

Append a *SetOptions* operation to the list of operations.

Parameters

- **inflation_dest** (Optional[str]) – Account of the inflation destination.
- **clear_flags** (Union[int, *Flag*, None]) – Indicates which flags to clear. For details about the flags, please refer to the accounts doc. The bit mask integer subtracts from the existing flags of the account. This allows for setting specific bits without knowledge of existing flags, you can also use *stellar_sdk.operation.set_options.Flag* - AUTHORIZATION_REQUIRED = 1 - AUTHORIZATION_REVOCABLE = 2 - AUTHORIZATION_IMMUTABLE = 4
- **set_flags** (Union[int, *Flag*, None]) – Indicates which flags to set. For details about the flags, please refer to the accounts doc. The bit mask integer adds onto the existing flags of the account. This allows for setting specific bits without knowledge of existing flags, you can also use *stellar_sdk.operation.set_options.Flag* - AUTHORIZATION_REQUIRED = 1 - AUTHORIZATION_REVOCABLE = 2 - AUTHORIZATION_IMMUTABLE = 4
- **master_weight** (Optional[int]) – A number from 0-255 (inclusive) representing the weight of the master key. If the weight of the master key is updated to 0, it is effectively disabled.
- **low_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a low threshold.

- **med_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a medium threshold.
- **high_threshold** (Optional[int]) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a high threshold.
- **home_domain** (Optional[str]) – sets the home domain used for reverse federation lookup.
- **signer** (Optional[*Signer*]) – Add, update, or remove a signer from the account.
- **source** (Optional[str]) – The source account (defaults to transaction source).

Return type *TransactionBuilder*

Returns This builder instance.

build()

This will build the transaction envelope. It will also increment the source account's sequence number by 1.

Return type *TransactionEnvelope*

Returns The transaction envelope.

static from_xdr(*xdr*, *network_passphrase*)

Create a *TransactionBuilder* via an XDR object.

In addition, sets the fields of this builder (the transaction envelope, transaction, operations, source, etc.) to all of the fields in the provided XDR transaction envelope, but the signature will not be added to it.

Parameters

- **xdr** (str) – The XDR object representing the transaction envelope to which this builder is setting its state to.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from.

Return type *TransactionBuilder*

set_timeout(*timeout*)

Set timeout for the transaction, actually set a TimeBounds.

Parameters **timeout** (int) – timeout in second.

Return type *TransactionBuilder*

Returns This builder instance.

Raises *ValueError*: if time_bound is already set.

2.1.17 Stellar Ecosystem Proposals

SEP 0001: stellar.toml

stellar_sdk.sep.stellar_toml.**fetch_stellar_toml**(*domain*, *client=None*, *use_http=False*)

Retrieve the stellar.toml file from a given domain.

Retrieve the stellar.toml file for information about interacting with Stellar's federation protocol for a given Stellar Anchor (specified by a domain).

Parameters

- **domain** (str) – The domain the .toml file is hosted at.
- **use_http** (bool) – Specifies whether the request should go over plain HTTP vs HTTPS. Note it is recommend that you *always* use HTTPS.
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*, None]) – Http Client used to send the request.

Return type Union[Coroutine[Any, Any, Dict[str, Any]], Dict[str, Any]]

Returns The stellar.toml file as a an object via `toml.loads()`.

Raises *StellarTomlNotFoundError*: if the Stellar toml file could not not be found.

SEP 0002: Federation protocol

`stellar_sdk.sep.federation.`**`resolve_stellar_address`**(*stellar_address*, *client=None*, *federation_url=None*, *use_http=False*)

Get the federation record if the user was found for a given Stellar address.

Parameters

- **stellar_address** (str) – address Stellar address (ex. bob*stellar.org).
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*, None]) – Http Client used to send the request.
- **federation_url** (Optional[str]) – The federation server URL (ex. *https://stellar.org/federation*), if you don't set this value, we will try to get it from `stellar_address`.
- **use_http** (bool) – Specifies whether the request should go over plain HTTP vs HTTPS. Note it is recommend that you *always* use HTTPS.

Return type Union[Coroutine[Any, Any, *FederationRecord*], *FederationRecord*]

Returns Federation record.

`stellar_sdk.sep.federation.`**`resolve_account_id`**(*account_id*, *domain=None*, *federation_url=None*, *client=None*, *use_http=False*)

Given an account ID, get their federation record if the user was found

Parameters

- **account_id** (str) – Account ID (ex. GBYNR2QJXLBCBTRN44MRORCMI4YO7FZPFBCNOKTOBCAAFC
- **domain** (Optional[str]) – Get `federation_url` from the domain, you don't need to set this value if `federation_url` is set.
- **federation_url** (Optional[str]) – The federation server URL (ex. https://stellar.org/federation).
- **client** (Union[*BaseAsyncClient*, *BaseSyncClient*, None]) – Http Client used to send the request.
- **use_http** (bool) – Specifies whether the request should go over plain HTTP vs HTTPS. Note it is recommend that you *always* use HTTPS.

Return type Union[Coroutine[Any, Any, *FederationRecord*], *FederationRecord*]

Returns Federation record.

class stellar_sdk.sep.federation.**FederationRecord**(*account_id*, *stellar_address*, *memo_type*, *memo*)

SEP 0005: Key Derivation Methods for Stellar Accounts

class stellar_sdk.sep.mnemonic.**StellarMnemonic**(*language=<Language.ENGLISH: 'english'>*)

Please use `Keypair.generate_mnemonic_phrase()` and `Keypair.from_mnemonic_phrase()`

class stellar_sdk.sep.mnemonic.**Language**

The type of language supported by the mnemonic.

CHINESE_SIMPLIFIED = 'chinese_simplified'

CHINESE_TRADITIONAL = 'chinese_traditional'

ENGLISH = 'english'

FRENCH = 'french'

ITALIAN = 'italian'

JAPANESE = 'japanese'

KOREAN = 'korean'

SPANISH = 'spanish'

SEP 0010: Stellar Web Authentication

stellar_sdk.sep.stellar_web_authentication.**build_challenge_transaction**(*server_secret*, *client_account_id*, *anchor_name*, *network_passphrase*, *timeout=900*)

Returns a valid SEP0010 challenge transaction which you can use for Stellar Web Authentication.

Parameters

- **server_secret** (`str`) – secret key for server's signing account.
- **client_account_id** (`str`) – The stellar account that the wallet wishes to authenticate with the server.
- **anchor_name** (`str`) – Anchor's name to be used in the manage_data key.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')
- **timeout** (`int`) – Challenge duration in seconds (default to 15 minutes).

Return type `str`

Returns A base64 encoded string of the raw TransactionEnvelope xdr struct for the transaction.

`stellar_sdk.sep.stellar_web_authentication.`**`read_challenge_transaction`**(*challenge_transaction*, *server_account_id*, *network_passphrase*)

Reads a SEP 10 challenge transaction and returns the decoded transaction envelope and client account ID contained within.

It also verifies that transaction is signed by the server.

It does not verify that the transaction has been signed by the client or that any signatures other than the servers on the transaction are valid. Use one of the following functions to completely verify the transaction:

- *stellar_sdk.sep.stellar_web_authentication.verify_challenge_transaction_threshold()*
- *stellar_sdk.sep.stellar_web_authentication.verify_challenge_transaction_signers()*

Parameters

- **challenge_transaction** (str) – SEP0010 transaction challenge transaction in base64.
- **server_account_id** (str) – public key for server's account.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')

Raises *InvalidSep10ChallengeError* - if the validation fails, the exception will be thrown.

Return type Tuple[*TransactionEnvelope*, str]

`stellar_sdk.sep.stellar_web_authentication.`**`verify_challenge_transaction_threshold`**(*challenge_tr server_accou network_passph threshold, signers*)

Verifies that for a SEP 10 challenge transaction all signatures on the transaction are accounted for and that the signatures meet a threshold on an account. A transaction is verified if it is signed by the server account, and all other signatures match a signer that has been provided as an argument, and those signatures meet a threshold on the account.

Parameters

- **challenge_transaction** (str) – SEP0010 transaction challenge transaction in base64.
- **server_account_id** (str) – public key for server's account.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')
- **threshold** (int) – The medThreshold on the client account.
- **signers** (List[Ed25519PublicKeySigner]) – The signers of client account.

Raises *InvalidSep10ChallengeError*: - The transaction is invalid according to *stellar_sdk.sep.stellar_web_authentication.read_challenge_transaction()*. - One or more signatures in the transaction are not identifiable as the server account or one of the signers provided in the arguments. - The signatures are all valid but do not meet the threshold.

Return type `List[Ed25519PublicKeySigner]`

`stellar_sdk.sep.stellar_web_authentication.`**`verify_challenge_transaction_signed_by_client_ma`**

An alias for *stellar_sdk.sep.stellar_web_authentication.verify_challenge_transaction()*.

Parameters

- **challenge_transaction** (`str`) – SEP0010 transaction challenge transaction in base64.
- **server_account_id** (`str`) – public key for server's account.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')

Raises *InvalidSep10ChallengeError* - if the validation fails, the exception will be thrown.

Return type `None`

`stellar_sdk.sep.stellar_web_authentication.`**`verify_challenge_transaction_signers`**(*challenge_trans server_account net- work_passphra sign- ers*)

Verifies that for a SEP 10 challenge transaction all signatures on the transaction are accounted for. A transaction is verified if it is signed by the server account, and all other signatures match a signer that has been provided as an argument. Additional signers can be provided that do not have a signature, but all signatures must be matched to a signer for verification to succeed. If verification succeeds a list of signers that were found is returned, excluding the server account ID.

Parameters

- **challenge_transaction** (`str`) – SEP0010 transaction challenge transaction in base64.
- **server_account_id** (`str`) – public key for server's account.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')
- **signers** (`List[Ed25519PublicKeySigner]`) – The signers of client account.

Raises *InvalidSep10ChallengeError*: - The transaction is invalid according to *stellar_sdk.sep.stellar_web_authentication.read_challenge_transaction()*. - One or more signatures in the transaction are not identifiable as the server account or one of the signers provided in the arguments.

Return type `List[Ed25519PublicKeySigner]`

`stellar_sdk.sep.stellar_web_authentication.`**`verify_challenge_transaction`**(*challenge_transaction, server_account_id, net- work_passphrase*)

Verifies if a transaction is a valid SEP0010 v1.2 challenge transaction, if the validation fails, an exception will be thrown.

This function performs the following checks:

1. verify that transaction sequenceNumber is equal to zero;
2. verify that transaction source account is equal to the server's signing key;
3. verify that transaction has time bounds set, and that current time is between the minimum and maximum bounds;
4. verify that transaction contains a single Manage Data operation and it's source account is not null;
5. verify that transaction envelope has a correct signature by server's signing key;
6. verify that transaction envelope has a correct signature by the operation's source account;

Parameters

- **challenge_transaction** (str) – SEP0010 transaction challenge transaction in base64.
- **server_account_id** (str) – public key for server's account.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from. (ex. 'Public Global Stellar Network ; September 2015')

Raises *InvalidSep10ChallengeError* - if the validation fails, the exception will be thrown.

Return type None

Exceptions

class stellar_sdk.sep.exceptions.**StellarTomlNotFoundError**

If the SEP 0010 toml file not found, the exception will be thrown.

class stellar_sdk.sep.exceptions.**InvalidSep10ChallengeError**

If the SEP 0010 validation fails, the exception will be thrown.

CHAPTER 3

Links

- Document: https://stellar-sdk.readthedocs.io
- Code: https://github.com/StellarCN/py-stellar-base/tree/v2
- Docker: https://hub.docker.com/r/overcat/py-stellar-base
- Examples: https://github.com/StellarCN/py-stellar-base/blob/v2/examples
- Issue tracker: https://github.com/StellarCN/py-stellar-base/issues
- License: Apache License 2.0
- Releases: https://pypi.org/project/stellar-sdk/

CHAPTER 4

Thanks

This document is based on Stellar JavaScript SDK documentation. Thank you to all the people who have already contributed to Stellar ecosystem!

CHAPTER 5

genindex

Python Module Index

S

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

V

X