py-stellar-base Documentation

Release 14.1.0

Stellar Community

Jun 12, 2026

CONTENTS

py-stellar-base is a Python library for communicating with a Stellar Horizon server and Stellar RPC server. It is used for building Stellar apps on Python. It supports **Python 3.10+** as well as PyPy 3.11.

It provides:

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

CHAPTER

ONE

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 pip

Use pip to install and update py-stellar-base:

```
pip install -U stellar-sdk
```

The py-stellar-base release follows Semantic Versioning 2.0.0, and I strongly recommend that you specify its major version number in the dependency file to avoid the unknown effects of a corrupt update. 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
pip install .
```

1.2 Generate Keypair

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

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

```
from stellar_sdk import Keypair

secret = "SBK2VIYYSVG76E7VC3QHYARNFLY2EAQXDHRC7BMXBBGIFG74ARPRMNQM"
keypair = Keypair.from_secret(secret)

# GDHMW6QZOL73SHKG2JA3YHXFDHM46SS5ZRWEYF5BCYHX2C5TVO6KZBYL
public_key = keypair.public_key
```

(continues on next page)

(continued from previous page)

```

can_sign = keypair.can_sign()  # True
```

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

```
from stellar_sdk import Keypair

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

You can create a randomly generated keypair:

```
from stellar_sdk import Keypair

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

You can also generate a mnemonic phrase and later use it to generate a keypair:

```
from stellar_sdk import Keypair

mnemonic_phrase = Keypair.generate_mnemonic_phrase()
print(f"Mnemonic phrase: {mnemonic_phrase}")
keypair = Keypair.from_mnemonic_phrase(mnemonic_phrase)
print(f"Public Key: {keypair.public_key}")
print(f"Secret Seed: {keypair.secret}")
```

Lastly, you can also use the Shamir secret sharing method to split a mnemonic phrase into multiple phrases. In the following example, we need exactly 2 phrases in order to reconstruct the secret:

```
from stellar_sdk import Keypair

mnemonic_phrases = Keypair.generate_shamir_mnemonic_phrases(member_threshold=2, member_
↪count=3)
print(f"Mnemonic phrases: {mnemonic_phrases}")
keypair = Keypair.from_shamir_mnemonic_phrases(mnemonic_phrases[:2])  # any combinations
print(f"Public Key: {keypair.public_key}")
print(f"Secret Seed: {keypair.secret}")
```

If you want to convert an existing mnemonic phrase to Shamir, you need to get the corresponding entropy. You can use these lower level functions:

```
import shamir_mnemonic
from stellar_sdk.sep.mnemonic import StellarMnemonic

seed_raw = StellarMnemonic("english").to_entropy(mnemonic)
mnemonic_phrases = shamir_mnemonic.generate_mnemonics(
    group_threshold=1,
    groups=[(2, 3)],
    master_secret=seed_raw,
    passphrase=passphrase.encode(),
```

(continues on next page)

(continued from previous page)

```
)[0]
print(f"Mnemonic phrases: {mnemonic_phrases}")
```

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:

```
"""
=============================================================
Example: Activating a Stellar Account via Friendbot (Testnet)
=============================================================

This example demonstrates how to create and activate a Stellar account
on the Test Network using the Friendbot service.

Friendbot is a special service provided by the Stellar Testnet that
automatically funds new accounts with test XLM, allowing developers
to experiment without spending real money.

Steps performed:
1. Generate a new random keypair.
2. Request Friendbot to fund the account.
3. Print the public and secret keys for use in further operations.

Official Documentation:
https://developers.stellar.org/docs/tutorials/create-account/#create-account
"""

# ============================================================
# === Installation Guideline ================================
# ============================================================
# To run this example, install the Stellar SDK and Requests library:
#
#     pip install stellar-sdk requests
#
# Save this file as `friendbot_create_account.py` and execute:
#
#     python friendbot_create_account.py
#
# ============================================================
# === Import Required Libraries =============================
# ============================================================

# Import the 'requests' library to send HTTP requests to Friendbot
```

(continues on next page)

(continued from previous page)

```
import requests

# Import the 'Keypair' class to generate Stellar keypairs
from stellar_sdk import Keypair

# ============================================================
# === 1. Generate a Random Keypair ===========================
# ============================================================

print("=== Generate a Random Keypair ===")
# Generate a completely new random Stellar keypair
keypair = Keypair.random()

# Display the public and secret keys for the newly created account
print("Public Key: " + keypair.public_key)

# Print a truncated version of the secret for clarity, avoiding full exposure
# CodeQL [py/clear-text-logging-sensitive-data]: Safe truncated output for testnet␣
↪demonstration only.
print("Secret Seed: " + keypair.secret)
print("-" * 68)

# ============================================================
# === 2. Fund the Account using Friendbot ====================
# ============================================================

print("=== Requesting Funds from Friendbot ===")
# Friendbot is available only on the Stellar Testnet.
# It automatically funds accounts when given a valid public key.
url = "https://friendbot.stellar.org"

# Send a GET request to Friendbot with the account's public key as a parameter
response = requests.get(url, params={"addr": keypair.public_key})

# Print the Friendbot response - should confirm account creation and funding
print(response)
print("-" * 68)

# ============================================================
# === Expected Output ========================================
# ============================================================
# --------------------------------------------------------------------
# === Generate a Random Keypair ===
# Public Key: G*************** (randomly generated)
# Secret Seed: S*************** (randomly generated)
# --------------------------------------------------------------------
# === Requesting Funds from Friendbot ===
# <Response [200]>  # Indicates success
# --------------------------------------------------------------------
#
# Once the account is funded, you can check it on the Stellar Testnet Explorer:
# https://stellar.expert/explorer/testnet/account/G***************
```

(continues on next page)

(continued from previous page)

```
#
# The account is now active on the Stellar Testnet and ready for use.
# ----------------------------------------------------------------------
```

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:

```
"""
============================================================
Example: Creating and Funding a Stellar Account using Python SDK
============================================================

This example demonstrates how to create a new Stellar account and fund it
with a specified starting balance using the Stellar Testnet.

Operations performed:
1. Load an existing source account (already funded).
2. Generate a new random destination keypair.
3. Build a `Create Account` transaction to fund the new account.
4. Sign and submit the transaction.
5. Print transaction hash and new account credentials.

Official Documentation:
- Create Account: https://developers.stellar.org/docs/tutorials/create-account/#create-
↪account
- List of operations: https://developers.stellar.org/docs/start/list-of-operations/
↪#create-account
"""

# ============================================================
# === Installation Guideline ================================
# ============================================================
# To run this example, you need to install the Stellar SDK for Python:
#
#     pip install stellar-sdk
#
# Save this script as `create_account.py` and run with:
#
#     python create_account.py
#
# ============================================================
# === Import Required Libraries =============================
# ============================================================

from stellar_sdk import Keypair, Network, Server, TransactionBuilder

# ============================================================
# === 1. Setup Server and Source Account ====================
```

(continues on next page)

(continued from previous page)

```
# ============================================================

# Connect to the Stellar Testnet Horizon server
server = Server(horizon_url="https://horizon-testnet.stellar.org")

# Load an existing account that will fund the new account
# Replace this secret with your own testnet secret key
source = Keypair.from_secret("SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD")

# ============================================================
# === 2. Generate a New Random Keypair for Destination =========
# ============================================================

# Create a new random keypair for the account to be created
destination = Keypair.random()

# ============================================================
# === 3. Load Source Account Details ===========================
# ============================================================

# Load the source account from the Testnet Horizon server
source_account = server.load_account(account_id=source.public_key)

# ============================================================
# === 4. Build Create Account Transaction ======================
# ============================================================

# Build a transaction to create a new account
# and fund it with the starting balance of 12.25 XLM
transaction = (
    TransactionBuilder(
        source_account=source_account,
        network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
        base_fee=100,  # set fee per operation in stroops (0.00001 XLM)
    )
    .append_create_account_op(
        destination=destination.public_key,
        starting_balance="12.25",  # fund with 12.25 XLM
    )
    .set_timeout(30)  # transaction will timeout in 30 seconds
    .build()
)

# ============================================================
# === 5. Sign and Submit Transaction ===========================
# ============================================================

# Sign the transaction with the source account's secret key
transaction.sign(source)

# Submit the transaction to the Testnet
response = server.submit_transaction(transaction)
```

(continues on next page)

(continued from previous page)

```

# ============================================================
# === 6. Output Transaction and Account Details ==============
# ============================================================

print(f"Transaction hash: {response['hash']}")
print(
    f"New Keypair: \n\taccount id: {destination.public_key}\n\tsecret seed: {destination.
↪secret}"
)

# ============================================================
# === Expected Output ========================================
# ============================================================
# Example output (hash and keys will differ every run):
#
# Transaction hash: 70e023123fbf8b417ecea5ed923484e8d200beb792995d73d7692ab0875843b2
# New Keypair:
#       account id: GAPASWTZYIM2JZE5QLGID52PVD4QWCLCQLXWDSL7AA4ECES23WZGHMR4
#       secret seed: SB2TJS54Y2J52G5XVRJZLARUMYKSGAGQIG3NTWLR6SRY3MDQMGZUWJI2
#
# The new account has been created on the Stellar Testnet and funded
# with the starting balance specified.
# ------------------------------------------------------------
```

> **Note**
>
> To avoid risks, `TESTNET` is used in the example above. In order to use the Stellar Live Network you will have to change the network passphrase to `Network.PUBLIC_NETWORK_PASSPHRASE` and the server URL to point to `https://horizon.stellar.org` too.

1.4 Querying Horizon

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

1.4.1 Building requests

py-stellar-base 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.)

```
"""
See: https://stellar-sdk.readthedocs.io/en/latest/querying_horizon.html#building-requests
"""

from stellar_sdk import Server

server = Server(horizon_url="https://horizon.stellar.org")
account = "GB6NVEN5HSUBKMYCE5ZOWSK5K23TBWRUQLZY3KNMXUZ3AQ2ESC4MY4AQ"

# get a list of transactions that occurred in ledger 1400
```

(continues on next page)

(continued from previous page)

```
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=account).call()
print(transactions)

# The following example will show you how to handle paging
print(f"Gets all payment operations associated with {account}.")
payments_records = []
payments_call_builder = (
    server.payments().for_account(account).order(desc=False).limit(10)
)  # limit can be set to a maximum of 200
payments_records += payments_call_builder.call()["_embedded"]["records"]
page_count = 0
while page_records := payments_call_builder.next()["_embedded"]["records"]:
    payments_records += page_records
    print(f"Page {page_count} fetched")
    print(f"data: {page_records}")
    page_count += 1
print(f"Payments count: {len(payments_records)}")
```

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:

```
"""
See: https://stellar-sdk.readthedocs.io/en/latest/querying_horizon.html#streaming-
↪requests
"""

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.

```
"""
This example demonstrates how to use TransactionBuilder
to quickly build a transaction. For a beginner,
most of the work can be done with TransactionBuilder.

See: https://stellar-sdk.readthedocs.io/en/latest/building_transactions.html#building-
↪transactions
"""

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

root_keypair = Keypair.from_secret(
    "SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC"
)
# Create an Account object from an address and sequence number.
root_account = Account(account=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=Asset.native(),
        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.

```
"""
This example shows how to add memo to a transaction.

See: https://developers.stellar.org/docs/glossary/transactions/#memo
See: https://stellar-sdk.readthedocs.io/en/latest/building_transactions.html#building-
↪transactions
"""

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

root_keypair = Keypair.from_secret(
    "SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOUXSQA7BOYN7XHC"
)
# Create an Account object from an address and sequence number.
root_account = Account(account=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=Asset.native(),
    )
    .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.

```
"""
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.

See: https://developers.stellar.org/docs/start/list-of-operations/#payment
"""

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

# The source account is the account we will be signing and sending from.
# Derive Keypair object and public key (that starts with a G) from the secret
source_keypair = Keypair.from_secret(
    "SCDG4ORIDX4QGPMMHQY36KDHHMTJEM4RQ2AWKH3G7AXHTVBJWEV6XOUM"
)
source_public_key = source_keypair.public_key

# We are sending lumen to the receiver account
receiver_public_key = "GD2JXEFGEO53CNQ22KN2ICOQ2LOASCABQHAIOMLZV265C246PFKKHPYU"

# 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 = 100
# 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(
```

(continues on next page)

(continued from previous page)

```
        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, Asset.native(), "350.1234567")
    .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 = server.submit_transaction(transaction)
print(response)
```

1.7.2 Path Payment

In the example below we're sending 1000 XLM (at max) from *GABJLI6IVBKJ7HIC5NN7HHDCIEW3CMWQ2DWYHREQQUFWSWZ2C* 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.

```
"""
A path payment sends an amount of a specific asset to a destination account through a␣
↪path of offers.
Since the asset sent (e.g., 450 XLM) can be different from the asset received (e.g, 6␣
↪BTC),
path payments allow for the simultaneous transfer and conversion of currencies.

A Path Payment Strict Send allows a user to specify the amount of the asset to send.
The amount received will vary based on offers in the order books. If you would like to
instead specify the amount received, use the Path Payment Strict Receive operation.

See: https://developers.stellar.org/docs/start/list-of-operations/#path-payment-strict-
```

(continues on next page)

(continued from previous page)

```
↪send
See: https://youtu.be/KzlSgSPStz8
"""

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

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_strict_receive_op(
        destination="GBBM6BKZPEHWYO3E3YKREDPQXMS4VK35YLNU7NFBRI26RAN7GI5POFBB",
        send_asset=Asset.native(),
        send_max="1000",
        dest_asset=Asset(
            "GBP", "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-base 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.

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.
```

(continues on next page)

(continued from previous page)

```
5
6   Assumes that you have the following items:
7   1. Secret key of a funded account to be the source account
8   2. Public key of an existing account as a recipient
9       These two keys can be created and funded by the friendbot at
10      https://www.stellar.org/laboratory/ under the heading "Quick Start: Test Account"
11  3. Access to Python Stellar SDK (https://github.com/StellarCN/py-stellar-base) through␣
    ↪Python shell.
12
13  See: https://developers.stellar.org/docs/start/list-of-operations/#payment
14  """
15
16  import asyncio
17
18  from stellar_sdk import (
19      AiohttpClient,
20      Asset,
21      Keypair,
22      Network,
23      ServerAsync,
24      TransactionBuilder,
25  )
26
27  # The source account is the account we will be signing and sending from.
28  # Derive Keypair object and public key (that starts with a G) from the secret
29  source_keypair = Keypair.from_secret(
30      "SCDG4ORIDX4QGPMMHQY36KDHHMTJEM4RQ2AWKH3G7AXHTVBJWEV6XOUM"
31  )
32  source_public_key = source_keypair.public_key
33
34  # We are sending lumen to the receiver account
35  receiver_public_key = "GD2JXEFGEO53CNQ22KN2ICOQ2LOASCABQHAIOMLZV265C246PFKKHPYU"
36
37
38  async def main():
39      # Configure StellarSdk to talk to the horizon instance hosted by Stellar.org
40      # To use the live network, set the hostname to 'horizon.stellar.org'
41      # When we use the `with` syntax, it automatically releases the resources it occupies.
42      async with ServerAsync(
43          horizon_url="https://horizon-testnet.stellar.org", client=AiohttpClient()
44      ) as server:
45          # Transactions require a valid sequence number that is specific to this account.
46          # We can fetch the current sequence number for the source account from Horizon.
47          source_account = await server.load_account(source_public_key)
48
49          base_fee = 100
50          # we are going to submit the transaction to the test network,
51          # so network_passphrase is `Network.TESTNET_NETWORK_PASSPHRASE`,
52          # if you want to submit to the public network, please use `Network.PUBLIC_
    ↪NETWORK_PASSPHRASE`.
53          transaction = (
54              TransactionBuilder(
```

(continues on next page)

(continued from previous page)

```
                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, Asset.native(), "350.1234567")
            .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__":
    asyncio.run(main())
```

The following example helps you listen to multiple endpoints asynchronously.

```
"""
See: https://stellar-sdk.readthedocs.io/en/latest/asynchronous.html
"""

import asyncio

from stellar_sdk import AiohttpClient, ServerAsync

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


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


async def effects():
```

(continues on next page)

(continued from previous page)

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


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


async def transactions():
    async with ServerAsync(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

```
"""
Stellar uses signatures as authorization. Transactions always need authorization
from at least one public key in order to be considered valid. Generally,
transactions only need authorization from the public key of the source account.

Transaction signatures are created by cryptographically signing the
transaction object contents with a secret key. Stellar currently uses the ed25519␣
```

(continues on next page)

(continued from previous page)

```
↪signature
scheme, but there's also a mechanism for adding additional types of public/private key␣
↪schemes.
A transaction with an attached signature is considered to have authorization from that␣
↪public key.

In two cases, a transaction may need more than one signature. If the transaction has
operations that affect more than one account, it will need authorization from every␣
↪account
in question. A transaction will also need additional signatures if the account associated
with the transaction has multiple public keys.

See: https://developers.stellar.org/docs/glossary/multisig/
"""

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

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

(continues on next page)

(continued from previous page)

```

# 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=Asset.native())
    .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)
```

1.10 XDR

XDR, also known as External Data Representation, is used throughout the Stellar network and protocol. The ledger, transactions, results, history, and even the messages passed between computers running stellar-core are encoded using XDR.

stellar_sdk.xdr module provides a complete ability to build and parse XDR.

This example shows how to parse XDR string into an XDR object.

```
"""
This example shows how to parse XDR string into an XDR object.
But please note that if you need to parse a transaction envelope,
please refer to `parse_transaction_envelope.py`
"""

from stellar_sdk.xdr import TransactionResult

result_xdr = "AAAAAAAAAGQAAAAAAAAAAQAAAAAAAAADAAAAAAAAAAAAAAAABAAAAAD/
↪jlpBCTX53ogvts02Ryn5GjO6gx0qW3/3ARB+gOh/nAAAAADGRC/
↪wAAAAAAAAAAU5VQwAAAAAAR74W04RzO2ryJo94Oi0FUs0KHIVQisRnpe9FWrqvumQAAAAAAEFWjwjgcksQkG4uAAAAAAAAAAAAAAAA
↪"
transaction_result = TransactionResult.from_xdr(result_xdr)
print(transaction_result.fee_charged)
print(transaction_result.result.code)
```

CHAPTER

TWO

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*, *sequence*, *raw_data=None*)

The *Account* object represents a single account on the Stellar network and its sequence number.

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

Normally, you can get an *Account* instance through *stellar_sdk.server.Server.load_account()* or *stellar_sdk.server_async.ServerAsync.load_account()*.

An example:

```
from stellar_sdk import Keypair, Server

server = Server(horizon_url="https://horizon-testnet.stellar.org")
source = Keypair.from_secret(
↪"SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD")
# `account` can also be a muxed account
source_account = server.load_account(account=source.public_key)
```

See Accounts for more information.

Parameters

- **account** (str | *MuxedAccount*) – Account Id of the account (ex. "GB3KJPLFUYN5VL6R3GU3EGCGVCKFDSD7BEDX42HWG5BWFKB3KQGJJRMA") or muxed account (ex. "MBZSQ3YZMZEWL5ZRCEQ5CCSOTXCFCMKDGFFP4IEQN2KN6LCHCLI46AAAAAAAAAAE2L2QE")
- **sequence** (int) – Current sequence number of the account.
- **raw_data** (dict[str, Any] | None) – Raw horizon response data.

increment_sequence_number()

Increments sequence number in this object by one.

Return type
None

load_ed25519_public_key_signers()

Load ed25519 public key signers.

Return type
list[Ed25519PublicKeySigner]

property universal_account_id: str

Get the universal account id, if *account* is ed25519 public key, it will return ed25519 public key (ex. "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD"), otherwise it will return muxed account (ex. "MAAAAAAAAAAAJURAAB2X52XFQP6FBXLGT6LWOOWMEXWHEWBDVRZ7V5WH34Y22MPFBHUHY")

2.1.2 Address

class stellar_sdk.address.**Address**(*address*)

Represents a single address in the Stellar network. An address can represent an account or a contract.

Parameters
address (str) – ID of the account, contract, muxed account, claimable balance, or liquidity pool.

property address: str

Returns the encoded address.

Returns
The encoded address.

static from_raw_account(*account*)

Creates a new account Address object from raw bytes.

Parameters
account (bytes | str) – The raw bytes of the account, it can be a byte array or a hex encoded string.

Return type
Address

Returns
A new Address object.

static from_raw_claimable_balance(*claimable_balance*)

Creates a new claimable balance Address object from raw bytes.

Parameters
claimable_balance (bytes | str) – The raw bytes of the claimable balance.

Return type
Address

Returns
A new Address object.

static from_raw_contract(*contract*)

Creates a new contract Address object from a buffer of raw bytes.

Parameters
contract (bytes | str) – The raw bytes of the contract.

Return type
Address

Returns
A new Address object.

static from_raw_liquidity_pool(*liquidity_pool*)

Creates a new liquidity pool Address object from raw bytes.

Parameters
liquidity_pool (bytes | str) – The raw bytes of the liquidity pool.

Return type
Address

Returns
A new Address object.

static from_raw_muxed_account(*muxed_account*)

Creates a new muxed account Address object from raw bytes.

Parameters
muxed_account (bytes | str) – The raw bytes of the muxed account.

Return type
Address

Returns
A new Address object.

classmethod from_xdr_sc_address(*sc_address*)

Creates a new Address object from a stellar_sdk.xdr.SCAddress XDR object.

Parameters
sc_address (*SCAddress*) – The stellar_sdk.xdr.SCAddress XDR object.

Return type
Address

Returns
A new Address object.

to_xdr_sc_address()

Converts the Address object to a stellar_sdk.xdr.SCAddress XDR object.

Return type
SCAddress

Returns
A stellar_sdk.xdr.SCAddress XDR object.

class stellar_sdk.address.**AddressType**(**values*)

Represents an Address type.

ACCOUNT = 0

An account address, address looks like GBJCHUKZMTFSLOMNC7P4TS4VJJBTCYL3XKSOLXAUJSD56C4LHND5T.
...

CLAIMABLE_BALANCE = 3

A claimable balance address, address looks like BAAD6DBUX6J22DMZOHIEZTEQ64CVCHEDRKWZONFEUL5Q26QD7R76RGR.
...

CONTRACT = 1

An contract address, address looks like CCJZ5DGASBWQXR5MPFCJXMBI333XE5U3FSJTNQU7RIKE3P5GN2K2W.
...

LIQUIDITY_POOL = 4

A liquidity pool address, address looks like `LA7QYNF7SOWQ3GLR2BGMZEHXAVIRZA4KVWLTJJFC7MGXUA74P7UJU...`

MUXED_ACCOUNT = 2

A muxed account address, address looks like `MAQAA5L65LSYH7CQ3VTJ7F3HHLGCL3DSLAR2Y47263D56MNNGHSQSAAAAAAAAA...`

2.1.3 Asset

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

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

The following example shows how to create an Asset object:

```
from stellar_sdk import Asset

native_asset = Asset.native()  # You can also create a native asset through␣
↪Asset("XLM").
credit_alphanum4_asset = Asset("USD",
↪"GBSKJPM2FM6O2C6GVZNAUAMGXZ6I4QIUPMNWVDN2NZULPWWTV3GI2SOX")
credit_alphanum12_asset = Asset("BANANA",
↪"GA6VT2PDD73TNNRYLPJPJYAAI7EGKBATZ7V562S7XY7TJD4GNOXRG6OS")
print(f"Asset type: {credit_alphanum4_asset.type}
```

“

f”Asset code: {credit_alphanum4_asset.code}

“

f”Asset issuer: {credit_alphanum4_asset.issuer}

“

f”Is native asset: {credit_alphanum4_asset.is_native()}”)

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** (str | None) –

  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.

static check_if_asset_code_is_valid(*code*)

Check whether the *code* passed in by the user is a valid asset code, if not, an exception will be thrown.

Parameters
code (str) – The asset code.

Raises
AssetCodeInvalidError: if code is invalid.

Return type
None

contract_id(*network_passphrase*)

Return the contract Id for the asset contract.

Parameters
network_passphrase (str) – The network where the asset is located.

Return type
str

Returns
The contract Id for the asset contract.

classmethod from_xdr_object(*xdr_object*)

Create a *Asset* from an XDR Asset/ChangeTrustAsset/TrustLineAsset object.

Please note that this function only supports processing the following types of assets:

- ASSET_TYPE_NATIVE
- ASSET_TYPE_CREDIT_ALPHANUM4
- ASSET_TYPE_CREDIT_ALPHANUM12

Parameters
xdr_object (*Asset* | *ChangeTrustAsset* | *TrustLineAsset*) – The XDR Asset/ChangeTrustAsset/TrustLineAsset object.

Return type
Asset

Returns
A new *Asset* object from the given XDR 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 Ture if the *Asset* object is the native asset.

Return type
bool

Returns
True if the asset object is native, False otherwise.

classmethod native()

Returns an asset object for the native asset.

Return type
Asset

Returns
An asset object for the native asset.

to_change_trust_asset_xdr_object()

Returns the xdr object for this asset.

Return type
ChangeTrustAsset

Returns
XDR ChangeTrustAsset object

to_dict()

Generate a dict for this object's attributes.

Return type
dict[str, str | None]

Returns
A dict representing an *Asset*

to_trust_line_asset_xdr_object()

Returns the xdr object for this asset.

Return type
TrustLineAsset

Returns
XDR TrustLineAsset object

to_xdr_object()

Returns the xdr object for this asset.

Return type
Asset

Returns
XDR Asset object

property type: str

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

Returns
The type of the asset.

2.1.4 Call Builder

AccountsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.accounts()`.

See List All Accounts for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*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 Retrieve an Account for more information.

Parameters
account_id (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI

Returns
current AccountCallBuilder instance

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_asset(*asset*)

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

See List All Accounts for more information.

Parameters
asset (*Asset*) – an issued asset

Returns
current AccountCallBuilder instance

for_liquidity_pool(*liquidity_pool_id*)

Filtering accounts who have a trustline for the given pool. The result is a list of accounts.

See List All Accounts for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string., for example: "dd7b1ab831c273310ddbec6f97870aa83c2fbd78ce22aded37ecbf4f3380fac7"

Returns
current AccountCallBuilder instance

for_signer(*signer*)

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

See List All Accounts for more information.

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

Returns
current AccountCallBuilder instance

for_sponsor(*sponsor*)

Filtering accounts where the given account is sponsoring the account or any of its sub-entries.

See List All Accounts for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Accounts* endpoint.

See Streaming for more information.

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

AssetsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.assets()`.

See List All Assets for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_code(*asset_code*)

This endpoint filters all assets by the asset code.

See List All Assets for more information.

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

Returns
current AssetCallBuilder instance

for_issuer(*asset_issuer*)

This endpoint filters all assets by the asset issuer.

See List All Assets for more information.

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

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 Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

ClaimableBalancesCallBuilder

`class stellar_sdk.call_builder.call_builder_sync.`**ClaimableBalancesCallBuilder**(*horizon_url*, *client*)

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

See List Claimable Balances for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

`call()`

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

Return type
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

`claimable_balance(claimable_balance_id)`

Returns information and links relating to a single claimable balance.

See Retrieve a Claimable Balance for more information.

Parameters
claimable_balance_id (str) – claimable balance id

Returns
current AccountCallBuilder instance

`cursor(cursor)`

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

See Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

`for_asset(asset)`

Returns all claimable balances which provide a balance for the given asset.

See List Claimable Balances for more information.

Parameters
asset (*Asset*) – an asset

Returns
current ClaimableBalancesCallBuilder instance

`for_claimant(claimant)`

Returns all claimable balances which can be claimed by the given account ID.

See List Claimable Balances for more information.

Parameters
claimant (str) – the account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD"

Returns
current ClaimableBalancesCallBuilder instance

`for_sponsor`(*sponsor*)

Returns all claimable balances which are sponsored by the given account ID.

See List Claimable Balances for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

Returns
current ClaimableBalancesCallBuilder instance

`limit`(*limit*)

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

See Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

DataCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.data()`.

See Retrieve an Account's Data for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.
- **account_id** (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI
- **data_name** (str) – Key name

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Account Data* endpoint.

See Streaming for more information.

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

EffectsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.effects().

See List All Effects for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Retrieve an Account's Effects for more information.

Parameters
account_id (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI

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 Retrieve a Ledger's Effects for more information.

Parameters
sequence (int | str) – ledger sequence

Returns
this EffectCallBuilder instance

`for_liquidity_pool`(*liquidity_pool_id*)

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

See Liquidity Pools - Retrieve related Effects for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this EffectsCallBuilder instance

`for_operation`(*operation_id*)

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

See Retrieve an Operation's Effects for more information.

Parameters
operation_id (int | str) – operation ID

Returns
this EffectCallBuilder instance

`for_transaction`(*transaction_hash*)

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

See Retrieve a Transaction's Effects for more information.

Parameters
transaction_hash (str) – transaction hash

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 Pagination

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

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.

Returns
current CallBuilder instance

`stream()`

Creates an EventSource that listens for events from the *Effects* endpoint.

See Streaming for more information.

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

FeeStatsCallBuilder

`class stellar_sdk.call_builder.call_builder_sync.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.fee_stats()`.

See Fee Stats for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

`call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

This endpoint does not support streaming.

LedgersCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.ledgers()`.

See List All Ledgers for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

`ledger`(*sequence*)

Provides information on a single ledger.

See Retrieve a Ledger for more information.

Parameters
sequence (int | str) – Ledger sequence

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 Pagination

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

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`.

Returns
current CallBuilder instance

`stream`()

Creates an EventSource that listens for events from the *Ledgers* endpoint.

See Streaming for more information.

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

LiquidityPoolsBuilder

class stellar_sdk.call_builder.call_builder_sync.**LiquidityPoolsBuilder**(*horizon_url*, *client*)

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

See List Liquidity Pools for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

`call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

`for_account(account_id)`

Filter pools for a specific account

See List Liquidity Pools for more information.

Parameters
account_id (str) – account id

Returns
current LiquidityPoolsBuilder instance

`for_reserves(reserves)`

Get pools by reserves.

Horizon will provide an endpoint to find all liquidity pools which contain a given set of reserve assets.

See List Liquidity Pools for more information.

Returns
current LiquidityPoolsBuilder instance

`limit(limit)`

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

See Pagination

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

Returns

liquidity_pool(*liquidity_pool_id*)

Provides information on a liquidity pool.

See Retrieve a Liquidity Pool for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
current LiquidityPoolsBuilder 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.

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
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.

Raise
StreamClientError - Failed to fetch stream resource.

OffersCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.offers().

See List All Offers for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

This endpoint represents all offers a given account has currently open and can be used in streaming mode.

See Retrieve an Account's Offers for more information.

Parameters
account_id (str) – Account ID

Returns
current PaymentsCallBuilder 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 List All Offers for more information.

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 List All Offers for more information.

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 List All Offers for more information.

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

Returns
this OffersCallBuilder instance

for_sponsor(*sponsor*)

Filtering offers where the given account is sponsoring the offer entry.

See List All Offers for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

Returns
current OffersCallBuilder instance

limit(*limit*)

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

See Pagination

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

Returns

offer(*offer_id*)

Returns information and links relating to a single offer.

See Retrieve an Offer for more information.

Parameters
offer_id (int | str) – 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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Offers* endpoint.

See Streaming for more information.

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

OperationsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.operations()`.

See List All Operations for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Retrieve an Account's Operations for more information.

Parameters
account_id (str) – Account ID

Returns
this OperationCallBuilder instance

for_claimable_balance(*claimable_balance_id*)

This endpoint represents successful operations referencing a given claimable balance and can be used in streaming mode.

See Claimable Balances - Retrieve related Operations for more information.

Parameters
claimable_balance_id (str) – This claimable balance's id encoded in a hex string representation.

Returns
this OperationCallBuilder instance

for_ledger(*sequence*)

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

See Retrieve a Ledger's Operations for more information.

Parameters
sequence (int | str) – Sequence ID

Returns
this OperationCallBuilder instance

for_liquidity_pool(*liquidity_pool_id*)

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

See Liquidity Pools - Retrieve related Operations for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this OperationCallBuilder instance

for_transaction(*transaction_hash*)

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

See Retrieve a Transaction's Operations for more information.

Parameters
transaction_hash (str) – Transaction Hash

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.

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*

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 Pagination

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

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 Retrieve an Operation for more information.

Parameters
operation_id (int | str) – Operation ID

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.

Returns
current CallBuilder instance

`stream`()

Creates an EventSource that listens for events from the *Operations* endpoint.

See Streaming for more information.

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

OrderbookCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.orderbook()`.

See Orderbook for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*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
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Orderbook* endpoint.

See Streaming for more information.

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

PaymentsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.payments()`.

See List All Payments for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

`call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Retrieve an Account's Payments for more information.

Parameters
account_id (str) – Account ID

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 Retrieve a Ledger's Payments for more information.

Parameters
sequence (int | str) – Ledger sequence

Returns
current PaymentsCallBuilder instance

for_transaction(*transaction_hash*)

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

See Retrieve a Transaction's Payments for more information.

Parameters
transaction_hash (str) – Transaction hash

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.

Returns
current PaymentsCallBuilder 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*

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Payments* endpoint.

See Streaming for more information.

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

RootCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.root()`.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
`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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

StrictReceivePathsCallBuilder

`class stellar_sdk.call_builder.call_builder_sync.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.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 List Strict Receive Payment Paths for more information.

Parameters

- **horizon_url** (`str`) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.
- **source** (`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 | Decimal`) – 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
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

StrictSendPathsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.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 List Strict Send Payment Paths for more information.

Parameters

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

call()

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

Return type
`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 Pagination

Parameters

cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

Parameters

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

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`.

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

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`.

Raise

StreamClientError - Failed to fetch stream resource.

TradeAggregationsCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.trade_aggregations()`.

Trade Aggregations facilitate efficient gathering of historical trade data.

See List Trade Aggregations for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*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** (int | None) – lower time boundary represented as millis since epoch
- **end_time** (int | None) – upper time boundary represented as millis since epoch
- **offset** (int | None) – 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
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
`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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

TradesCallBuilder

class stellar_sdk.call_builder.call_builder_sync.**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.trades()`.

See List All Trades for more information.

Parameters

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

call()

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

Return type
`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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

Filter trades for a specific account

See Retrieve an Account's Trades for more information.

Parameters
account_id (str) – account id

Returns
current TradesCallBuilder instance

for_asset_pair(*base*, *counter*)

Filter trades for a specific asset pair (orderbook)

See List All Trades for more information.

Parameters

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

Returns
current TradesCallBuilder instance

for_liquidity_pool(*liquidity_pool_id*)

Filter trades for a specific liquidity pool.

See Liquidity Pools - Retrieve related Trades

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
current TradesCallBuilder instance

for_offer(*offer_id*)

Filter trades for a specific offer

See List All Trades for more information.

Parameters
offer_id (int | str) – offer id

Returns
current TradesCallBuilder instance

`for_trade_type`(*trade_type*)

Filter trades for a specific trade type

Horizon will reject requests which attempt to set *trade_type* to `liquidity_pools` when using the offer id filter.

Parameters
trade_type (str) – trade type, the currently supported types are `"orderbook"`, `"liquidity_pool"` and `"all"`, defaults to `"all"`.

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 Pagination

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

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`.

Returns
current CallBuilder instance

`stream`()

Creates an EventSource that listens for events from the *Trades* endpoint.

See Streaming for more information.

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

TransactionsCallBuilder

`class stellar_sdk.call_builder.call_builder_sync.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.transactions()`.

See List All Transactions for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseSyncClient*) – The client instance used to send request.

`call`()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

This endpoint represents all transactions that affected a given account.

See Retrieve an Account's Transactions for more information.

Parameters
account_id (str) – account id

Returns
current TransactionsCallBuilder instance

for_claimable_balance(*claimable_balance_id*)

This endpoint represents all transactions referencing a given claimable balance and can be used in streaming mode.

See Claimable Balances - Retrieve related Transactions

Parameters
claimable_balance_id (str) – This claimable balance's id encoded in a hex string representation.

Returns
current TransactionsCallBuilder instance

for_ledger(*sequence*)

This endpoint represents all transactions in a given ledger.

See Retrieve a Ledger's Transactions for more information.

Parameters
sequence (int | str) – ledger sequence

Returns
current TransactionsCallBuilder instance

for_liquidity_pool(*liquidity_pool_id*)

This endpoint represents all transactions referencing a given liquidity pool.

See Liquidity Pools - Retrieve related Transactions

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this 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.

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Transactions* endpoint.

See Streaming for more information.

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

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 Retrieve a Transaction for more information.

Parameters
transaction_hash (str) – transaction hash

Returns
current TransactionsCallBuilder instance

2.1.5 Call Builder Async

AccountsCallBuilder

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

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

See List All Accounts for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*BaseAsyncClient*) – 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 Retrieve an Account for more information.

Parameters

account_id (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI

Returns

current AccountCallBuilder instance

async call()

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

Return type

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 Pagination

Parameters

cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns

current CallBuilder instance

for_asset(*asset*)

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

See List All Accounts for more information.

Parameters
asset (*Asset*) – an issued asset

Returns
current AccountCallBuilder instance

for_liquidity_pool(*liquidity_pool_id*)

Filtering accounts who have a trustline for the given pool. The result is a list of accounts.

See List All Accounts for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string., for example: "dd7b1ab831c273310ddbec6f97870aa83c2fbd78ce22aded37ecbf4f3380fac7"

Returns
current AccountCallBuilder instance

for_signer(*signer*)

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

See List All Accounts for more information.

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

Returns
current AccountCallBuilder instance

for_sponsor(*sponsor*)

Filtering accounts where the given account is sponsoring the account or any of its sub-entries.

See List All Accounts for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Accounts* endpoint.

See Streaming for more information.

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

AssetsCallBuilder

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

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

See Assets for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_code(*asset_code*)

This endpoint filters all assets by the asset code.

See List All Assets for more information.

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

Returns
current AssetCallBuilder instance

`for_issuer`(*asset_issuer*)

This endpoint filters all assets by the asset issuer.

See List All Assets for more information.

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

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 Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

ClaimableBalancesCallBuilder

`class` stellar_sdk.call_builder.call_builder_async.`ClaimableBalancesCallBuilder`(*horizon_url*, *client*)

Creates a new *ClaimableBalancesCallBuilder* pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.ServerAsync.claimable_balance()`.

See List Claimable Balances for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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

claimable_balance(*claimable_balance_id*)

Returns information and links relating to a single claimable balance.

See Retrieve a Claimable Balance for more information.

Parameters
claimable_balance_id (str) – claimable balance id

Returns
current AccountCallBuilder instance

cursor(*cursor*)

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

See Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_asset(*asset*)

Returns all claimable balances which provide a balance for the given asset.

See List Claimable Balances for more information.

Parameters
asset (*Asset*) – an asset

Returns
current ClaimableBalancesCallBuilder instance

for_claimant(*claimant*)

Returns all claimable balances which can be claimed by the given account ID.

See List Claimable Balances for more information.

Parameters
claimant (str) – the account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD"

Returns
current ClaimableBalancesCallBuilder instance

`for_sponsor`(*sponsor*)

Returns all claimable balances which are sponsored by the given account ID.

See List Claimable Balances for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

Returns
current ClaimableBalancesCallBuilder instance

`limit`(*limit*)

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

See Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

DataCallBuilder

`class` stellar_sdk.call_builder.call_builder_async.**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.ServerAsync.data()`.

See Retrieve an Account's Data for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.
- **account_id** (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI
- **data_name** (str) – Key name

`async call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Account Data* endpoint.

See Streaming for more information.

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

EffectsCallBuilder

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

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

See List All Effects for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Retrieve an Account's Effects for more information.

Parameters
account_id (str) – account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BI

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 Retrieve a Ledger's Effects for more information.

Parameters
sequence (int | str) – ledger sequence

Returns
this EffectCallBuilder instance

`for_liquidity_pool`(*liquidity_pool_id*)

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

See Liquidity Pools - Retrieve related Effects for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this EffectsCallBuilder instance

`for_operation`(*operation_id*)

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

See Retrieve an Operation's Effects for more information.

Parameters
operation_id (int | str) – operation ID

Returns
this EffectCallBuilder instance

`for_transaction`(*transaction_hash*)

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

See Retrieve a Transaction's Effects for more information.

Parameters
transaction_hash (str) – transaction hash

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 Pagination

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

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.

Returns
current CallBuilder instance

`stream()`

Creates an EventSource that listens for events from the *Effects* endpoint.

See Streaming for more information.

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

FeeStatsCallBuilder

`class stellar_sdk.call_builder.call_builder_async.FeeStatsCallBuilder(horizon_url, client)`

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

See Fee Stats for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

`async call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
`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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

LedgersCallBuilder

class stellar_sdk.call_builder.call_builder_async.**LedgersCallBuilder**(*horizon_url*, *client*)

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

See List All Ledgers for more information.

Parameters

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

async call()

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

Return type
`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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

ledger(*sequence*)

Provides information on a single ledger.

See Retrieve a Ledger for more information.

Parameters
sequence (int | str) – Ledger sequence

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Ledgers* endpoint.

See Streaming for more information.

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

LiquidityPoolsBuilder

class stellar_sdk.call_builder.call_builder_async.**LiquidityPoolsBuilder**(*horizon_url*, *client*)

Creates a new *LiquidityPoolsBuilder* pointed to server defined by horizon_url. Do not create this object directly, use `stellar_sdk.ServerAsync.liquidity_pools()`.

See List Liquidity Pools for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

Filter pools for a specific account

See List Liquidity Pools for more information.

Parameters
account_id (str) – account id

Returns
current LiquidityPoolsBuilder instance

for_reserves(*reserves*)

Get pools by reserves.

Horizon will provide an endpoint to find all liquidity pools which contain a given set of reserve assets.

See List Liquidity Pools for more information.

Returns
current LiquidityPoolsBuilder instance

`limit`(*limit*)

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

See Pagination

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

Returns

`liquidity_pool`(*liquidity_pool_id*)

Provides information on a liquidity pool.

See Retrieve a Liquidity Pool for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
current LiquidityPoolsBuilder 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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

OffersCallBuilder

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

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

See List All Offers for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.

- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

This endpoint represents all offers a given account has currently open and can be used in streaming mode.

See Retrieve an Account's Offers for more information.

Parameters
account_id (str) – Account ID

Returns
current PaymentsCallBuilder 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 List All Offers for more information.

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 List All Offers for more information.

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 List All Offers for more information.

Parameters
selling (Asset) – The asset being sold.

Returns
this OffersCallBuilder instance

for_sponsor(*sponsor*)

Filtering offers where the given account is sponsoring the offer entry.

See List All Offers for more information.

Parameters
sponsor (str) – the sponsor id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BR

Returns
current OffersCallBuilder instance

limit(*limit*)

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

See Pagination

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

Returns

offer(*offer_id*)

Returns information and links relating to a single offer.

See Retrieve an Offer for more information.

Parameters
offer_id (int | str) – 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`.

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
`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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

OperationsCallBuilder

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

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

See List All Operations for more information.

Parameters

- **horizon_url** – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

`async call()`

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

Return type
`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 Pagination

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

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 Retrieve an Account's Operations for more information.

Parameters
account_id (str) – Account ID

Returns
this OperationCallBuilder instance

`for_claimable_balance`(*claimable_balance_id*)

This endpoint represents successful operations referencing a given claimable balance and can be used in streaming mode.

See Claimable Balances - Retrieve related Operations for more information.

Parameters
claimable_balance_id (str) – This claimable balance's id encoded in a hex string representation.

Returns
this OperationCallBuilder instance

`for_ledger`(*sequence*)

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

See Retrieve a Ledger's Operations for more information.

Parameters
sequence (int | str) – Sequence ID

Returns
this OperationCallBuilder instance

`for_liquidity_pool`(*liquidity_pool_id*)

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

See Liquidity Pools - Retrieve related Operations for more information.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this OperationCallBuilder instance

`for_transaction`(*transaction_hash*)

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

See Retrieve a Transaction's Operations for more information.

Parameters
transaction_hash (str) – Transaction Hash

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.

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*

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 Pagination

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

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 Retrieve an Operation for more information.

Parameters
operation_id (int | str) – Operation ID

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Operations* endpoint.

See Streaming for more information.

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

OrderbookCallBuilder

class stellar_sdk.call_builder.call_builder_async.**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.ServerAsync.orderbook()`.

See Orderbook for more information.

Parameters

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

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Orderbook* endpoint.

See Streaming for more information.

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

PaymentsCallBuilder

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

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

See List All Payments for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Retrieve an Account's Payments for more information.

Parameters
account_id (str) – Account ID

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 Retrieve a Ledger's Payments for more information.

Parameters
sequence (int | str) – Ledger sequence

Returns
current PaymentsCallBuilder instance

for_transaction(*transaction_hash*)

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

See Retrieve a Transaction's Payments for more information.

Parameters
transaction_hash (str) – Transaction hash

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.

Returns
current PaymentsCallBuilder 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*

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Payments* endpoint.

See Streaming for more information.

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

RootCallBuilder

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

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

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

StrictReceivePathsCallBuilder

class stellar_sdk.call_builder.call_builder_async.**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.ServerAsync.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 List Strict Receive Payment Paths for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.
- **source** (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 | Decimal) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

`async call()`

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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.

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
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.

Raise
StreamClientError - Failed to fetch stream resource.

StrictSendPathsCallBuilder

class stellar_sdk.call_builder.call_builder_async.**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.ServerAsync.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 List Strict Send Payment Paths for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

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

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
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.

Raise
StreamClientError - Failed to fetch stream resource.

TradeAggregationsCallBuilder

class stellar_sdk.call_builder.call_builder_async.**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.ServerAsync.trade_aggregations().

Trade Aggregations facilitate efficient gathering of historical trade data.

See List Trade Aggregations for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – 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** (int | None) – lower time boundary represented as millis since epoch
- **end_time** (int | None) – upper time boundary represented as millis since epoch
- **offset** (int | None) – 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.*

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

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 Pagination

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

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`.

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
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`.

Raise
`StreamClientError` - Failed to fetch stream resource.

TradesCallBuilder

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

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

See List All Trades for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

Filter trades for a specific account

See Retrieve an Account's Trades for more information.

Parameters
account_id (str) – account id

Returns
current TradesCallBuilder instance

for_asset_pair(*base*, *counter*)

Filter trades for a specific asset pair (orderbook)

See List All Trades for more information.

Parameters

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

Returns
current TradesCallBuilder instance

`for_liquidity_pool`(*liquidity_pool_id*)

Filter trades for a specific liquidity pool.

See Liquidity Pools - Retrieve related Trades

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
current TradesCallBuilder instance

`for_offer`(*offer_id*)

Filter trades for a specific offer

See List All Trades for more information.

Parameters
offer_id (int | str) – offer id

Returns
current TradesCallBuilder instance

`for_trade_type`(*trade_type*)

Filter trades for a specific trade type

Horizon will reject requests which attempt to set *trade_type* to `liquidity_pools` when using the offer id filter.

Parameters
trade_type (str) – trade type, the currently supported types are `"orderbook"`, `"liquidity_pool"` and `"all"`, defaults to `"all"`.

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 Pagination

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

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`.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Trades* endpoint.

See Streaming for more information.

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

TransactionsCallBuilder

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

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

See List All Transactions for more information.

Parameters

- **horizon_url** (str) – Horizon server URL.
- **client** (*BaseAsyncClient*) – The client instance used to send request.

async call()

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

Return type
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 Pagination

Parameters
cursor (int | str) – A cursor is a value that points to a specific location in a collection of resources.

Returns
current CallBuilder instance

for_account(*account_id*)

This endpoint represents all transactions that affected a given account.

See Retrieve an Account's Transactions for more information.

Parameters
account_id (str) – account id

Returns
current TransactionsCallBuilder instance

`for_claimable_balance`(*claimable_balance_id*)

This endpoint represents all transactions referencing a given claimable balance and can be used in streaming mode.

See Claimable Balances - Retrieve related Transactions

Parameters
claimable_balance_id (str) – This claimable balance's id encoded in a hex string representation.

Returns
current TransactionsCallBuilder instance

`for_ledger`(*sequence*)

This endpoint represents all transactions in a given ledger.

See Retrieve a Ledger's Transactions for more information.

Parameters
sequence (int | str) – ledger sequence

Returns
current TransactionsCallBuilder instance

`for_liquidity_pool`(*liquidity_pool_id*)

This endpoint represents all transactions referencing a given liquidity pool.

See Liquidity Pools - Retrieve related Transactions

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Returns
this 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.

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 Pagination

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

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.

Returns
current CallBuilder instance

stream()

Creates an EventSource that listens for events from the *Transactions* endpoint.

See Streaming for more information.

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

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 Retrieve a Transaction for more information.

Parameters
transaction_hash (str) – transaction hash

Returns
current TransactionsCallBuilder instance

2.1.6 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.

abstractmethod async get(*url*, *params=None*, *max_content_size=None*)

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (dict[str, str] | None) – the request params
- **max_content_size** (int | None) – the maximum allowed response content size in bytes. If the response exceeds this limit, a ContentSizeLimitExceededError is raised. If None, no limit is applied.

Return type
Response

Returns
the response from server

Raise
ConnectionError

Raise
ContentSizeLimitExceededError

abstractmethod async post(*url*, *data=None*, *json_data=None*)

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (dict[str, str] | None) – the data send to server
- **json_data** (dict[str, Any] | None) – the json data send to server

Return type
Response

Returns
the response from server

Raise
ConnectionError

abstractmethod 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** (dict[str, str] | None) – 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.

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

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (dict[str, str] | None) – the request params
- **max_content_size** (int | None) – the maximum allowed response content size in bytes. If the response exceeds this limit, a ContentSizeLimitExceededError is raised. If None, no limit is applied.

Return type
Response

Returns
the response from server

Raise
ConnectionError

Raise
ContentSizeLimitExceededError

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

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (dict[str, str] | None) – the data send to server
- **json_data** (dict[str, Any] | None) – the json data send to server

Return type
Response

Returns
the response from server

Raise
ConnectionError

abstractmethod 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** (dict[str, str] | None) – 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=11*, *post_timeout=33.0*, *backoff_factor=0.5*, *user_agent=None*, *custom_headers=None*, ***kwargs*)

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

Parameters

- **pool_size** (int | None) – persistent connection to Horizon and connection pool
- **request_timeout** (float) – the timeout for all GET requests
- **post_timeout** (float) – the timeout for all POST requests
- **backoff_factor** (float | None) – a backoff factor to apply between attempts after the second try

- **user_agent** (str | None) – the server can use it to identify you
- **custom_headers** (dict[str, str] | None) – any additional HTTP headers to add in requests

async close()

Close underlying connector.

Release all acquired resources.

Return type
None

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

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (dict[str, str] | None) – the request params
- **max_content_size** (int | None) – the maximum allowed response content size in bytes. If the response exceeds this limit, a ContentSizeLimitExceededError is raised. If None, no limit is applied.

Return type
Response

Returns
the response from server

Raise
ConnectionError

Raise
ContentSizeLimitExceededError

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

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (dict[str, str] | None) – the data send to server
- **json_data** (dict[str, Any] | None) – the json data send to server

Return type
Response

Returns
the response from server

Raise
ConnectionError

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

Perform Stream request.

Parameters

- **url** (str) – the request url

- **params** (dict[str, str] | None) – the request params

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

Returns
the stream response from server

Raise
StreamClientError - Failed to fetch stream resource.

RequestsClient

class stellar_sdk.client.requests_client.**RequestsClient**(*pool_size=10*, *num_retries=3*, *request_timeout=11*, *post_timeout=33.0*, *backoff_factor=0.5*, *session=None*, *stream_session=None*, *custom_headers=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 GET requests (for each retry)
- **post_timeout** (float) – the timeout for all POST requests (for each retry)
- **backoff_factor** (float) – a backoff factor to apply between attempts after the second try
- **session** (Session | None) – the request session
- **stream_session** (Session | None) – the stream request session
- **custom_headers** (dict[str, str] | None) – any additional HTTP headers to add in requests

close()

Close underlying connector.

Release all acquired resources.

Return type
None

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

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (dict[str, str] | None) – the request params
- **max_content_size** (int | None) – the maximum allowed response content size in bytes. If the response exceeds this limit, a ContentSizeLimitExceededError is raised. If None, no limit is applied.

Return type
Response

Returns
the response from server

Raise
ConnectionError

Raise
ContentSizeLimitExceededError

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

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (dict[str, str] | None) – the data send to server
- **json_data** (dict[str, Any] | None) – the json 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** (dict[str, str] | None) – the request params

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

Returns
a Generator for server response

Raise
StreamClientError

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*, *max_content_size=None*)

Perform HTTP GET request.

Parameters

- **url** (str) – the request url
- **params** (dict[str, str] | None) – the request params
- **max_content_size** (int | None) – **Not supported in SimpleRequestsClient**.

Return type
Response

Returns
the response from server

Raise
ConnectionError

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

Perform HTTP POST request.

Parameters

- **url** (str) – the request url
- **data** (dict[str, str] | None) – the data send to server
- **json_data** (dict[str, Any] | None) – the json 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** (dict[str, str] | None) – the request params

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

2.1.7 Contract

ContractMeta

class stellar_sdk.sep.contract_meta.**ContractMeta**(*entries=()*)

The *ContractMeta* object, which represents SEP-46 contract metadata.

`entries` are normalized to a read-only tuple in entry order.

Parameters
entries (tuple[*SCMetaEntry*, ...]) – The raw SEP-46 metadata XDR entries.

property entries: tuple[*SCMetaEntry*, ...]

Returns the raw SEP-46 metadata XDR entries.

classmethod from_wasm(*wasm*)

Creates a *ContractMeta* object from contract Wasm bytes.

Parameters
wasm (bytes) – The contract Wasm bytes.

Return type
ContractMeta

Returns
A *ContractMeta* object.

Raises
InvalidWasmError – If the Wasm module or metadata section cannot be decoded.

classmethod from_wasm_file(*path*)

Creates a *ContractMeta* object from a contract Wasm file.

Parameters
path (str | PathLike[str]) – The path to the contract Wasm file.

Return type
ContractMeta

Returns
A *ContractMeta* object.

Raises
InvalidWasmError – If the Wasm module or metadata section cannot be decoded.

classmethod from_xdr_bytes(*data*)

Creates a *ContractMeta* object from SEP-46 XDR stream bytes.

Parameters
data (bytes) – The XDR stream bytes.

Return type
ContractMeta

Returns
A *ContractMeta* object.

Raises
InvalidWasmError – If the XDR stream cannot be decoded.

get(*key*, *default=None*)

Returns the first SC_META_V0 value for key.

Parameters
- **key** (str) – The metadata key.
- **default** (str | None) – The default value returned when the key is not present.

Return type
str | None

Returns
The metadata value or default.

Raises
InvalidWasmError – If a key or value is not UTF-8.

get_all(*key*)

Returns all SC_META_V0 values for key in entry order.

Parameters
key (str) – The metadata key.

Return type
tuple[str, ...]

Returns
All metadata values for the key.

Raises
InvalidWasmError – If a key or value is not UTF-8.

implements_sep(*sep*)

Returns whether the contract declares support for sep via SEP-47.

Parameters
sep (int) – The SEP number.

Return type
bool

Returns
True if the contract declares support for the SEP.

items()

Returns SC_META_V0 key/value pairs in entry order.

Return type
tuple[tuple[str, str], ...]

Returns
The decoded key/value pairs.

Raises
InvalidWasmError – If a key or value is not UTF-8.

supported_seps(*strict=False*)

Returns SEP-47 SEP identifiers from `sep` meta entries.

Values are returned in first-seen order with duplicates removed. Invalid identifiers are skipped by default and raise `ValueError` when `strict` is true.

Parameters
strict (bool) – Whether to reject invalid SEP identifiers.

Return type
tuple[int, ...]

Returns
SEP identifiers declared by the contract.

Raises

- **ValueError** – If `strict` is true and an identifier is invalid.
- **InvalidWasmError** – If a key or value is not UTF-8.

to_xdr_bytes()

Serializes the metadata entries as SEP-46 XDR stream bytes.

Return type
bytes

Returns
The XDR stream bytes.

ContractSpec

class stellar_sdk.sep.contract_spec.**ContractSpec**(*entries=()*)

The *ContractSpec* object, which represents a SEP-48 contract interface specification.

`entries` are normalized to a read-only tuple in entry order.

Parameters
entries (tuple[*SCSpecEntry*, ...]) – The raw SEP-48 specification XDR entries.

property entries: tuple[*SCSpecEntry*, ...]

Returns the raw SEP-48 specification XDR entries.

property enums: tuple[*SCSpecUDTEnumV0*, ...]

Returns all enum entries.

property error_enums: tuple[*SCSpecUDTErrorEnumV0*, ...]

Returns all error enum entries.

property events: tuple[*SCSpecEventV0*, ...]

Returns all event entries.

classmethod from_wasm(*wasm*)

Creates a *ContractSpec* object from contract Wasm bytes.

Parameters
wasm (bytes) – The contract Wasm bytes.

Return type
ContractSpec

Returns
A *ContractSpec* object.

Raises
InvalidWasmError – If the Wasm module or specification section cannot be decoded.

classmethod from_wasm_file(*path*)

Creates a *ContractSpec* object from a contract Wasm file.

Parameters
path (str | PathLike[str]) – The path to the contract Wasm file.

Return type
ContractSpec

Returns
A *ContractSpec* object.

Raises
InvalidWasmError – If the Wasm module or specification section cannot be decoded.

classmethod from_xdr_bytes(*data*)

Creates a *ContractSpec* object from SEP-48 XDR stream bytes.

Parameters
data (bytes) – The XDR stream bytes.

Return type
ContractSpec

Returns
A *ContractSpec* object.

Raises
InvalidWasmError – If the XDR stream cannot be decoded.

property functions: tuple[*SCSpecFunctionV0*, ...]

Returns all function entries.

get_event(*name*)

Returns an event by name.

Parameters
name (str) – The event name.

Return type
SCSpecEventV0 | None

Returns
The event entry if present.

Raises
InvalidWasmError – If an event name is not UTF-8.

get_function(*name*)

Returns a function by name.

Parameters
name (str) – The function name.

Return type
SCSpecFunctionV0 | None

Returns
The function entry if present.

Raises
InvalidWasmError – If a function name is not UTF-8.

`get_udt`(*name*)

Returns a user-defined type by name.

Parameters
name (str) – The type name.

Return type
SCSpecEntry | None

Returns
The user-defined type entry if present.

Raises
InvalidWasmError – If a type name is not UTF-8.

`property structs: tuple[SCSpecUDTStructV0, ...]`

Returns all struct entries.

`to_xdr_bytes`()

Serializes the specification entries as SEP-48 XDR stream bytes.

Return type
bytes

Returns
The XDR stream bytes.

`property unions: tuple[SCSpecUDTUnionV0, ...]`

Returns all union entries.

ContractInfo

`class stellar_sdk.sep.contract_info.ContractInfo`(*meta*, *spec*, *env_meta=()*)

The *ContractInfo* object, which aggregates Soroban contract metadata and interface information.

`meta`, `spec`, and `env_meta` are read-only. `env_meta` is normalized to a tuple in entry order.

Parameters

- **meta** (*ContractMeta*) – The contract metadata.
- **spec** (*ContractSpec*) – The contract interface specification.
- **env_meta** (tuple[*SCEnvMetaEntry*, ...]) – The contract environment metadata entries.

`property env_meta: tuple[SCEnvMetaEntry, ...]`

Returns the contract environment metadata entries.

`classmethod from_wasm`(*wasm*)

Creates a *ContractInfo* object from contract Wasm bytes.

Parameters
wasm (bytes) – The contract Wasm bytes.

Return type
ContractInfo

Returns
A *ContractInfo* object.

Raises
InvalidWasmError – If the Wasm module or any introspection section cannot be decoded.

classmethod from_wasm_file(*path*)

Creates a *ContractInfo* object from a contract Wasm file.

Parameters
path (str | PathLike[str]) – The path to the contract Wasm file.

Return type
ContractInfo

Returns
A *ContractInfo* object.

Raises
InvalidWasmError – If the Wasm module or any introspection section cannot be decoded.

property meta: *ContractMeta*

Returns the contract metadata.

property spec: *ContractSpec*

Returns the contract interface specification.

ContractClient

class stellar_sdk.contract.**ContractClient**(*contract_id*, *rpc_url*, *network_passphrase*, *request_client=None*)

A client to interact with Soroban smart contracts.

This client is a wrapper for `TransactionBuilder` and *SorobanServer*. If you need more fine-grained control, please consider using them directly.

I strongly recommend that you do not use this client directly, but instead use stellar-contract-bindings to generate contract binding code, which will make calling the contract much simpler.

Parameters

- **contract_id** (str) – The ID of the Soroban contract.
- **rpc_url** (str) – The URL of the RPC server.
- **network_passphrase** (str) – The network passphrase.
- **request_client** (*BaseSyncClient* | None) – The request client used to send the request.

static create_contract(*wasm_id*, *source*, *signer*, *soroban_server*, *constructor_args=None*, *salt=None*, *network_passphrase=None*, *base_fee=100*, *transaction_timeout=300*, *submit_timeout=120*, *restore=True*)

Create a contract.

Parameters

- **wasm_id** (bytes | str) – The wasm ID.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServer*) – The Soroban server.

- **constructor_args** (Sequence[*SCVal*] | None) – The constructor arguments.
- **salt** (bytes | None) – The salt.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.
- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.
- **restore** (bool) – Whether to restore the transaction.

Return type
str

Returns
The contract ID.

static create_stellar_asset_contract_from_asset(*asset*, *source*, *signer*, *soroban_server*, *network_passphrase=None*, *base_fee=100*, *submit_timeout=120*)

Create a Stellar asset contract from an asset.

Parameters

- **asset** (*Asset*) – The asset.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServer*) – The Soroban server.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.
- **base_fee** (int) – The base fee for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.

Return type
str

Returns
The contract ID.

invoke(*function_name*, *parameters=None*, *source='GAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAWHF'*, *signer=None*, *parse_result_xdr_fn=None*, *base_fee=100*, *transaction_timeout=300*, *submit_timeout=30*, *simulate=True*, *restore=True*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

Build an *AssembledTransaction* to invoke a contract function.

Parameters

- **function_name** (str) – The name of the function to invoke.
- **parameters** (Sequence[*SCVal*] | None) – The parameters to pass to the function.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair* | None) – The signer for the transaction.

- **parse_result_xdr_fn** (Callable[[*SCVal*], TypeVar(T)] | None) – The function to parse the result XDR returned by the contract function, keep the result as SCVal if not provided.
- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.
- **simulate** (bool) – Whether to simulate the transaction.
- **restore** (bool) – Whether to restore the transaction, only valid when simulate is True, and the signer is provided.
- **addl_resources** (ResourceLeeway | None) – Additional resource leeway forwarded to every simulation performed by the returned *AssembledTransaction*.
- **auth_mode** (*AuthMode* | None) – Authorization mode forwarded to every simulation performed by the returned *AssembledTransaction*.
- **auth_v2** (bool) – Whether simulations performed by the returned *AssembledTransaction* should request ADDRESS_V2 credentials in returned auth entries. Set to True when targeting Protocol 27 RPC behavior and V2 auth entries are required.

Return type
AssembledTransaction[TypeVar(T)]

Returns

static upload_contract_wasm(*contract*, *source*, *signer*, *soroban_server*, *network_passphrase=None*, *base_fee=100*, *transaction_timeout=300*, *submit_timeout=120*)

Upload a contract wasm.

Parameters

- **contract** (bytes | str) – The contract wasm.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServer*) – The Soroban server.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.
- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.

Return type
bytes

Returns
The wasm ID.

AssembledTransaction

class stellar_sdk.contract.**AssembledTransaction**(*transaction_builder*, *server*, *transaction_signer=None*, *parse_result_xdr_fn=None*, *submit_timeout=180*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

A class representing an assembled Soroban transaction that can be simulated and sent.

The lifecycle of a transaction typically follows these steps:

1. Construct the transaction (usually via a Client)
2. Simulate the transaction
3. Sign the transaction
4. Submit the transaction

Parameters

- **transaction_builder** (*TransactionBuilder*) – The transaction builder including the operation to invoke
- **server** (*SorobanServer*) – The Soroban server instance to use
- **transaction_signer** (*Keypair* | None) – Optional keypair for signing transactions, if you don't need to submit the transaction, you can set this to *None*.
- **parse_result_xdr_fn** (Callable[[*SCVal*], TypeVar(T)] | None) – Optional function to parse XDR results, keep *None* for raw XDR
- **submit_timeout** (int) – Timeout in seconds for transaction submission (default: 180s)
- **addl_resources** (ResourceLeeway | None) – Additional resource leeway forwarded to every internal simulation call (initial simulate, prepare, and the post-restore re-simulation). Not applied to the inner RestoreFootprint transaction created by automatic state restoration.
- **auth_mode** (*AuthMode* | None) – Authorization mode forwarded to every internal simulation call. Use *AuthMode.RECORD_ALL_NOROOT* to opt into non-root authorization in recording mode.
- **auth_v2** (bool) – Whether internal simulation calls should request ADDRESS_V2 credentials in returned auth entries. Set to True when targeting Protocol 27 RPC behavior and V2 auth entries are required.

authorize(*address=None*, *signer=None*, *valid_until_ledger_sequence=None*, ** (Keyword-only parameters separator (PEP 3102))*, *valid_for_ledger_count=None*)

Authorize matching Soroban authorization entries.

Parameters

- **address** (*Address* | str | *Keypair* | Callable[[*HashIDPreimage*], *SCVal*] | None) – Classic account (G...) or contract (C...) address whose authorization entries should be signed. Required when signer is not a Keypair; otherwise inferred from the keypair's public key. For convenience, a Keypair or custom signer may be passed as the first positional argument when the address can be inferred.
- **signer** (*Keypair* | Callable[[*HashIDPreimage*], *SCVal*] | None) – A Keypair, or any custom *AuthorizationSigner* for non-default account contracts (BLS, WebAuthn, ...).

- **valid_until_ledger_sequence** (int | None) – Optional ledger sequence until which the authorization is valid.
- **valid_for_ledger_count** (int | None) – Optional number of ledgers from the latest simulation ledger for which the authorization remains valid. Defaults to 100 when valid_until_ledger_sequence is not provided.

Return type
AssembledTransaction

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

is_read_call()

Check if the transaction is a read call.

Return type
bool

Returns
True if the transaction is a read call, False otherwise

Raises
NotYetSimulatedError: If the transaction has not been simulated

needs_non_invoker_signing_by(*include_already_signed=False*)

Get the addresses that need to sign the authorization entries.

Parameters
include_already_signed (bool) – Whether to include addresses that have already signed the authorization entries.

Return type
set[str]

Returns
The addresses that need to sign the authorization entries.

Raises
NotYetSimulatedError: If the transaction has not been simulated

prepare(*restore=True*)

Prepare the current built transaction for signing and submission.

Unlike *simulate()*, this method simulates built_transaction as it currently exists, including any authorization entries that were signed after the initial simulation.

Parameters
restore (bool) – Whether to automatically restore contract state if needed, defaults to True

Return type
AssembledTransaction

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

Raises
NeedsMoreSignaturesError: If authorization entries still require signatures

Raises
SimulationFailedError: If the simulation fails

`result()`

Get the result of the function invocation from the simulation.

Return type
TypeVar(T) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

`sign`(*transaction_signer=None*, *force=False*)

Signs the transaction.

Parameters

- **transaction_signer** (*Keypair* | None) – Optional keypair to sign with (overrides instance signer)
- **force** (bool) – Whether to sign even if the transaction is a read call

Return type
AssembledTransaction

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

Raises
NoSignatureNeededError: If the transaction is a read call

Raises
NeedsMoreSignaturesError: If the transaction requires more signatures for authorization entries.

`sign_and_submit`(*transaction_signer=None*, *force=False*)

Signs and submits the transaction in one step.

A convenience method combining sign() and submit().

Parameters

- **transaction_signer** (*Keypair* | None) – Optional keypair to sign with (overrides instance signer)
- **force** (bool) – Whether to sign and submit even if the transaction is a read call

Return type
TypeVar(T) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

sign_auth_entries(*auth_entries_signer*, *address=None*, *valid_until_ledger_sequence=None*, ***, *valid_for_ledger_count=None*)

Signs the transaction's authorization entries.

This method is kept for backwards compatibility. New code can use *authorize()*, which also supports relative expiration via `valid_for_ledger_count`.

Parameters

- **auth_entries_signer** (*Keypair* | `Callable`[[*HashIDPreimage*], *SCVal*]) – A `Keypair`, or any custom *AuthorizationSigner* for non-default account contracts (BLS, WebAuthn, ...).
- **address** (*Address* | `str` | `None`) – Classic account (`G...`) or contract (`C...`) address whose authorization entries should be signed. Required when `auth_entries_signer` is not a `Keypair`; otherwise inferred from the keypair's public key.
- **valid_until_ledger_sequence** (`int` | `None`) – Optional ledger sequence until which the authorization is valid, if not set, defaults to 100 ledgers from the current ledger.
- **valid_for_ledger_count** (`int` | `None`) – Optional number of ledgers from the latest simulation ledger for which the authorization remains valid.

Return type
AssembledTransaction

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

simulate(*restore=True*)

Simulates the transaction on the network.

Must be called before signing or submitting the transaction. Will automatically restore required contract state if restore to True.

Parameters
restore (`bool`) – Whether to automatically restore contract state if needed, defaults to True

Return type
AssembledTransaction

Returns
Self for chaining

Raises
SimulationFailedError: If the simulation fails

Raises
ExpiredStateError: If state restoration failed

submit()

Submits the transaction to the network.

It will send the transaction to the network and wait for the result.

Return type
`TypeVar`(`T`) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

Raises
SendTransactionFailedError: If sending the transaction fails

Raises
TransactionStillPendingError: If the transaction is still pending after the timeout, you can re-call this method to wait longer

Raises
TransactionFailedError: If the transaction fails

`to_xdr()`

Get the XDR representation of the transaction envelope.

Returns
The XDR representation of the transaction envelope

ContractClientAsync

`class stellar_sdk.contract.ContractClientAsync(contract_id, rpc_url, network_passphrase, request_client=None)`

A client to interact with Soroban smart contracts.

This client is a wrapper for `TransactionBuilder` and *SorobanServerAsync*. If you need more fine-grained control, please consider using them directly.

I strongly recommend that you do not use this client directly, but instead use stellar-contract-bindings to generate contract binding code, which will make calling the contract much simpler.

Parameters

- **contract_id** (str) – The ID of the Soroban contract.
- **rpc_url** (str) – The URL of the RPC server.
- **network_passphrase** (str) – The network passphrase.
- **request_client** (*BaseAsyncClient* | None) – The request client used to send the request.

`async static create_contract(wasm_id, source, signer, soroban_server, constructor_args=None, salt=None, network_passphrase=None, base_fee=100, transaction_timeout=300, submit_timeout=120, restore=True)`

Create a contract.

Parameters

- **wasm_id** (bytes | str) – The wasm ID.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServerAsync*) – The Soroban server.
- **constructor_args** (Sequence[*SCVal*] | None) – The constructor arguments.
- **salt** (bytes | None) – The salt.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.

- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.
- **restore** (bool) – Whether to restore the transaction.

Return type
str

Returns
The contract ID.

async static create_stellar_asset_contract_from_asset(*asset*, *source*, *signer*, *soroban_server*, *network_passphrase=None*, *base_fee=100*, *submit_timeout=120*)

Create a Stellar asset contract from an asset.

Parameters

- **asset** (*Asset*) – The asset.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServerAsync*) – The Soroban server.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.
- **base_fee** (int) – The base fee for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.

Return type
str

Returns
The contract ID.

async invoke(*function_name*, *parameters=None*, *source='GAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAWHF'*, *signer=None*, *parse_result_xdr_fn=None*, *base_fee=100*, *transaction_timeout=300*, *submit_timeout=30*, *simulate=True*, *restore=True*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

Build an *AssembledTransactionAsync* to invoke a contract function.

Parameters

- **function_name** (str) – The name of the function to invoke.
- **parameters** (Sequence[*SCVal*] | None) – The parameters to pass to the function.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair* | None) – The signer for the transaction.
- **parse_result_xdr_fn** (Callable[[*SCVal*], TypeVar(T)] | None) – The function to parse the result XDR returned by the contract function, keep the result as SCVal if not provided.
- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.

- **submit_timeout** (int) – The timeout for submitting the transaction.
- **simulate** (bool) – Whether to simulate the transaction.
- **restore** (bool) – Whether to restore the transaction, only valid when simulate is True, and the signer is provided.
- **addl_resources** (ResourceLeeway | None) – Additional resource leeway forwarded to every simulation performed by the returned *AssembledTransactionAsync*.
- **auth_mode** (*AuthMode* | None) – Authorization mode forwarded to every simulation performed by the returned *AssembledTransactionAsync*.
- **auth_v2** (bool) – Whether simulations performed by the returned *AssembledTransactionAsync* should request ADDRESS_V2 credentials in returned auth entries. Set to True when targeting Protocol 27 RPC behavior and V2 auth entries are required.

Return type
AssembledTransactionAsync[TypeVar(T)]

Returns

async static **upload_contract_wasm**(*contract*, *source*, *signer*, *soroban_server*, *network_passphrase=None*, *base_fee=100*, *transaction_timeout=300*, *submit_timeout=120*)

Upload a contract wasm.

Parameters

- **contract** (bytes | str) – The contract wasm.
- **source** (str | *MuxedAccount*) – The source account for the transaction.
- **signer** (*Keypair*) – The signer for the transaction.
- **soroban_server** (*SorobanServerAsync*) – The Soroban server.
- **network_passphrase** (str | None) – The network passphrase, default to the network of the Soroban server.
- **base_fee** (int) – The base fee for the transaction.
- **transaction_timeout** (int) – The timeout for the transaction.
- **submit_timeout** (int) – The timeout for submitting the transaction.

Return type
bytes

Returns
The wasm ID.

AssembledTransactionAsync

class stellar_sdk.contract.**AssembledTransactionAsync**(*transaction_builder*, *server*, *transaction_signer=None*, *parse_result_xdr_fn=None*, *submit_timeout=180*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

A class representing an assembled Soroban transaction that can be simulated and sent.

The lifecycle of a transaction typically follows these steps:

1. Construct the transaction (usually via a Client)
2. Simulate the transaction
3. Sign the transaction
4. Submit the transaction

Parameters

- **transaction_builder** (*TransactionBuilder*) – The transaction builder including the operation to invoke
- **server** (*SorobanServerAsync*) – The Soroban server instance to use
- **transaction_signer** (*Keypair* | None) – Optional keypair for signing transactions, if you don't need to submit the transaction, you can set this to *None*.
- **parse_result_xdr_fn** (Callable[[*SCVal*], TypeVar(T)] | None) – Optional function to parse XDR results, keep *None* for raw XDR
- **submit_timeout** (int) – Timeout in seconds for transaction submission (default: 180s)
- **addl_resources** (ResourceLeeway | None) – Additional resource leeway forwarded to every internal simulation call (initial simulate, prepare, and the post-restore re-simulation). Not applied to the inner RestoreFootprint transaction created by automatic state restoration.
- **auth_mode** (*AuthMode* | None) – Authorization mode forwarded to every internal simulation call. Use *AuthMode.RECORD_ALL_NOROOT* to opt into non-root authorization in recording mode.
- **auth_v2** (bool) – Whether internal simulation calls should request ADDRESS_V2 credentials in returned auth entries. Set to True when targeting Protocol 27 RPC behavior and V2 auth entries are required.

async authorize(*address=None*, *signer=None*, *valid_until_ledger_sequence=None*, ***, *valid_for_ledger_count=None*)

Authorize matching Soroban authorization entries.

Parameters

- **address** (*Address* | str | *Keypair* | Callable[[*HashIDPreimage*], *SCVal*] | None) – Classic account (G...) or contract (C...) address whose authorization entries should be signed. Required when signer is not a Keypair; otherwise inferred from the keypair's public key. For convenience, a Keypair or custom signer may be passed as the first positional argument when the address can be inferred.
- **signer** (*Keypair* | Callable[[*HashIDPreimage*], *SCVal*] | None) – A Keypair, or any custom *AuthorizationSigner* for non-default account contracts (BLS, WebAuthn, ...).
- **valid_until_ledger_sequence** (int | None) – Optional ledger sequence until which the authorization is valid.
- **valid_for_ledger_count** (int | None) – Optional number of ledgers from the latest simulation ledger for which the authorization remains valid. Defaults to 100 when valid_until_ledger_sequence is not provided.

Return type

AssembledTransactionAsync

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

`is_read_call()`

Check if the transaction is a read call.

Return type
bool

Returns
True if the transaction is a read call, False otherwise

Raises
NotYetSimulatedError: If the transaction has not been simulated

`needs_non_invoker_signing_by(include_already_signed=False)`

Get the addresses that need to sign the authorization entries.

Parameters
include_already_signed (bool) – Whether to include addresses that have already signed the authorization entries.

Return type
set[str]

Returns
The addresses that need to sign the authorization entries.

Raises
NotYetSimulatedError: If the transaction has not been simulated

`async prepare(restore=True)`

Prepare the current built transaction for signing and submission.

Unlike *simulate()*, this method simulates `built_transaction` as it currently exists, including any authorization entries that were signed after the initial simulation.

Parameters
restore (bool) – Whether to automatically restore contract state if needed, defaults to True

Return type
AssembledTransactionAsync

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

Raises
NeedsMoreSignaturesError: If authorization entries still require signatures

Raises
SimulationFailedError: If the simulation fails

`result()`

Get the result of the function invocation from the simulation.

Return type
TypeVar(T) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

sign(*transaction_signer=None*, *force=False*)

Signs the transaction.

Parameters

- **transaction_signer** (*Keypair* | None) – Optional keypair to sign with (overrides instance signer)
- **force** (bool) – Whether to sign even if the transaction is a read call

Return type
AssembledTransactionAsync

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

Raises
NoSignatureNeededError: If the transaction is a read call

Raises
NeedsMoreSignaturesError: If the transaction requires more signatures for authorization entries.

async sign_and_submit(*transaction_signer=None*, *force=False*)

Signs and submits the transaction in one step.

A convenience method combining sign() and submit().

Parameters

- **transaction_signer** (*Keypair* | None) – transaction_signer: Optional keypair to sign with (overrides instance signer)
- **force** (bool) – Whether to sign and submit even if the transaction is a read call

Return type
TypeVar(T) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

async sign_auth_entries(*auth_entries_signer*, *address=None*, *valid_until_ledger_sequence=None*, ***, *valid_for_ledger_count=None*)

Signs the transaction's authorization entries.

This method is kept for backwards compatibility. New code can use *authorize()*, which also supports relative expiration via `valid_for_ledger_count`.

Parameters

- **auth_entries_signer** (*Keypair* | Callable[[*HashIDPreimage*], *SCVal*]) – A `Keypair`, or any custom *AuthorizationSigner* for non-default account contracts (BLS, WebAuthn, ...).

- **address** (*Address* | str | None) – Classic account (G...) or contract (C...) address whose authorization entries should be signed. Required when auth_entries_signer is not a Keypair; otherwise inferred from the keypair's public key.
- **valid_until_ledger_sequence** (int | None) – Optional ledger sequence until which the authorization is valid, if not set, defaults to 100 ledgers from the current ledger.
- **valid_for_ledger_count** (int | None) – Optional number of ledgers from the latest simulation ledger for which the authorization remains valid.

Return type
AssembledTransactionAsync

Returns
Self for chaining

Raises
NotYetSimulatedError: If the transaction has not been simulated

async simulate(*restore=True*)

Simulates the transaction on the network.

Must be called before signing or submitting the transaction. Will automatically restore required contract state if restore to True.

Parameters
restore (bool) – Whether to automatically restore contract state if needed, defaults to True

Return type
AssembledTransactionAsync

Returns
Self for chaining

Raises
SimulationFailedError: If the simulation fails

Raises
ExpiredStateError: If state restoration failed

async submit()

Submits the transaction to the network.

It will send the transaction to the network and wait for the result.

Return type
TypeVar(T) | *SCVal*

Returns
The value returned by the invoked function, parsed if parse_result_xdr_fn was set, otherwise raw XDR

Raises
SendTransactionFailedError: If sending the transaction fails

Raises
TransactionStillPendingError: If the transaction is still pending after the timeout, you can re-call this method to wait longer

Raises
TransactionFailedError: If the transaction fails

`to_xdr()`

Get the XDR representation of the transaction envelope.

Returns
The XDR representation of the transaction envelope

Exceptions

exception `stellar_sdk.contract.exceptions.`**AssembledTransactionError**(*message*, *assembled_transaction*)

Raised when an assembled transaction fails.

exception `stellar_sdk.contract.exceptions.`**ExpiredStateError**(*message*, *assembled_transaction*)

Raised when the state has expired.

exception `stellar_sdk.contract.exceptions.`**NeedsMoreSignaturesError**(*message*, *assembled_transaction*)

Raised when more signatures are needed.

exception `stellar_sdk.contract.exceptions.`**NeedsPreparationError**(*message*, *assembled_transaction*)

Raised when the transaction needs to be prepared before continuing.

exception `stellar_sdk.contract.exceptions.`**NoSignatureNeededError**(*message*, *assembled_transaction*)

Raised when no signature is needed.

exception `stellar_sdk.contract.exceptions.`**NotYetSimulatedError**(*message*, *assembled_transaction*)

Raised when trying to get the result of a transaction that has not been simulated yet.

exception `stellar_sdk.contract.exceptions.`**RestorationFailureError**(*message*, *assembled_transaction*)

Raised when a restoration fails.

exception `stellar_sdk.contract.exceptions.`**SendTransactionFailedError**(*message*, *assembled_transaction*)

Raised when invoking *send_transaction* fails.

exception `stellar_sdk.contract.exceptions.`**SimulationFailedError**(*message*, *assembled_transaction*)

Raised when a simulation fails.

exception `stellar_sdk.contract.exceptions.`**TransactionFailedError**(*message*, *assembled_transaction*)

Raised when invoking *get_transaction* fails.

exception `stellar_sdk.contract.exceptions.`**TransactionStillPendingError**(*message*, *assembled_transaction*)

Raised when the transaction is still pending.

2.1.8 Exceptions

SdkError

class stellar_sdk.exceptions.**SdkError**

Base exception for all stellar sdk 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

FeatureNotEnabledError

class stellar_sdk.exceptions.**FeatureNotEnabledError**

The feature is not enabled.

2.1.9 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()*
- *Keypair.from_mnemonic_phrase()*
- *Keypair.from_shamir_mnemonic_phrases()*

Learn how to create a key through our documentation: Generate Keypair.

Parameters

- **verify_key** (VerifyKey) – The verifying (public) Ed25519 key in the keypair.
- **signing_key** (SigningKey | None) – 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*, *language=Language.ENGLISH*, *passphrase=''*, *index=0*)

Generate a *Keypair* object via a mnemonic phrase.

Parameters

- **mnemonic_phrase** (str) – A unique string used to deterministically generate keypairs.
- **language** (*Language* | str) – The language of the mnemonic phrase, defaults to english.
- **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)
```

Return type
Keypair

Returns
A new *Keypair* object derived from the mnemonic.

classmethod from_public_key(*public_key*)

Generate a *Keypair* object from a public key.

Parameters
public_key (str) – public key (ex. "GATPGGOIE6VWADVKD3ER3IFO2IH6DTOA5G535ITB3TT66FZFSIZEAU2B")

Return type
Keypair

Returns
A new *Keypair* object 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* object 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* object derived by the ed25519 secret key seed raw bytes

classmethod from_secret(*secret*)

Generate a *Keypair* object from a secret key.

Parameters
secret (str) – secret key (ex. "SB2LHKBL24ITV2Y346BU46XPEL45BDAFOOJLZ6SESCJZ6V5JMP7D6G5X")

Return type
Keypair

Returns
A new *Keypair* object derived by the secret.

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

classmethod from_shamir_mnemonic_phrases(*mnemonic_phrases*, *passphrase=''*, *index=0*)

Generate a *Keypair* object via a list of mnemonic phrases.

Parameters

- **mnemonic_phrases** (Iterable[str]) – A list of unique strings used to deterministically generate a keypair.
- **passphrase** (str) – An optional passphrase used to decrypt the secret key.
- **index** (int) – The index of the keypair generated by the mnemonic. This allows for multiple Keypairs to be derived from the same mnemonic.

Return type
Keypair

Returns
A new *Keypair* object derived from the mnemonic phrases.

static generate_mnemonic_phrase(*language=Language.ENGLISH*, *strength=128*)

Generate a mnemonic phrase.

Parameters

- **language** (*Language* | str) – The language of the mnemonic phrase, defaults to english.
- **strength** (int) – The complexity of the mnemonic, its possible value is 128, 160, 192, 224 and 256.

Return type
str

Returns
A mnemonic phrase.

static generate_shamir_mnemonic_phrases(*member_threshold*, *member_count*, *passphrase=''*, *strength=256*)

Generate mnemonic phrases using Shamir secret sharing method.

A randomly generated secret key is generated and split into *member_count* mnemonic phrases. The secret key can be later reconstructed using any subset of *member_threshold* phrases.

Parameters

- **member_threshold** (int) – Number of members required to reconstruct the secret key.
- **member_count** (int) – Number of shares the secret is split into.
- **passphrase** (str) – An optional passphrase used to decrypt the secret key.
- **strength** (int) – The complexity of the mnemonics in terms of bites, its possible value is 128, 160, 192, 224 and 256. Strengths of 128 and 256 lead respectively to shares with 20 and 33 words.

Return type
list[str]

Returns
A list of mnemonic phrases.

property public_key: str

Returns public key associated with this *Keypair* object

Returns
public key

classmethod random()

Generate a *Keypair* object from a randomly generated seed.

Return type
Keypair

Returns
A new *Keypair* object 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

property secret: str

Returns secret key associated with this *Keypair* object

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 (bytes) – signed bytes

Return type
DecoratedSignature

Returns
sign decorated

sign_message(*message*)

Sign a message according to SEP-53.

Parameters
message (str | bytes) – The message to sign, as a string or bytes.

Return type
bytes

Returns
The signature bytes.

sign_payload_decorated(*data*)

Returns the decorated signature hint for a signed payload signer.

The signature hint of an ed25519 signed payload signer is the last 4 bytes of the ed25519 public key XORed with last 4 bytes of the payload. If the payload has a length less than 4 bytes, then 1 to 4 zero bytes are appended to the payload such that it has a length of 4 bytes, for calculating the hint.

Parameters
data (bytes) – data to both sign and treat as the payload

Return type
DecoratedSignature

Returns
sign decorated

signature_hint()

Returns signature hint associated with this *Keypair* object

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

verify_message(*message*, *signature*)

Verify a SEP-53 signed message.

Parameters

- **message** (str | bytes) – The original message, as a string or bytes.
- **signature** (bytes) – The signature to verify.

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

Return type
None

xdr_public_key()

Return type
PublicKey

Returns
xdr public key

2.1.10 LiquidityPoolAsset

stellar_sdk.liquidity_pool_asset.**LIQUIDITY_POOL_FEE_V18 = 30**

LIQUIDITY_POOL_FEE_V18 is the default liquidity pool fee in protocol v18. It defaults to 30 base points (0.3%).

class stellar_sdk.liquidity_pool_asset.**LiquidityPoolAsset**(*asset_a*, *asset_b*, *fee=30*)

The *LiquidityPoolAsset* object, which represents a liquidity pool trustline change.

Parameters

- **asset_a** (*Asset*) – The first asset in the Pool, it must respect the rule asset_a < asset_b. See *stellar_sdk.liquidity_pool_asset.LiquidityPoolAsset.is_valid_lexicographic_order()* for more details on how assets are sorted.
- **asset_b** (*Asset*) – The second asset in the Pool, it must respect the rule asset_a < asset_b. See *stellar_sdk.liquidity_pool_asset.LiquidityPoolAsset.is_valid_lexicographic_order()* for more details on how assets are sorted.
- **fee** (int) – The liquidity pool fee. For now the only fee supported is *30*.

Raise

ValueError

classmethod from_xdr_object(*xdr_object*)

Create a *LiquidityPoolAsset* from an XDR ChangeTrustAsset object.

Parameters

xdr_object (*ChangeTrustAsset*) – The XDR ChangeTrustAsset object.

Return type

LiquidityPoolAsset

Returns

A new *LiquidityPoolAsset* object from the given XDR ChangeTrustAsset object.

static is_valid_lexicographic_order(*asset_a*, *asset_b*)

Compares if asset_a < asset_b according with the criteria:

1. First compare the type (eg. native before alphanum4 before alphanum12).
2. If the types are equal, compare the assets codes.
3. If the asset codes are equal, compare the issuers.

Parameters

- **asset_a** (*Asset*) – The first asset in the lexicographic order.
- **asset_b** (*Asset*) – The second asset in the lexicographic order.

Return type

bool

Returns

return *True* if asset_a < asset_b

property liquidity_pool_id: str

Computes the liquidity pool id for current instance.

Returns

Liquidity pool id.

to_change_trust_asset_xdr_object()

Returns the xdr object for this ChangeTrustAsset object.

Return type

ChangeTrustAsset

Returns
XDR ChangeTrustAsset object

2.1.11 LiquidityPoolId

class stellar_sdk.liquidity_pool_id.**LiquidityPoolId**(*liquidity_pool_id*)

The *LiquidityPoolId* object, which represents the asset referenced by a trustline to a liquidity pool.

Parameters
liquidity_pool_id (str) – The ID of the liquidity pool in hex string.

Raise
ValueError

classmethod from_xdr_object(*xdr_object*)

Create a *LiquidityPoolId* from an XDR Asset object.

Parameters
xdr_object (*TrustLineAsset*) – The XDR TrustLineAsset object.

Return type
LiquidityPoolId

Returns
A new *LiquidityPoolId* object from the given XDR TrustLineAsset object.

to_trust_line_asset_xdr_object()

Returns the xdr object for this LiquidityPoolId object.

Return type
TrustLineAsset

Returns
XDR TrustLineAsset object

2.1.12 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 <https://developers.stellar.org/docs/learn/fundamentals/stellar-data-structures/operations-and-transactions#memo`__** for more information on how memos are used within transactions, as well as information on the available types of memos.

classmethod from_xdr_object(*xdr_object*)

Returns an Memo object from XDR memo object.

Return type
Memo

abstractmethod 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_object*)

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. Note, *text* can be anything, see this issue for more information.

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

classmethod from_xdr_object(*xdr_object*)

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_object*)

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 (`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_object*)

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` | `str`) – 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_object*)

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

2.1.13 MuxedAccount

class stellar_sdk.muxed_account.**MuxedAccount**(*account_id*, *account_muxed_id=None*)

The *MuxedAccount* object, which represents a multiplexed account on Stellar's network.

An example:

```
from stellar_sdk import MuxedAccount

account_id = "GAQAA5L65LSYH7CQ3VTJ7F3HHLGCL3DSLAR2Y47263D56MNNGHSQSTVY"
account_muxed_id = 1234
account_muxed =
↪"MAQAA5L65LSYH7CQ3VTJ7F3HHLGCL3DSLAR2Y47263D56MNNGHSQSAAAAAAAAAAE2LP26"

# generate account_muxed
muxed = MuxedAccount(account=account_id, account_muxed_id=1234)  # account_muxed_id␣
↪is optional.
print(f"account_muxed: {muxed.account_muxed}")  # `account_muxed` returns ``None`` if␣
↪`account_muxed_id` is ``None``.

# parse account_muxed
muxed = MuxedAccount.from_account(account_muxed)
print(f"account_id: {muxed.account_id}\n"
      f"account_muxed_id: {muxed.account_muxed_id}")
```

See SEP-0023 for more information.

Parameters

- **account_id** (str) – ed25519 account id, for example: "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD". It should be a string starting with G. If you want to build a MuxedAccount object using an address starting with M, please use stellar_sdk.MuxedAccount.from_account().
- **account_muxed_id** (int | None) – account multiplexing id, can be None or a non-negative integer. (ex. 1234)

property account_muxed: str | None

Get the multiplex address starting with M, return None if *account_id_id* is None.

classmethod from_account(*account*)

Create a *MuxedAccount* object from account id or muxed account id.

Parameters
account (str) – account id or muxed account id (ex. "GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD" or "MAAAAAAAAAAAAJURAAB2X52XFQP6FBXLGT6LWOOWMEXWHEWBDVRZ7V5WH34Y22MPFBHUHY")

Return type
MuxedAccount

classmethod from_xdr_object(*xdr_object*)

Create a *MuxedAccount* object from an XDR Asset object.

Parameters
xdr_object (*MuxedAccount*) – The MuxedAccount object.

Return type
MuxedAccount

Returns
A new *MuxedAccount* object from the given XDR MuxedAccount object.

to_xdr_object()

Returns the xdr object for this MuxedAccount object.

Return type
MuxedAccount

Returns
XDR MuxedAccount object

property universal_account_id: str

Get the universal account id, if *account_muxed_id* is `None`, it will return ed25519 public key (ex. `"GDGQVOKHW4VEJRU2TETD6DBRKEO5ERCNF353LW5WBFW3JJWQ2BRQ6KDD"`), otherwise it will return muxed account (ex. `"MAAAAAAAAAAAJURAAB2X52XFQP6FBXLGT6LWOOWMEXWHEWBDVRZ7V5WH34Y22MPFBHUHY"`)

2.1.14 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"`)

FUTURENET_NETWORK_PASSPHRASE: str = 'Test SDF Future Network ; October 2022'

The Future network passphrase.

PUBLIC_NETWORK_PASSPHRASE: str = 'Public Global Stellar Network ; September 2015'

The Public network passphrase.

SANDBOX_NETWORK_PASSPHRASE = 'Local Sandbox Stellar Network ; September 2022'

The Sandbox network passphrase.

STANDALONE_NETWORK_PASSPHRASE: str = 'Standalone Network ; February 2017'

The Standalone network passphrase.

TESTNET_NETWORK_PASSPHRASE: str = 'Test SDF Network ; September 2015'

The Test network passphrase.

network_id()

Get the network ID of the network, which is an 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

2.1.15 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 (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

static from_xdr_amount(*value*)

Converts a 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(*xdr_object*)

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

Parameters
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
MuxedAccount | None

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 (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

See Account Merge for more information.

Parameters

- **destination** (*MuxedAccount* | str) – Destination to merge the source account into.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Allow Trust for more information.

Parameters

- **trustor** (str) – The trusting account (the one being authorized).
- **asset_code** (str) – The asset code being authorized.
- **authorize** (*TrustLineEntryFlag* | bool) – *True* to authorize the line, *False* to deauthorizeif you need further control, you can also use *stellar_sdk.operation.allow_trust.TrustLineEntryFlag*.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

class stellar_sdk.operation.allow_trust.**TrustLineEntryFlag**(**values*)

Indicates which flags to set. For details about the flags, please refer to the CAP-0018.

- **UNAUTHORIZED_FLAG**: The account can hold a balance but cannot receive payments, send payments, maintain offers or manage offers
- **AUTHORIZED_FLAG**: The account can hold a balance, receive payments, send payments, maintain offers or manage offers
- **AUTHORIZED_TO_MAINTAIN_LIABILITIES_FLAG**: The account can hold a balance and maintain offers but cannot receive payments, send payments or manage offers

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

See Bump Sequence for more information.

Parameters

- **bump_to** (int) – Sequence number to bump to.
- **source** (*MuxedAccount* | str | None) – The optional source account.

classmethod from_xdr_object(*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.

Threshold: Medium

See Change Trust for more information.

Parameters

- **asset** (*Asset* | *LiquidityPoolAsset*) – The asset for the trust line.
- **limit** (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** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Create Account for more information.

Parameters

- **destination** (str) – Destination account ID to create an account for.
- **starting_balance** (str | Decimal) – Amount in XLM the account should be funded for. Must be greater than the reserve balance amount.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod **from_xdr_object**(*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 *ManageBuyOffer* for more details.

Threshold: Medium

See Create Passive Sell Offer for more information.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – The total amount you're selling.

- **price** (*Price* | str | Decimal) – Price of 1 unit of *selling* in terms of *buying*.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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 an Inflation operation on Stellar's network.

This operation runs inflation.

Threshold: Low

Parameters
source (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

LiquidityPoolDeposit

class stellar_sdk.operation.**LiquidityPoolDeposit**(*liquidity_pool_id*, *max_amount_a*, *max_amount_b*, *min_price*, *max_price*, *source=None*)

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

Creates a liquidity pool deposit operation.

Threshold: Medium

See Liquidity Pool Deposit for more information.

Parameters

- **liquidity_pool_id** (str) – The liquidity pool ID.
- **max_amount_a** (str | Decimal) – Maximum amount of first asset to deposit.
- **max_amount_b** (str | Decimal) – Maximum amount of second asset to deposit.
- **min_price** (str | Decimal | *Price*) – Minimum deposit_a/deposit_b price.

- **max_price** (str | Decimal | *Price*) – Maximum deposit_a/deposit_b price.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
LiquidityPoolDeposit

to_xdr_object()

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

Return type
Operation

LiquidityPoolWithdraw

class stellar_sdk.operation.**LiquidityPoolWithdraw**(*liquidity_pool_id*, *amount*, *min_amount_a*, *min_amount_b*, *source=None*)

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

Creates a liquidity pool withdraw operation.

Threshold: Medium

See Liquidity Pool Withdraw for more information.

Parameters

- **liquidity_pool_id** (str) – The liquidity pool ID.
- **amount** (str | Decimal) – Amount of pool shares to withdraw.
- **min_amount_a** (str | Decimal) – Minimum amount of first asset to withdraw.
- **min_amount_b** (str | Decimal) – Minimum amount of second asset to withdraw.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
LiquidityPoolWithdraw

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

See Manage Buy Offer for more information.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – Amount being bought. if set to `0`, delete the offer.
- **price** (*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** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Manage Data for more information.

Parameters

- **data_name** (str) – 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. Up to 64 bytes long.
- **data_value** (str | bytes | None) – If not present then the existing *data_name* will be deleted. If present then this value will be set in the DataEntry. Up to 64 bytes long.
- **source** (*MuxedAccount* | str | None) – The optional source account.

classmethod from_xdr_object(*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

See Manage Sell Offer for more information.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – The total amount you're selling. If `0`, deletes the offer.
- **price** (*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** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

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

See Path Payment Strict Receive for more information.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account to send to.
- **send_asset** (*Asset*) – The *asset* to pay with.
- **send_max** (str | Decimal) – The maximum amount of *send_asset* to send.
- **dest_asset** (*Asset*) – The asset the *destination* will receive.
- **dest_amount** (str | Decimal) – The amount the *destination* receives.
- **path** (Sequence[*Asset*]) – A list of Asset objects to use as the path.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Path Payment Strict Send for more information.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account to send to.
- **send_asset** (*Asset*) – The *asset* to pay with.
- **send_amount** (str | Decimal) – Amount of *send_asset* to send.
- **dest_asset** (*Asset*) – The asset the *destination* will receive.

- **dest_min** (str | Decimal) – The minimum amount of *dest_asset* to be received.
- **path** (Sequence[*Asset*]) – A list of Asset objects to use as the path.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Payment for more information.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account ID.
- **asset** (*Asset*) – The asset to send.
- **amount** (str | Decimal) – The amount to send.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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

See Set Options for more information.

Parameters

- **inflation_dest** (str | None) – Account of the inflation destination.
- **clear_flags** (int | *AuthorizationFlag* | None) – Indicates which flags to clear. For details about the flags, please refer to the Control Access to an Asset - Flag. 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.AuthorizationFlag*
  - AUTHORIZATION_REQUIRED = 1
  - AUTHORIZATION_REVOCABLE = 2
  - AUTHORIZATION_IMMUTABLE = 4
  - AUTHORIZATION_CLAWBACK_ENABLED = 8
- **set_flags** (int | *AuthorizationFlag* | None) – Indicates which flags to set. For details about the flags, please refer to the Control Access to an Asset - Flag. 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.AuthorizationFlag*
  - AUTHORIZATION_REQUIRED = 1
  - AUTHORIZATION_REVOCABLE = 2
  - AUTHORIZATION_IMMUTABLE = 4
  - AUTHORIZATION_CLAWBACK_ENABLED = 8
- **master_weight** (int | None) – 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** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a low threshold.
- **med_threshold** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a medium threshold.
- **high_threshold** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a high threshold.
- **home_domain** (str | None) – sets the home domain used for reverse federation lookup.
- **signer** (*Signer* | None) – Add, update, or remove a signer from the account.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*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.**AuthorizationFlag**(**values*)

Indicates which flags to set. For details about the flags, please refer to the Control Access to an Asset - Flag.

CreateClaimableBalance

class stellar_sdk.operation.**CreateClaimableBalance**(*asset*, *amount*, *claimants*, *source=None*)

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

Creates a ClaimableBalanceEntry. See Claimable Balance for more information on parameters and usage.

Threshold: Medium

See Create Claimable Balance for more information.

Parameters

- **asset** (*Asset*) – The asset for the claimable balance.
- **amount** (str | Decimal) – the amount of the asset.
- **claimants** (Sequence[*Claimant*]) – A list of Claimants.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
CreateClaimableBalance

to_xdr_object()

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

Return type
Operation

class stellar_sdk.operation.**Claimant**(*destination*, *predicate=None*)

The *Claimant* object represents a claimable balance claimant.

Parameters

- **destination** (str) – The destination account ID.
- **predicate** (*ClaimPredicate* | None) – The claim predicate. It is optional, it defaults to unconditional if none is specified.

class stellar_sdk.operation.**ClaimPredicate**(*claim_predicate_type*, *and_predicates*, *or_predicates*, *not_predicate*, *abs_before*, *rel_before*)

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

We do not recommend that you build it through the constructor, please use the helper function.

Parameters

- **claim_predicate_type** (*ClaimPredicateType*) – Type of ClaimPredicate.

- **and_predicates** (*ClaimPredicateGroup* | None) – The ClaimPredicates.
- **or_predicates** (*ClaimPredicateGroup* | None) – The ClaimPredicates.
- **not_predicate** (*ClaimPredicate* | None) – The ClaimPredicate.
- **abs_before** (int | None) – Unix epoch.
- **rel_before** (int | None) – seconds since closeTime of the ledger in which the ClaimableBalanceEntry was created.

classmethod predicate_and(*left*, *right*)

Returns an **and** claim predicate

Parameters

- **left** (*ClaimPredicate*) – a ClaimPredicate.
- **right** (*ClaimPredicate*) – a ClaimPredicate.

Return type
ClaimPredicate

Returns
an **and** claim predicate.

classmethod predicate_before_absolute_time(*abs_before*)

Returns a **before_absolute_time** claim predicate.

This predicate will be fulfilled if the closing time of the ledger that includes the *CreateClaimableBalance* operation is less than this (absolute) Unix timestamp.

Parameters
abs_before (int) – Unix epoch.

Return type
ClaimPredicate

Returns
a **before_absolute_time** claim predicate.

classmethod predicate_before_relative_time(*seconds*)

Returns a **before_relative_time** claim predicate.

This predicate will be fulfilled if the closing time of the ledger that includes the *CreateClaimableBalance* operation plus this relative time delta (in seconds) is less than the current time.

Parameters
seconds (int) – seconds since closeTime of the ledger in which the ClaimableBalanceEntry was created.

Return type
ClaimPredicate

Returns
a **before_relative_time** claim predicate.

classmethod predicate_not(*predicate*)

Returns a **not** claim predicate.

Parameters
predicate (*ClaimPredicate*) – a ClaimPredicate.

Return type
ClaimPredicate

Returns
a **not** claim predicate.

classmethod predicate_or(*left*, *right*)

Returns an **or** claim predicate

Parameters

- **left** (*ClaimPredicate*) – a ClaimPredicate.
- **right** (*ClaimPredicate*) – a ClaimPredicate.

Return type
ClaimPredicate

Returns
an **or** claim predicate.

classmethod predicate_unconditional()

Returns an **unconditional** claim predicate.

Return type
ClaimPredicate

Returns
an **unconditional** claim predicate.

class stellar_sdk.operation.create_claimable_balance.**ClaimPredicateType**(**values*)

Currently supported claim predicate types.

class stellar_sdk.operation.create_claimable_balance.**ClaimPredicateGroup**(*left*, *right*)

Used to assemble the left and right values for and_predicates and or_predicates.

Parameters

- **left** (*ClaimPredicate*) – The ClaimPredicate.
- **right** (*ClaimPredicate*) – The ClaimPredicate.

ClaimClaimableBalance

class stellar_sdk.operation.**ClaimClaimableBalance**(*balance_id*, *source=None*)

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

Claims a ClaimableBalanceEntry and adds the amount of asset on the entry to the source account.

Threshold: Low

See Claim Claimable Balance for more information.

Parameters

- **balance_id** (str) – The claimable balance id to be claimed.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
ClaimClaimableBalance

to_xdr_object()

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

Return type
Operation

BeginSponsoringFutureReserves

class stellar_sdk.operation.**BeginSponsoringFutureReserves**(*sponsored_id*, *source=None*)

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

Establishes the is-sponsoring-future-reserves-for relationship between the source account and sponsoredID. See Sponsored Reserves for more information.

Threshold: Medium

See Begin Sponsoring Future Reserves for more information.

Parameters

- **sponsored_id** (str) – The sponsored account id.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
BeginSponsoringFutureReserves

to_xdr_object()

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

Return type
Operation

EndSponsoringFutureReserves

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

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

Terminates the current is-sponsoring-future-reserves-for relationship in which the source account is sponsored. See Sponsored Reserves for more information.

Threshold: Medium

See End Sponsoring Future Reserves.

Parameters
source (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
EndSponsoringFutureReserves

to_xdr_object()

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

Return type
Operation

RevokeSponsorship

class stellar_sdk.operation.**RevokeSponsorship**(*revoke_sponsorship_type*, *account_id*, *trustline*, *offer*, *data*, *claimable_balance_id*, *signer*, *liquidity_pool_id*, *source=None*)

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

The logic of this operation depends on the state of the source account.

If the source account is not sponsored or is sponsored by the owner of the specified entry or sub-entry, then attempt to revoke the sponsorship. If the source account is sponsored, the next step depends on whether the entry is sponsored or not. If it is sponsored, attempt to transfer the sponsorship to the sponsor of the source account. If the entry is not sponsored, then establish the sponsorship. See Sponsored Reserves for more information.

Threshold: Medium

See Revoke Sponsorship for more information.

Parameters

- **revoke_sponsorship_type** (*RevokeSponsorshipType*) – The sponsored account id.
- **account_id** (str | None) – The sponsored account ID.
- **trustline** (*TrustLine* | None) – The sponsored trustline.
- **offer** (*Offer* | None) – The sponsored offer.
- **data** (*Data* | None) – The sponsored data.
- **claimable_balance_id** (str | None) – The sponsored claimable balance.
- **signer** (*Signer* | None) – The sponsored signer.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
RevokeSponsorship

to_xdr_object()

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

Return type
Operation

class stellar_sdk.operation.revoke_sponsorship.**RevokeSponsorshipType**(**values*)

Currently supported RevokeSponsorship types.

class stellar_sdk.operation.revoke_sponsorship.**TrustLine**(*account_id*, *asset*)

class stellar_sdk.operation.revoke_sponsorship.**Offer**(*seller_id*, *offer_id*)

class stellar_sdk.operation.revoke_sponsorship.**Data**(*account_id*, *data_name*)

class stellar_sdk.operation.revoke_sponsorship.**Signer**(*account_id*, *signer_key*)

Clawback

class stellar_sdk.operation.**Clawback**(*asset*, *from_*, *amount*, *source=None*)

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

Claws back an amount of an asset from an account.

Threshold: Medium

See Clawback for more information.

Parameters

- **asset** (*Asset*) – The asset being clawed back.
- **from** – The public key of the account to claw back from.
- **amount** (str | Decimal) – The amount of the asset to claw back.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod **from_xdr_object**(*xdr_object*)

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

Return type
Clawback

to_xdr_object()

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

Return type
Operation

ClawbackClaimableBalance

class stellar_sdk.operation.**ClawbackClaimableBalance**(*balance_id*, *source=None*)

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

Claws back a claimable balance

Threshold: Medium

See Clawback Claimable Balance for more information.

Parameters

- **balance_id** (str) – The claimable balance ID to be clawed back.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
ClawbackClaimableBalance

to_xdr_object()

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

Return type
Operation

SetTrustLineFlags

class stellar_sdk.operation.**SetTrustLineFlags**(*trustor*, *asset*, *clear_flags=None*, *set_flags=None*, *source=None*)

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

Updates the flags of an existing trust line. This is called by the issuer of the related asset.

Threshold: Low

See Set Trustline Flags for more information.

Parameters

- **trustor** (str) – The account whose trustline this is.
- **asset** (*Asset*) – The asset on the trustline.
- **clear_flags** (*TrustLineFlags* | None) – The flags to clear.
- **set_flags** (*TrustLineFlags* | None) – The flags to set.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
SetTrustLineFlags

to_xdr_object()

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

Return type
Operation

class stellar_sdk.operation.set_trust_line_flags.**TrustLineFlags**(**values*)

Indicates which flags to set. For details about the flags, please refer to the CAP-0035.

- **AUTHORIZED_FLAG**: issuer has authorized account to perform transactions with its credit
- **AUTHORIZED_TO_MAINTAIN_LIABILITIES_FLAG**: issuer has authorized account to maintain and reduce liabilities for its credit
- **TRUSTLINE_CLAWBACK_ENABLED_FLAG**: issuer has specified that it may clawback its credit, and that claimable balances created with its credit may also be clawed back

InvokeHostFunction

class stellar_sdk.operation.**InvokeHostFunction**(*host_function*, *auth=None*, *source=None*)

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

Threshold: Medium

See Invoke Host Function. See Interacting with Soroban via Stellar.

Parameters

- **host_function** (*HostFunction*) – The host function to invoke.
- **auth** (Sequence[*SorobanAuthorizationEntry*] | None) – The authorizations required to execute the host function.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
InvokeHostFunction

to_xdr_object()

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

Return type
Operation

ExtendFootprintTTL

class stellar_sdk.operation.**ExtendFootprintTTL**(*extend_to*, *source=None*)

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

Threshold: Low

See Extend Footprint TTL.

Parameters

- **extend_to** (int) – The number of ledgers past the LCL (last closed ledger) by which to extend the validity of the ledger keys in this transaction.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
ExtendFootprintTTL

to_xdr_object()

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

Return type
Operation

RestoreFootprint

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

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

Threshold: Low

See Restore Footprint.

Parameters
source (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

classmethod from_xdr_object(*xdr_object*)

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

Return type
RestoreFootprint

to_xdr_object()

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

Return type
Operation

2.1.16 Price

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

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

An example:

```
from stellar_sdk import Price

price_a = Price(1, 2)
price_b = Price.from_raw_price("0.5")
```

Parameters

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

classmethod from_raw_price(*price*)

Create a *Price* from the given str or Decimal price.

Parameters
price (str | Decimal) – the str or Decimal price. (ex. "0.125")

Return type
Price

Returns
A new *Price* object from the given str or Decimal price.

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

classmethod from_xdr_object(*xdr_object*)

Create a *Price* from an XDR Price object.

Parameters
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

2.1.17 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.

An example:

```
from stellar_sdk import Server

server = Server("https://horizon-testnet.stellar.org")
resp = server.transactions().limit(10).order(desc=True).call()
print(resp)
```

Parameters

- **horizon_url** (str) – Horizon Server URL (ex. "https://horizon-testnet.stellar.org" for test network, "https://horizon.stellar.org" for public network)
- **client** (*BaseSyncClient* | None) – Http client used to send the request

accounts()

Return type
AccountsCallBuilder

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

assets()

Return type
AssetsCallBuilder

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

claimable_balances()

Return type
ClaimableBalancesCallBuilder

Returns
New *stellar_sdk.call_builder.call_builder_sync.ClaimableBalancesCallBuilder* object configured by a current Horizon server configuration.

close()

Close underlying connector, and release all acquired resources.

Return type
None

data(*account_id*, *data_name*)

Return type
DataCallBuilder

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

effects()

Return type
EffectsCallBuilder

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

fee_stats()

Return type
FeeStatsCallBuilder

Returns
New *stellar_sdk.call_builder.call_builder_sync.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
int

Returns
the base fee

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError

ledgers()

Return type
LedgersCallBuilder

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

`liquidity_pools`()

Return type
LiquidityPoolsBuilder

Returns
New *stellar_sdk.call_builder.call_builder_sync.LiquidityPoolsBuilder* object configured by a current Horizon server configuration.

`load_account`(*account_id*)

Fetches an account's most current base state (like sequence) in the ledger and then creates and returns an *stellar_sdk.account.Account* object.

If you want to get complete account information, please use *stellar_sdk.server.Server.accounts()*.

Parameters
account_id (*MuxedAccount* | *Keypair* | str) – The account to load.

Return type
Account

Returns
an *stellar_sdk.account.Account* object.

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError

`offers`()

Return type
OffersCallBuilder

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

`operations`()

Return type
OperationsCallBuilder

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

payments()

Return type
PaymentsCallBuilder

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

root()

Return type
RootCallBuilder

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

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

Parameters

- **source** (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 | Decimal) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

Return type
StrictReceivePathsCallBuilder

Returns
New *stellar_sdk.call_builder.call_builder_sync.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 | Decimal) – The amount, denominated in the source asset, that any returned path should be able to satisfy.
- **destination** (str | list[*Asset*]) – The destination account or the destination assets.

Return type
StrictSendPathsCallBuilder

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

submit_transaction(*transaction_envelope*, *skip_memo_required_check=False*)

Submits a transaction to the network.

Parameters

- **transaction_envelope** (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – *stellar_sdk.transaction_envelope.TransactionEnvelope* object or base64 encoded xdr
- **skip_memo_required_check** (bool) – Allow skipping memo

Return type
dict[str, Any]

Returns
the response from horizon

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError *AccountRequiresMemoError*

submit_transaction_async(*transaction_envelope*, *skip_memo_required_check=False*)

Submits an asynchronous transaction to the network. Unlike the synchronous version, which blocks and waits for the transaction to be ingested in Horizon, this endpoint relays the response from core directly back to the user.

See Horizon Documentation - Submit a Transaction Asynchronously

Parameters

- **transaction_envelope** (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – *stellar_sdk.transaction_envelope.TransactionEnvelope* object or base64 encoded xdr
- **skip_memo_required_check** (bool) – Allow skipping memo

Return type
dict[str, Any]

Returns
the response from horizon

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError *AccountRequiresMemoError*

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** (int | None) – lower time boundary represented as millis since epoch
- **end_time** (int | None) – upper time boundary represented as millis since epoch

- **offset** (int | None) – 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.call_builder_sync.TradeAggregationsCallBuilder* object configured by a current Horizon server configuration.

trades()

Return type
TradesCallBuilder

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

transactions()

Return type
TransactionsCallBuilder

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

2.1.18 ServerAsync

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

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

An example:

```
import asyncio
from stellar_sdk import ServerAsync

async def example():
    async with ServerAsync("https://horizon-testnet.stellar.org") as server:
        resp = await server.transactions().limit(10).order(desc=True).call()
        print(resp)

asyncio.run(example())
```

Parameters

- **horizon_url** (str) – Horizon Server URL (ex. "https://horizon-testnet.stellar.org" for test network, "https://horizon.stellar.org" for public network)
- **client** (*BaseAsyncClient* | None) – Http client used to send the request

`accounts()`

Return type
AccountsCallBuilder

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

`assets()`

Return type
AssetsCallBuilder

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

`claimable_balances()`

Return type
ClaimableBalancesCallBuilder

Returns
New *stellar_sdk.call_builder.call_builder_async.ClaimableBalancesCallBuilder* object configured by a current Horizon server configuration.

`async close()`

Close underlying connector, and release all acquired resources.

Return type
None

`data(account_id, data_name)`

Return type
DataCallBuilder

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

`effects()`

Return type
EffectsCallBuilder

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

`fee_stats()`

Return type
FeeStatsCallBuilder

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

async 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
int

Returns
the base fee

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError

ledgers()

Return type
LedgersCallBuilder

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

liquidity_pools()

Return type
LiquidityPoolsBuilder

Returns
New *stellar_sdk.call_builder.call_builder_async.LiquidityPoolsBuilder* object configured by a current Horizon server configuration.

async load_account(*account_id*)

Fetches an account's most current base state (like sequence) in the ledger and then creates and returns an *stellar_sdk.account.Account* object.

If you want to get complete account information, please use *stellar_sdk.server.Server.accounts()*.

Parameters
account_id (*MuxedAccount* | *Keypair* | str) – The account to load.

Return type
Account

Returns
an *stellar_sdk.account.Account* object.

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError

offers()

Return type
OffersCallBuilder

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

operations()

Return type
OperationsCallBuilder

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

payments()

Return type
PaymentsCallBuilder

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

root()

Return type
RootCallBuilder

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

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

Parameters

- **source** (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 | Decimal) – The amount, denominated in the destination asset, that any returned path should be able to satisfy.

Return type
StrictReceivePathsCallBuilder

Returns
New *stellar_sdk.call_builder.call_builder_async.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 | Decimal) – The amount, denominated in the source asset, that any returned path should be able to satisfy.
- **destination** (str | list[*Asset*]) – The destination account or the destination assets.

Return type
StrictSendPathsCallBuilder

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

async submit_transaction(*transaction_envelope*, *skip_memo_required_check=False*)

Submits a transaction to the network.

Parameters

- **transaction_envelope** (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – *stellar_sdk.transaction_envelope.TransactionEnvelope* object or base64 encoded xdr
- **skip_memo_required_check** (bool) – Allow skipping memo

Return type
dict[str, Any]

Returns
the response from horizon

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError *AccountRequiresMemoError*

async submit_transaction_async(*transaction_envelope*, *skip_memo_required_check=False*)

Submits an asynchronous transaction to the network. Unlike the synchronous version, which blocks and waits for the transaction to be ingested in Horizon, this endpoint relays the response from core directly back to the user.

See Horizon Documentation - Submit a Transaction Asynchronously

Parameters

- **transaction_envelope** (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – *stellar_sdk.transaction_envelope.TransactionEnvelope* object or base64 encoded xdr
- **skip_memo_required_check** (bool) – Allow skipping memo

Return type
dict[str, Any]

Returns
the response from horizon

Raises
ConnectionError *NotFoundError* *BadRequestError* *BadResponseError* UnknownRequestError *AccountRequiresMemoError*

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** (int | None) – lower time boundary represented as millis since epoch
- **end_time** (int | None) – upper time boundary represented as millis since epoch
- **offset** (int | None) – 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.call_builder_async.TradeAggregationsCallBuilder* object configured by a current Horizon server configuration.

trades()

Return type
TradesCallBuilder

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

transactions()

Return type
TransactionsCallBuilder

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

2.1.19 Signer

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

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

An example:

```
from stellar_sdk import Signer

signer_ed25519 = Signer.ed25519_public_key(
```

(continues on next page)

(continued from previous page)

```
↪"GCC3U63F5OJIG4VS6XCFUJGCQRRMNCVGASDGIZZEPA3AZ242K4JVPIYV", 1)
signer_sha256_hash = Signer.sha256_hash(
↪"XCC3U63F5OJIG4VS6XCFUJGCQRRMNCVGASDGIZZEPA3AZ242K4JVPRP5", 2)
signer_pre_auth_tx = Signer.pre_auth_tx(
↪"TCC3U63F5OJIG4VS6XCFUJGCQRRMNCVGASDGIZZEPA3AZ242K4JVOVKE", 3)
print(f"signer_ed25519 account id: {signer_ed25519.signer_key.encoded_signer_key}")
print(f"signer_ed25519 weight: {signer_ed25519.weight}")
```

Parameters

- **signer_key** (*SignerKey*) – The signer object
- **weight** (int) – The weight of the key

classmethod ed25519_public_key(*account_id*, *weight*)

Create ED25519 PUBLIC KEY Signer from account id.

Parameters

- **account_id** (str | bytes) – account id (ex. "GDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH2354AD")
- **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(*xdr_object*)

Create a *Signer* from an XDR Signer object.

Parameters
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** (str | bytes) – The sha256 hash of a transaction (ex. "TDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH234BSS" or bytes)
- **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** (str | bytes) – a sha256 hash of a preimage (ex. "XDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH235FXL" or bytes)
- **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

2.1.20 SignerKey

class stellar_sdk.signer_key.**SignerKey**(*signer_key*, *signer_key_type*)

The *SignerKey* object, which represents an account signer key on Stellar's network.

Parameters

- **signer_key** (bytes) – The signer key.
- **signer_key** – The signer key type.

classmethod ed25519_public_key(*account_id*)

Create ED25519 PUBLIC KEY Signer from account id.

Parameters
account_id (str | bytes) – account id

Return type
SignerKey

Returns
ED25519 PUBLIC KEY Signer

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

classmethod ed25519_signed_payload(*ed25519_signed_payload*)

Create ed25519 signed payload Signer from an ed25519 signed payload, click here for more information.

Parameters
ed25519_signed_payload (str | bytes | SignedPayloadSigner) – a sha256 hash of a preimage

Return type
SignerKey

Returns
ed25519 signed payload signer

property encoded_signer_key: str

return: The signer key encoded in Strkey format.

classmethod from_encoded_signer_key(*encoded_signer_key*)

Parse the encoded signer key.

Parameters
encoded_signer_key (str) – The encoded signer key. (ex. GBJCHUKZMTFSLOMNC7P4TS4VJJBTCYL3XKSOLXAUJSD56C4LHND5TWUC)

Return type
SignerKey

Returns
The *SignerKey* object.

classmethod from_xdr_object(*xdr_object*)

Create a *SignerKey* from an XDR SignerKey object.

Parameters
xdr_object (*SignerKey*) – The XDR SignerKey object.

Return type
SignerKey

Returns
A new *SignerKey* object from the given XDR SignerKey object.

classmethod pre_auth_tx(*pre_auth_tx_hash*)

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

Parameters
pre_auth_tx_hash (str | bytes) – The sha256 hash of a transaction.

Return type
SignerKey

Returns
Pre AUTH TX Signer

classmethod sha256_hash(*sha256_hash*)

Create SHA256 HASH Signer from a sha256 hash of a preimage, click here for more information.

Parameters
sha256_hash (str | bytes) – a sha256 hash of a preimage

Return type
SignerKey

Returns
SHA256 HASH Signer

to_xdr_object()

Returns the xdr object for this SignerKey object.

Return type
SignerKey

Returns
XDR Signer object

class stellar_sdk.signer_key.**SignerKeyType**(**values*)

2.1.21 StrKey

class stellar_sdk.strkey.**StrKey**

StrKey is a helper class that allows encoding and decoding strkey.

static **decode_claimable_balance**(*data*)

Decodes encoded claimable balance strkey to raw data (B...).

Parameters
data (str) – encoded claimable balance strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static **decode_contract**(*data*)

Decodes encoded contract strkey (C...) to raw data.

Parameters
data (str) – encoded contract strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static **decode_ed25519_public_key**(*data*)

Decodes encoded ed25519 public key strkey (G...) to raw data.

Parameters
data (str) – encoded ed25519 public key strkey

Return type
bytes

Returns
raw bytes

Raises
Ed25519PublicKeyInvalidError

static **decode_ed25519_secret_seed**(*data*)

Decodes encoded ed25519 secret seed strkey (S...) to raw data.

Parameters
data (str) – encoded ed25519 secret seed strkey

Return type
bytes

Returns
raw bytes

Raises
Ed25519SecretSeedInvalidError

static decode_ed25519_signed_payload(*data*)

Decodes encoded ed25519 signed payload strkey (P...) to raw data.

Parameters
data (str) – encoded ed25519 signed payload strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static decode_liquidity_pool(*data*)

Decodes encoded liquidity pool strkey (L...) to raw data.

Parameters
data (str) – encoded liquidity pool strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static decode_med25519_public_key(*data*)

Decodes encoded med25519 public key strkey (M...) to raw data.

Parameters
data (str) – encoded med25519 public key strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static decode_pre_auth_tx(*data*)

Decodes encoded pre auth tx strkey (T...) to raw data.

Parameters
data (str) – encoded pre auth tx strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static decode_sha256_hash(*data*)

Decodes encoded sha256 hash strkey (X...) to raw data.

Parameters
data (str) – encoded sha256 hash strkey

Return type
bytes

Returns
raw bytes

Raises
ValueError

static encode_claimable_balance(*data*)

Encodes data to encoded claimable balance strkey (B...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded claimable balance strkey

Raises
ValueError

static encode_contract(*data*)

Encodes data to encoded contract strkey (C...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded contract strkey

Raises
ValueError

static encode_ed25519_public_key(*data*)

Encodes data to encoded ed25519 public key strkey (G...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded ed25519 public key strkey

Raises
ValueError

static encode_ed25519_secret_seed(*data*)

Encodes data to encoded ed25519 secret seed strkey (S...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded ed25519 secret seed strkey

Raises
ValueError

static encode_ed25519_signed_payload(*data*)

Encodes data to encoded ed25519 signed payload strkey (P...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded ed25519 signed payload strkey

Raises
ValueError

static encode_liquidity_pool(*data*)

Encodes data to encoded liquidity pool strkey (L...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded liquidity pool strkey

Raises
ValueError

static encode_med25519_public_key(*data*)

Encodes data to encoded med25519 public key strkey (M...).

Parameters
data (bytes) – data to encode, should be 40 bytes long (32 bytes for ed25519 public key + 8 bytes for muxed id)

Return type
str

Returns
encoded med25519 public key strkey

Raises
ValueError

static encode_pre_auth_tx(*data*)

Encodes data to encoded pre auth tx strkey (T...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded pre auth tx strkey

Raises
ValueError

static encode_sha256_hash(*data*)

Encodes data to encoded sha256 hash strkey (X...).

Parameters
data (bytes) – data to encode

Return type
str

Returns
encoded sha256 hash strkey

Raises
ValueError

static is_valid_claimable_balance(*claimable_balance*)

Returns True if the given *claimable_balance* is a valid encoded claimable balance strkey (B...).

Parameters
claimable_balance (str) – encoded claimable balance strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_contract(*contract*)

Returns True if the given *contract* is a valid encoded contract strkey (C...).

Parameters
pre_auth_tx – encoded contract strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_ed25519_public_key(*public_key*)

Returns True if the given *public_key* is a valid ed25519 public key strkey (G...).

Parameters
public_key (str) – encoded ed25519 public key strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_ed25519_secret_seed(*seed*)

Returns True if the given *seed* is a valid ed25519 secret seed strkey (S...).

Parameters
seed (str) – encoded ed25519 secret seed strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_ed25519_signed_payload(*ed25519_signed_payload*)

Returns True if the given *ed25519_signed_payload* is a valid encoded ed25519 signed payload strkey (P...).

Parameters
ed25519_signed_payload (str) – encoded ed25519 signed payload strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_liquidity_pool(*liquidity_pool*)

Returns True if the given *liquidity_pool* is a valid encoded liquidity pool strkey (L...).

Parameters
liquidity_pool (str) – encoded liquidity pool strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_med25519_public_key(*public_key*)

Returns True if the given *public_key* is a valid med25519 public key strkey (G...).

Parameters
public_key (str) – encoded med25519 public key strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_pre_auth_tx(*pre_auth_tx*)

Returns True if the given *pre_auth_tx* is a valid encoded pre auth tx strkey (T...).

Parameters
pre_auth_tx (str) – encoded pre auth tx strkey

Return type
bool

Returns
True if the given key is valid

static is_valid_sha256_hash(*sha256_hash*)

Returns `True` if the given *sha256_hash* is a valid encoded sha256 hash(HashX) strkey (X...).

Parameters
sha256_hash (`str`) – encoded sha256 hash(HashX) strkey

Return type
`bool`

Returns
`True` if the given key is valid

2.1.22 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(*xdr_object*)

Create a *TimeBounds* from an XDR TimeBounds object.

Parameters
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

2.1.23 DecoratedSignature

class stellar_sdk.decorated_signature.**DecoratedSignature**(*signature_hint*, *signature*)

classmethod from_xdr_object(*xdr_object*)

Create a *DecoratedSignature* from an XDR DecoratedSignature object.

Parameters
xdr_object (*DecoratedSignature*) – The XDR DecoratedSignature object.

Return type
DecoratedSignature

Returns
A new *DecoratedSignature* object from the given XDR DecoratedSignature object.

`to_xdr_object()`

Returns the xdr object for this DecoratedSignature object.

Return type
DecoratedSignature

Returns
XDR DecoratedSignature object

2.1.24 Transaction

`class stellar_sdk.transaction.Transaction(source, sequence, fee, operations, memo=None, preconditions=None, soroban_data=None, v1=True)`

The *Transaction* object, which represents a transaction(Transaction or TransactionV0) 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** (*MuxedAccount* | *Keypair* | `str`) – the source account for the transaction.
- **sequence** (`int`) – The sequence number for the transaction.
- **fee** (`int`) – The max fee amount for the transaction, which should equal FEE (currently least 100 stroops) multiplied by the number of operations in the transaction. See Stellar's latest documentation on fees for more information.
- **operations** (`Sequence`[*Operation*]) – A list of operations objects (typically its subclasses as defined in *stellar_sdk.operation.Operation*.
- **preconditions** (`Preconditions` | `None`) – The preconditions for the validity of this transaction.
- **memo** (*Memo* | `None`) – The memo being sent with the transaction, being represented as one of the subclasses of the *Memo* object.
- **soroban_data** (*SorobanTransactionData* | `None`) – The soroban data being sent with the transaction, being represented as `SorobanTransactionData`.
- **v1** (`bool`) – When this value is set to `True`, V1 transactions will be generated, otherwise V0 transactions will be generated. See CAP-0015 for more information.

`classmethod from_xdr(xdr, v1=True)`

Create a new *Transaction* from an XDR string.

Parameters

- **xdr** (`str`) – The XDR string that represents a transaction.

- **v1** (bool) – Temporary feature flag to allow alpha testing of Stellar Protocol 13 transactions. We will remove this once all transactions are supposed to be v1. See CAP-0015 for more information.

Return type
Transaction

Returns
A new *Transaction* object from the given XDR Transaction base64 string object.

classmethod from_xdr_object(*xdr_object*, *v1=True*)

Create a new *Transaction* from an XDR object.

Parameters

- **xdr_object** (*Transaction* | *TransactionV0*) – The XDR object that represents a transaction.
- **v1** (bool) – Temporary feature flag to allow alpha testing of Stellar Protocol 13 transactions. We will remove this once all transactions are supposed to be v1. See CAP-0015 for more information.

Return type
Transaction

Returns
A new *Transaction* object from the given XDR Transaction object.

get_claimable_balance_id(*operation_index*)

Calculate the claimable balance ID for an operation within the transaction.

Parameters
operation_index (int) – the index of the CreateClaimableBalance operation.

Return type
str

Returns
a hex string representing the claimable balance ID.

Raises

IndexError: if *operation_index* is invalid.

TypeError: if operation at *operation_index* is not *FeeBumpTransactionEnvelope*.

to_xdr_object()

Get an XDR object representation of this *Transaction*.

Return type
Transaction | *TransactionV0*

Returns
XDR Transaction object

2.1.25 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** (Sequence[*DecoratedSignature*] | None) – 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 BaseTransactionEnvelope from an XDR string.

Parameters

- **xdr** (str) – The XDR string that represents a transaction envelope.
- **network_passphrase** (str) – which network this transaction envelope is associated with.

Return type
TypeVar(T)

Returns
A new BaseTransactionEnvelope object from the given XDR TransactionEnvelope base64 string object.

classmethod from_xdr_object(*xdr_object*, *network_passphrase*)

Create a new *TransactionEnvelope* from an XDR object.

Parameters

- **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 (*Keypair* | str) – The keypair or secret to use for signing this transaction envelope.

Raise
SignatureExistError: if this signature already exists.

Return type
None

sign_extra_signers_payload(*signer*)

Sign this extra signers' payload with a given keypair.

Note that the signature must not already be in this instance's list of signatures.

Parameters
signer (*Keypair* | str) – The keypair or secret to use for signing this extra signers' payload.

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 | str) – Preimage of hash used as signer, byte hash or hex encoded string

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_transaction_envelope_v1()

Create a new *TransactionEnvelope*, if the internal tx is not v1, we will convert it to v1.

Return type
TransactionEnvelope

to_xdr()

Get the base64 encoded XDR string representing this `BaseTransactionEnvelope`.

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

2.1.26 FeeBumpTransaction

class stellar_sdk.fee_bump_transaction.**FeeBumpTransaction**(*fee_source*, *fee*, *inner_transaction_envelope*)

The *FeeBumpTransaction* object, which represents a fee bump transaction on Stellar's network.

See Fee-Bump Transactions for more information. See CAP-0015 for more information.

Parameters

- **fee_source** (*MuxedAccount* | *Keypair* | `str`) – The account paying for the transaction.
- **fee** (`int`) – The max fee willing to pay for the transaction (**in stroops**).
- **inner_transaction_envelope** (*TransactionEnvelope*) – The TransactionEnvelope to be bumped by the fee bump transaction.

classmethod from_xdr(*xdr*, *network_passphrase*)

Create a new *FeeBumpTransaction* from an XDR string.

Parameters

- **xdr** (`str`) – The XDR string that represents a transaction.
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from.

Return type
FeeBumpTransaction

Returns
A new *FeeBumpTransaction* object from the given XDR FeeBumpTransaction base64 string object.

classmethod from_xdr_object(*xdr_object*, *network_passphrase*)

Create a new *FeeBumpTransaction* from an XDR object.

Parameters

- **xdr_object** (*FeeBumpTransaction*) – The XDR object that represents a fee bump transaction.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from.

Return type
FeeBumpTransaction

Returns
A new *FeeBumpTransaction* object from the given XDR Transaction object.

`to_xdr_object()`

Get an XDR object representation of this *FeeBumpTransaction*.

Return type
FeeBumpTransaction

Returns
XDR Transaction object

2.1.27 FeeBumpTransactionEnvelope

`class stellar_sdk.fee_bump_transaction_envelope.FeeBumpTransactionEnvelope(transaction, network_passphrase, signatures=None)`

The *FeeBumpTransactionEnvelope* object, which represents a fee bump transaction envelope ready to sign and submit to send over the network.

When a fee bump transaction is ready to be prepared for sending over the network, it must be put into a *FeeBumpTransactionEnvelope*, 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.

See Fee-Bump Transactions for more information. See CAP-0015 for more information.

Parameters

- **transaction** (*FeeBumpTransaction*) – The fee bump transaction that is encapsulated in this envelope.
- **signatures** (Sequence[*DecoratedSignature*] | None) – 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 BaseTransactionEnvelope from an XDR string.

Parameters

- **xdr** (str) – The XDR string that represents a transaction envelope.
- **network_passphrase** (str) – which network this transaction envelope is associated with.

Return type
TypeVar(T)

Returns
A new BaseTransactionEnvelope object from the given XDR TransactionEnvelope base64 string object.

classmethod from_xdr_object(*xdr_object*, *network_passphrase*)

Create a new *FeeBumpTransactionEnvelope* from an XDR object.

Parameters

- **xdr_object** (*TransactionEnvelope*) – The XDR object that represents a fee bump transaction envelope.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from.

Return type
FeeBumpTransactionEnvelope

Returns
A new *FeeBumpTransactionEnvelope* 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 (*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 | str) – Preimage of hash used as signer, byte hash or hex encoded string

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 `BaseTransactionEnvelope`.

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

2.1.28 TransactionBuilder

`class stellar_sdk.transaction_builder.TransactionBuilder(source_account, network_passphrase='Test SDF Network ; September 2015', base_fee=100, v1=True)`

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.

Operations can be added to the transaction via their corresponding builder methods, and each returns the *TransactionEnvelope* object, so they can be chained together. After adding the desired operations, call the *build()* method on the TransactionBuilder to return a fully constructed *TransactionEnvelope* that can be signed.

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.Server.load_account()* or *stellar_sdk.server_async.ServerAsync.load_account()* before creating another transaction.

The following code example creates a new transaction with *CreateAccount* and *Payment* operations. The Transaction's source account(alice) first funds *bob*, then sends a payment to *bob*. The built transaction is then signed by *alice_keypair*:

```
# Alice funds Bob with 5 XLM and then pays Bob 10.25 XLM
from stellar_sdk import Server, Asset, Keypair, TransactionBuilder, Network

alice_keypair = Keypair.from_secret(
↪"SBFZCHU5645DOKRWYBXVOXY2ELGJKFRX6VGGPRYUWHQ7PMXXJNDZFMKD")
bob_address = "GA7YNBW5CBTJZ3ZZOWX3ZNBKD6OE7A7IHUQVWMY62W2ZBG2SGZVOOPVH"

server = Server("https://horizon-testnet.stellar.org")
alice_account = server.load_account(alice_keypair.public_key)
network_passphrase = Network.TESTNET_NETWORK_PASSPHRASE
base_fee = 100
transaction = (
    TransactionBuilder(
        source_account=alice_account,
        network_passphrase=network_passphrase,
        base_fee=base_fee,
    )
        .add_text_memo("Hello, Stellar!")
        .append_create_account_op(bob_address, "5")
        .append_payment_op(bob_address, Asset.native(), "10.25")
        .set_timeout(30)
        .build()
)
transaction.sign(alice_keypair)
response = server.submit_transaction(transaction)
print(response)
```

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) – Max fee you're willing to pay per operation in this transaction (**in stroops**).
- **v1** (bool) – When this value is set to True, V1 transactions will be generated, otherwise V0 transactions will be generated. See CAP-0015 for more information.

add_extra_signer(*signer_key*)

For the transaction to be valid, there must be a signature corresponding to every Signer in this array, even if the signature is not otherwise required by the source account or operations. Internally this will set the `SignerKey` precondition.

Parameters

signer_key (*SignerKey* | `SignedPayloadSigner` | str) – The signer key

Return type

TransactionBuilder

Returns

This builder instance.

add_hash_memo(*memo_hash*)

Set the memo for the transaction to a new *HashMemo*.

Parameters
memo_hash (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 (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 (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*)

Sets a timeout precondition on the transaction.

Because of the distributed nature of the Stellar network it is possible that the status of your transaction will be determined after a long time if the network is highly congested. If you want to be sure to receive the status of the transaction within a given period you should set the `TimeBounds` with *max_time* on the transaction (this is what *set_timeout()* does internally).

Please note that Horizon may still return **504 Gateway Timeout** error, even for short timeouts. In such case you need to resubmit the same transaction again without making any changes to receive a status. This method is using the machine system time (UTC), make sure it is set correctly.

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. *max_time* 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** (*MuxedAccount* | `str`) – The ID of the offer. 0 for new offer. Set to existing offer ID to update or delete.
- **source** (*MuxedAccount* | `str` | `None`) – 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** (*TrustLineEntryFlag* | bool) – *True* to authorize the line, *False* to deauthorizeif you need further control, you can also use *stellar_sdk.operation.allow_trust.TrustLineEntryFlag*.
- **source** (*MuxedAccount* | str | None) – The source address that is establishing the trust in the allow trust operation.

Return type
TransactionBuilder

Returns
This builder instance.

append_begin_sponsoring_future_reserves_op(*sponsored_id*, *source=None*)

Append a *BeginSponsoringFutureReserves* operation to the list of operations.

Parameters

- **sponsored_id** (str) – The sponsored account id.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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** (*MuxedAccount* | str | None) – The source address that is running the inflation operation.

Return type
TransactionBuilder

Returns
This builder instance.

append_change_trust_op(*asset*, *limit=None*, *source=None*)

Append a *ChangeTrust* operation to the list of operations.

Parameters

- **asset** (*Asset* | *LiquidityPoolAsset*) – The asset for the trust line.
- **limit** (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** (*MuxedAccount* | str | None) – The source address to add the trustline to.

Return type
TransactionBuilder

Returns
This builder instance.

append_claim_claimable_balance_op(*balance_id*, *source=None*)

Append a *ClaimClaimableBalance* operation to the list of operations.

Parameters

- **balance_id** (str) – The claimable balance id to be claimed.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

append_clawback_claimable_balance_op(*balance_id*, *source=None*)

Append an *ClawbackClaimableBalance* operation to the list of operations.

Parameters

- **balance_id** (str) – The claimable balance ID to be clawed back.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_clawback_op(*asset*, *from_*, *amount*, *source=None*)

Append an *Clawback* operation to the list of operations.

Parameters

- **asset** (*Asset*) – The asset being clawed back.
- **from** – The public key of the account to claw back from.
- **amount** (str | Decimal) – The amount of the asset to claw back.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

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** (str | Decimal) – Amount of XLM to send to the newly created account. This XLM comes from the source account.
- **source** (*MuxedAccount* | str | None) – The source address to deduct funds from to fund the new account.

Return type
TransactionBuilder

Returns
This builder instance.

append_create_claimable_balance_op(*asset*, *amount*, *claimants*, *source=None*)

Append a *CreateClaimableBalance* operation to the list of operations.

Parameters

- **asset** (*Asset*) – The asset for the claimable balance.
- **amount** (str | Decimal) – the amount of the asset.
- **claimants** (Sequence[*Claimant*]) – A list of Claimants.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

append_create_contract_op(*wasm_id*, *address*, *constructor_args=None*, *salt=None*, *auth=None*, *source=None*)

Append an *InvokeHostFunction* operation to the list of operations.

You can use this method to create a contract.

Parameters

- **wasm_id** (bytes | str) – The ID of the contract code to install.
- **address** (str | *Address*) – The address using to derive the contract ID.
- **constructor_args** (Sequence[*SCVal*] | None) – The optional parameters to pass to the constructor of this contract.
- **salt** (bytes | None) – The 32-byte salt to use to derive the contract ID.
- **auth** (Sequence[*SorobanAuthorizationEntry*] | None) – The authorizations required to execute the host function.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_create_passive_sell_offer_op(*selling*, *buying*, *amount*, *price*, *source=None*)

Append a *CreatePassiveSellOffer* operation to the list of operations.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – The total amount you're selling.
- **price** (*Price* | str | Decimal) – Price of 1 unit of *selling* in terms of *buying*.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_create_stellar_asset_contract_from_asset_op(*asset*, *source=None*)

Append an *InvokeHostFunction* operation to the list of operations.

You can use this method to deploy a contract that wraps a classic asset.

Parameters

- **asset** (*Asset*) – The asset to wrap.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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 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. (ex. "GDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH2354AD")
- **weight** (int) – The weight of the new signer.
- **source** (*MuxedAccount* | str | None) – The source account that is adding a signer to its list of signers.

Return type
TransactionBuilder

Returns
This builder instance.

append_end_sponsoring_future_reserves_op(*source=None*)

Append a *EndSponsoringFutureReserves* operation to the list of operations.

Parameters
source (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_extend_footprint_ttl_op(*extend_to*, *source=None*)

Append an *ExtendFootprintTTL* operation to the list of operations.

Parameters

- **extend_to** (int) – The number of ledgers past the LCL (last closed ledger) by which to extend the validity of the ledger keys in this transaction.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_hashx_signer(*sha256_hash*, *weight*, *source=None*)

Add a sha256 hash(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** (bytes | str) – The address of the new sha256 hash(hashX) signer, a 32 byte hash, hex encoded string or encode strkey. (ex. `"XDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH235FXL"`, `"da0d57da7d4850e7fc10d2a9d0ebc731f7afb40574c03395b17d49149b91f5be"` or bytes)
- **weight** (int) – The weight of the new signer.
- **source** (*MuxedAccount* | str | None) – 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 (*MuxedAccount* | str | None) – The source address that is running the inflation operation.

Return type
TransactionBuilder

Returns
This builder instance.

append_invoke_contract_function_op(*contract_id*, *function_name*, *parameters=None*, *auth=None*, *source=None*)

Append an *InvokeHostFunction* operation to the list of operations.

You can use this method to invoke a contract function.

Parameters

- **contract_id** (str) – The ID of the contract to invoke.
- **function_name** (str) – The name of the function to invoke.
- **parameters** (Sequence[*SCVal*] | None) – The parameters to pass to the method.
- **auth** (Sequence[*SorobanAuthorizationEntry*] | None) – The authorizations required to execute the host function.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_liquidity_pool_deposit_op(*liquidity_pool_id*, *max_amount_a*, *max_amount_b*, *min_price*, *max_price*, *source=None*)

Append an *LiquidityPoolDeposit* operation to the list of operations.

Parameters

- **liquidity_pool_id** (str) – The liquidity pool ID.
- **max_amount_a** (str | Decimal) – Maximum amount of first asset to deposit.
- **max_amount_b** (str | Decimal) – Maximum amount of second asset to deposit.
- **min_price** (str | Decimal | *Price*) – Minimum deposit_a/deposit_b price.
- **max_price** (str | Decimal | *Price*) – Maximum deposit_a/deposit_b price.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_liquidity_pool_withdraw_op(*liquidity_pool_id*, *amount*, *min_amount_a*, *min_amount_b*, *source=None*)

Append an *LiquidityPoolWithdraw* operation to the list of operations.

Parameters

- **liquidity_pool_id** (str) – The liquidity pool ID.
- **amount** (str | Decimal) – Amount of pool shares to withdraw.
- **min_amount_a** (str | Decimal) – Minimum amount of first asset to withdraw.
- **min_amount_b** (str | Decimal) – Minimum amount of second asset to withdraw.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_manage_buy_offer_op(*selling*, *buying*, *amount*, *price*, *offer_id=0*, *source=None*)

Append a *ManageBuyOffer* operation to the list of operations.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – Amount being bought. if set to `0`, delete the offer.
- **price** (*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** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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) – 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. Up to 64 bytes long.
- **data_value** (str | bytes | None) – If not present then the existing *data_name* will be deleted. If present then this value will be set in the DataEntry. Up to 64 bytes long.
- **source** (*MuxedAccount* | str | None) – The source account on which data is being managed. operation.

Return type
TransactionBuilder

Returns
This builder instance.

append_manage_sell_offer_op(*selling*, *buying*, *amount*, *price*, *offer_id=0*, *source=None*)

Append a *ManageSellOffer* operation to the list of operations.

Parameters

- **selling** (*Asset*) – What you're selling.
- **buying** (*Asset*) – What you're buying.
- **amount** (str | Decimal) – The total amount you're selling. If 0, deletes the offer.
- **price** (*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** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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_strict_receive_op(*destination*, *send_asset*, *send_max*, *dest_asset*, *dest_amount*, *path*, *source=None*)

Append a *PathPaymentStrictReceive* operation to the list of operations.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account to send to.
- **send_asset** (*Asset*) – The *asset* to pay with.
- **send_max** (str | Decimal) – The maximum amount of *send_asset* to send.
- **dest_asset** (*Asset*) – The asset the *destination* will receive.
- **dest_amount** (str | Decimal) – The amount the *destination* receives.
- **path** (Sequence[*Asset*]) – A list of Asset objects to use as the path.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_path_payment_strict_send_op(*destination*, *send_asset*, *send_amount*, *dest_asset*, *dest_min*, *path*, *source=None*)

Append a *PathPaymentStrictSend* operation to the list of operations.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account to send to.
- **send_asset** (*Asset*) – The *asset* to pay with.
- **send_amount** (str | Decimal) – Amount of *send_asset* to send.
- **dest_asset** (*Asset*) – The asset the *destination* will receive.
- **dest_min** (str | Decimal) – The minimum amount of *dest_asset* to be received.
- **path** (Sequence[*Asset*]) – A list of Asset objects to use as the path.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_payment_op(*destination*, *asset*, *amount*, *source=None*)

Append a *Payment* operation to the list of operations.

Parameters

- **destination** (*MuxedAccount* | str) – The destination account ID.

- **asset** (*Asset*) – The asset to send.
- **amount** (str | Decimal) – The amount to send.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_payment_to_contract_op(*destination*, *asset*, *amount*, *instructions=400000*, *disk_read_bytes=1000*, *write_bytes=1000*, *resource_fee=5000000*, *source=None*)

Append an *InvokeHostFunction* operation to send assets to a contract address.

The original intention of this interface design is to send assets to the contract account when the Stellar RPC server is inaccessible. Without Stellar RPC, we cannot accurately estimate the required resources, so we have preset some values that may be slightly higher than the actual resource consumption.

If you encounter the *entry_archived* error when submitting this transaction, you should consider calling the *append_restore_asset_balance_entry_op()* method to restore the entry, and then use the *append_payment_to_contract_op()* method to send assets again.

You can find the example code in the examples/send_asset_to_contract_without_rpc.py.

Note

1. This method should only be used to send assets to contract addresses (starting with 'C'). For sending assets to regular account addresses (starting with 'G'), please use the *append_payment_op()* method.
2. This method is suitable for sending assets to a contract account when you don't have access to a Stellar RPC server. If you have access to a Stellar RPC server, it is recommended to use the *stellar_sdk.contract.ContractClient* to build transactions for sending tokens to contracts.
3. This method may consume slightly more transaction fee than actually required.

Parameters

- **destination** (str) – The contract address to send the assets to.
- **asset** (*Asset*) – The asset to send.
- **amount** (str | Decimal) – The amount of the asset to send.
- **instructions** (int) – The instructions required to execute the contract function.
- **disk_read_bytes** (int) – The disk read bytes required to execute the contract function.
- **write_bytes** (int) – The write bytes required to execute the contract function.
- **resource_fee** (int) – The maximum fee (in stroops) that can be paid for the resources consumed by the contract function, defaults to 0.5 XLM. The actual consumption is generally much lower than this value.

- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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 via a SetOptions <stellar_sdk.operation.SetOptions operation. This is a helper function for *append_set_options_op()*.

Parameters

- **pre_auth_tx_hash** (str | bytes) – The address of the new preAuthTx signer - obtained by calling *hash* on the *TransactionEnvelope*, a 32 byte hash, hex encoded string or encode strkey. (ex. "TDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH234BSS", "da0d57da7d4850e7fc10d2a9d0ebc731f7afb40574c03395b17d49149b91f5be" or bytes)
- **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_restore_asset_balance_entry_op(*balance_owner*, *asset*, *disk_read_bytes=500*, *write_bytes=500*, *resource_fee=4000000*, *source=None*)

Append an *RestoreFootprint* operation to restore the asset balance entry.

This method is designed to be used in conjunction with the *append_payment_to_contract_op()* method.

Parameters

- **balance_owner** (str) – The owner of the asset, it should be the same as the *destination* address in the *append_payment_to_contract_op()* method.
- **asset** (*Asset*) – The asset
- **disk_read_bytes** (int) – The disk read bytes required to execute the function.
- **write_bytes** (int) – The write bytes required to execute the function.
- **resource_fee** (int) – The maximum fee (in stroops) that can be paid for the resources consumed by the function, defaults to 0.4 XLM.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_restore_footprint_op(*source=None*)

Append an *RestoreFootprint* operation to the list of operations.

Parameters
source (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Returns
This builder instance.

append_revoke_account_sponsorship_op(*account_id*, *source=None*)

Append a *RevokeSponsorship* operation for an account to the list of operations.

Parameters

- **account_id** (str) – The sponsored account ID.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_claimable_balance_sponsorship_op(*claimable_balance_id*, *source=None*)

Append a *RevokeSponsorship* operation for a claimable to the list of operations.

Parameters

- **claimable_balance_id** (str) – The sponsored claimable balance ID.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_data_sponsorship_op(*account_id*, *data_name*, *source=None*)

Append a *RevokeSponsorship* operation for a data entry to the list of operations.

Parameters

- **account_id** (str) – The account ID which owns the data entry.
- **data_name** (str) – The name of the data entry
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_ed25519_public_key_signer_sponsorship_op(*account_id*, *signer_key*, *source=None*)

Append a *RevokeSponsorship* operation for an ed25519_public_key signer to the list of operations.

Parameters

- **account_id** (str) – The account ID where the signer sponsorship is being removed from.
- **signer_key** (str) – The account id of the ed25519_public_key signer. (ex. "GDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH2354AD")
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_hashx_signer_sponsorship_op(*account_id*, *signer_key*, *source=None*)

Append a *RevokeSponsorship* operation for a hashx signer to the list of operations.

Parameters

- **account_id** (str) – The account ID where the signer sponsorship is being removed from.
- **signer_key** (bytes | str) – The account id of the hashx signer. (ex. "XDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH235FXL", "da0d57da7d4850e7fc10d2a9d0ebc731f7afb40574c03395b17d49149b91f5be" or bytes)
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_liquidity_pool_sponsorship_op(*liquidity_pool_id*, *source=None*)

Append a *RevokeSponsorship* operation for a claimable to the list of operations.

Parameters

- **liquidity_pool_id** (str) – The sponsored liquidity pool ID in hex string.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_offer_sponsorship_op(*seller_id*, *offer_id*, *source=None*)

Append a *RevokeSponsorship* operation for an offer to the list of operations.

Parameters

- **seller_id** (str) – The account ID which created the offer.
- **offer_id** (int) – The offer ID.

- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_pre_auth_tx_signer_sponsorship_op(*account_id*, *signer_key*, *source=None*)

Append a *RevokeSponsorship* operation for a pre_auth_tx signer to the list of operations.

Parameters

- **account_id** (str) – The account ID where the signer sponsorship is being removed from.
- **signer_key** (bytes | str) – The account id of the pre_auth_tx signer. (ex. "TDNA2V62PVEFBZ74CDJKTUHLY4Y7PL5UAV2MAM4VWF6USFE3SH234BSS", "da0d57da7d4850e7fc10d2a9d0ebc731f7afb40574c03395b17d49149b91f5be" or bytes)
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_revoke_trustline_sponsorship_op(*account_id*, *asset*, *source=None*)

Append a *RevokeSponsorship* operation for a trustline to the list of operations.

Parameters

- **account_id** (str) – The account ID which owns the trustline.
- **asset** (*Asset* | *LiquidityPoolId*) – The asset in the trustline.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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** (str | None) – Account of the inflation destination.
- **clear_flags** (int | *AuthorizationFlag* | None) – Indicates which flags to clear. For details about the flags, please refer to the Control Access to an Asset - Flag. 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.AuthorizationFlag*

- AUTHORIZATION_REQUIRED = 1
- AUTHORIZATION_REVOCABLE = 2
- AUTHORIZATION_IMMUTABLE = 4
- AUTHORIZATION_CLAWBACK_ENABLED = 8

- **set_flags** (int | *AuthorizationFlag* | None) – Indicates which flags to set. For details about the flags, please refer to the Control Access to an Asset - Flag. 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.AuthorizationFlag*
  - AUTHORIZATION_REQUIRED = 1
  - AUTHORIZATION_REVOCABLE = 2
  - AUTHORIZATION_IMMUTABLE = 4
  - AUTHORIZATION_CLAWBACK_ENABLED = 8
- **master_weight** (int | None) – 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** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a low threshold.
- **med_threshold** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a medium threshold.
- **high_threshold** (int | None) – A number from 0-255 (inclusive) representing the threshold this account sets on all operations it performs that have a high threshold.
- **home_domain** (str | None) – sets the home domain used for reverse federation lookup.
- **signer** (*Signer* | None) – Add, update, or remove a signer from the account.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

append_set_trust_line_flags_op(*trustor*, *asset*, *clear_flags=None*, *set_flags=None*, *source=None*)

Append an *SetTrustLineFlags* operation to the list of operations.

Parameters

- **trustor** (str) – The account whose trustline this is.
- **asset** (*Asset*) – The asset on the trustline.
- **clear_flags** (*TrustLineFlags* | None) – The flags to clear.
- **set_flags** (*TrustLineFlags* | None) – The flags to set.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

Return type
TransactionBuilder

Returns
This builder instance.

`append_upload_contract_wasm_op`(*contract*, *source=None*)

Append an *InvokeHostFunction* operation to the list of operations.

You can use this method to install a contract code, and then use *append_create_contract_op()* to create a contract.

Parameters

- **contract** (bytes | str) – The contract code to install, path to a file or bytes.
- **source** (*MuxedAccount* | str | None) – The source account for the operation. Defaults to the transaction's source account.

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
New transaction envelope.

`static build_fee_bump_transaction`(*fee_source*, *base_fee*, *inner_transaction_envelope*, *network_passphrase='Test SDF Network ; September 2015'*)

Create a *FeeBumpTransactionEnvelope* object.

See CAP-0015 for more information.

Parameters

- **fee_source** (*MuxedAccount* | *Keypair* | str) – The account paying for the transaction.
- **base_fee** (int) – The max fee willing to pay per operation in inner transaction (**in stroops**).
- **inner_transaction_envelope** (*TransactionEnvelope*) – The TransactionEnvelope to be bumped by the fee bump transaction.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from.

Return type
FeeBumpTransactionEnvelope

Returns
a TransactionBuilder via the XDR object.

`static from_xdr`(*xdr*, *network_passphrase*)

When you are not sure whether your XDR belongs to *TransactionEnvelope* or *FeeBumpTransactionEnvelope*, you can use this function.

An example:

```
from stellar_sdk import Network, TransactionBuilder

xdr = "AAAAAgAAAADHJNEDn33/C1uDkDfzDfKVq/
↪4XE9IxDfGiLCfoV7riZQAAA+gCI4TVABpRPgAAAAAAAAAAAAAAAQAAAAAAAAADAAAAAUxpcmEAAAAAabIaDgm0ypyJps
↪kEB2Z4UL20y536evnwmmSc4c2FnxlvUcPZl5jgWHcNwY8LTpFhdrUN9TZWciCRp/JCZYa0SJh8cYB
↪"
te = TransactionBuilder.from_xdr(xdr, Network.PUBLIC_NETWORK_PASSPHRASE)
print(te)
```

Parameters

- **xdr** (str) – Transaction envelope XDR
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from. (ex. `"Public Global Stellar Network ; September 2015"`)

Raises

ValueError - XDR is neither *TransactionEnvelope* nor *FeeBumpTransactionEnvelope*

Return type

TransactionEnvelope | *FeeBumpTransactionEnvelope*

`set_ledger_bounds`(*min_ledger*, *max_ledger*)

If you want to prepare a transaction which will only be valid within some range of ledgers, you can set a *ledger_bounds* precondition. Internally this will set the `LedgerBounds` preconditions.

Parameters

- **min_ledger** (int) – The minimum ledger this transaction is valid at, or after. Cannot be negative. If the value is `0`, the transaction is valid immediately.
- **max_ledger** (int) – The maximum ledger this transaction is valid before. Cannot be negative. If the value is `0`, the transaction is valid indefinitely.

Return type

TransactionBuilder

Returns

This builder instance.

`set_min_sequence_age`(*min_sequence_age*)

For the transaction to be valid, the current ledger time must be at least *min_sequence_age* greater than source account's *sequence_time*. Internally this will set the *min_sequence_age* precondition.

Parameters

min_sequence_age (int) – The minimum amount of time between source account sequence time and the ledger time when this transaction will become valid. If the value is `0` or `None`, the transaction is unrestricted by the account sequence age. Cannot be negative.

Return type

TransactionBuilder

Returns

This builder instance.

`set_min_sequence_ledger_gap`(*min_sequence_ledger_gap*)

For the transaction to be valid, the current ledger number must be at least *min_sequence_ledger_gap* greater than source account's ledger sequence. Internally this will set the *min_sequence_ledger_gap* precondition.

Parameters
min_sequence_ledger_gap (int) – The minimum number of ledgers between source account sequence and the ledger number when this transaction will become valid. If the value is `0` or `None`, the transaction is unrestricted by the account sequence ledger. Cannot be negative.

Return type
TransactionBuilder

Returns
This builder instance.

`set_min_sequence_number`(*min_sequence_number*)

If you want to prepare a transaction which will be valid only while the account sequence number is **min_sequence_number <= source_account_sequence_number < tx.sequence**.

Note that after execution the account's sequence number is always raised to *tx.sequence*. Internally this will set the *min_sequence_number* precondition.

Parameters
min_sequence_number (int) – The minimum source account sequence number this transaction is valid for. If the value is `None` the transaction is valid when **source account's sequence number == tx.sequence - 1**.

Return type
TransactionBuilder

Returns
This builder instance.

`set_soroban_data`(*soroban_data*)

Set the SorobanTransactionData. For non-contract(non-Soroban) transactions, this setting has no effect.

In the case of Soroban transactions, set to an instance of SorobanTransactionData. This can typically be obtained from the simulation response based on a transaction with a InvokeHostFunctionOp. It provides necessary resource estimations for contract invocation.

Parameters
soroban_data (*SorobanTransactionData* | str) – The SorobanTransactionData as XDR object or base64 encoded string.

Return type
TransactionBuilder

Returns
This builder instance.

`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.

2.1.29 SorobanDataBuilder

class stellar_sdk.**SorobanDataBuilder**

Supports building `Memo` structures with various items set to specific values.

This is recommended for when you are building *RestoreFootprint*, *ExtendFootprintTTL* operations to avoid (re)building the entire data structure from scratch.

By default, an empty instance will be created.

build()

Returns
a copy of the final data structure.

classmethod from_xdr(*soroban_data*)

Create a new *SorobanDataBuilder* object from an XDR object.

Parameters
soroban_data (`str` | *SorobanTransactionData*) – The XDR object that represents a SorobanTransactionData.

Return type
SorobanDataBuilder

Returns
This builder.

set_read_only(*read_only*)

Sets the read-only portion of the storage access footprint to be a certain set of ledger keys.

Parameters
read_only (`list`[*LedgerKey*]) – The read-only ledger keys to set.

Return type
SorobanDataBuilder

Returns
This builder.

set_read_write(*read_write*)

Sets the read-write portion of the storage access footprint to be a certain set of ledger keys.

Parameters
read_write (`list`[*LedgerKey*]) – The read-write ledger keys to set.

Return type
SorobanDataBuilder

Returns
This builder.

set_resource_fee(*fee*)

Sets the "resource" fee portion of the Soroban data.

Parameters
fee (`int`) – The resource fee to set (int64)

Return type
SorobanDataBuilder

Returns
This builder.

`set_resources`(*instructions*, *disk_read_bytes*, *write_bytes*)

Sets up the resource metrics.

You should almost NEVER need this, as its often generated / provided to you by transaction simulation/preflight from a Soroban RPC server.

Parameters

- **instructions** (`int`) – Number of CPU instructions (uint32)
- **disk_read_bytes** (`int`) – Number of bytes being read from disk (uint32)
- **write_bytes** (`int`) – Number of bytes being written to disk and memory (uint32)

Return type
SorobanDataBuilder

Returns
This builder.

2.1.30 SorobanServer

`class stellar_sdk.SorobanServer`(*server_url='https://soroban-testnet.stellar.org:443'*, *client=None*)

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

Parameters

- **server_url** (`str`) – Soroban RPC server URL. (ex. `https://soroban-testnet.stellar.org:443`)
- **client** (*BaseSyncClient* | `None`) – A client instance that will be used to make requests.

`close`()

Close underlying connector, and release all acquired resources.

Return type
`None`

`get_contract_data`(*contract_id*, *key*, *durability=Durability.PERSISTENT*)

Reads the current value of contract data ledger entries directly.

Parameters

- **contract_id** (`str`) – The contract ID containing the data to load. Encoded as Stellar Contract Address, for example: `"CCJZ5DGASBWQXR5MPFCJXMBI333XE5U3FSJTNQU7RIKE3P5GN2K2WYD5"`
- **key** (*SCVal*) – The key of the contract data to load.
- **durability** (`Durability`) – The "durability keyspace" that this ledger key belongs to, which is either `Durability.TEMPORARY` or `Durability.PERSISTENT`. Defaults to `Durability.PERSISTENT`.

Return type
LedgerEntryResult | `None`

Returns
A *LedgerEntryResult* object contains the ledger entry result or `None` if not found.

Raises
`SorobanRpcErrorResponse` - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_contract_info`(*contract_id*)

Fetch and parse a contract's metadata, spec, and environment metadata.

Parameters
contract_id (`str`) – The contract ID encoded as a Stellar contract address.

Return type
ContractInfo

Returns
The contract metadata, interface specification, and environment metadata.

`get_contract_meta`(*contract_id*)

Fetch and parse a contract's SEP-46 metadata.

Parameters
contract_id (`str`) – The contract ID encoded as a Stellar contract address.

Return type
ContractMeta

Returns
The contract metadata.

`get_contract_spec`(*contract_id*)

Fetch and parse a contract's SEP-48 interface specification.

Parameters
contract_id (`str`) – The contract ID encoded as a Stellar contract address.

Return type
ContractSpec

Returns
The contract interface specification.

`get_contract_wasm`(*contract_id*)

Fetch a contract's Wasm code.

Parameters
contract_id (`str`) – The contract ID encoded as a Stellar contract address.

Return type
`bytes`

Returns
The raw Wasm code.

Raises
`ContractInstanceNotFoundError` - If the contract instance ledger entry is not found.

Raises
`SACHasNoWasmError` - If the contract is a Stellar Asset Contract.

Raises
ContractCodeNotFoundError - If the contract code ledger entry is not found or archived.

Raises
ContractWasmRetrievalError - If the executable kind is not supported.

`get_contract_wasm_by_hash`(*wasm_hash*)

Fetch raw contract Wasm code by its 32-byte hash.

Parameters
wasm_hash (bytes) – The 32-byte Wasm hash.

Return type
bytes

Returns
The raw Wasm code.

Raises
- **ValueError** – If wasm_hash is not 32 bytes or is all zero bytes.
- **TypeError** – If wasm_hash is not bytes.

Raises
ContractCodeNotFoundError - If the contract code ledger entry is not found or archived.

Raises
ContractWasmRetrievalError - If the ledger entry response does not contain contract code.

`get_events`(*start_ledger=None*, *end_ledger=None*, *filters=None*, *cursor=None*, *limit=None*)

Fetch a list of events that occurred in the ledger range.

See Soroban RPC Documentation - getEvents

Parameters
- **start_ledger** (int | None) – Ledger sequence number to start fetching responses from (inclusive). This method will return an error if startLedger is less than the oldest ledger stored in this node, or greater than the latest ledger seen by this node. If a cursor is included in the request, startLedger must be omitted.
- **end_ledger** (int | None) – Ledger sequence number represents the end of search window (exclusive). If a cursor is included in the request, this must be omitted.
- **filters** (Sequence[*EventFilter*] | None) – A list of filters to apply to the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetEventsResponse

Returns
A *GetEventsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_fee_stats()`

General info about the fee stats.

See Soroban RPC Documentation - getFeeStats

Return type
GetFeeStatsResponse

Returns
A *GetFeeStatsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_health()`

General node health check.

See Soroban RPC Documentation - getHealth

Return type
GetHealthResponse

Returns
A *GetHealthResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_latest_ledger()`

Fetches the latest ledger meta info from network which Soroban-RPC is connected to.

See Soroban RPC Documentation - getLatestLedger

Return type
GetLatestLedgerResponse

Returns
A *GetLatestLedgerResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_ledger_entries(keys)`

For reading the current value of ledger entries directly.

Allows you to directly inspect the current state of a contract, a contract's code, or any other ledger entry. This is a backup way to access your contract data which may not be available via events or simulateTransaction.

See Soroban RPC Documentation - getLedgerEntries

Parameters
keys (list[*LedgerKey*]) – The ledger keys to fetch.

Return type
GetLedgerEntriesResponse

Returns
A *GetLedgerEntriesResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

get_ledgers(*start_ledger=None*, *cursor=None*, *limit=None*)

Fetch a detailed list of ledgers starting from the user specified starting point that you can paginate as long as the pages fall within the history retention of their corresponding RPC provider.

See Soroban RPC Documentation - getLedgers

Parameters

- **start_ledger** (int | None) – The first ledger to include in the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetLedgersResponse

Returns
A *GetLedgersResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

get_network()

General info about the currently configured network.

See Soroban RPC Documentation - getNetwork

Return type
GetNetworkResponse

Returns
A *GetNetworkResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_sac_balance`(*contract_id*, *sac*, *network_passphrase=None*)

Returns a contract's balance of a particular SAC asset, if any.

This is a convenience wrapper around *SorobanServer.get_ledger_entries()*.

Parameters
- **contract_id** (str) – The contract ID whose balance of *sac* you want to know.
- **sac** (*Asset*) – The build-in SAC token that you are querying from the given contract.
- **network_passphrase** (str | None) – The network passphrase to use for the contract ID. If not provided, it will use the network passphrase of the current network. We suggest you set it to enhance performance.

Return type
GetSACBalanceResponse

Returns
A *GetSACBalanceResponse* which will contain the balance entry details if and only if the request returned a valid balance ledger entry. If it doesn't, the *balance_entry* field will not exist.

`get_transaction`(*transaction_hash*)

Fetch the specified transaction.

See Soroban RPC Documentation - getTransaction

Parameters
transaction_hash (str) – The hash of the transaction to fetch.

Return type
GetTransactionResponse

Returns
A *GetTransactionResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_transactions`(*start_ledger=None*, *cursor=None*, *limit=None*)

Fetch a detailed list of transactions starting from the user specified starting point that you can paginate as long as the pages fall within the history retention of their corresponding RPC provider.

See Soroban RPC Documentation - getTransactions

Parameters
- **start_ledger** (int | None) – The first ledger to include in the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetTransactionsResponse

Returns
A *GetTransactionsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`get_version_info()`

Version information about the RPC and Captive core.

See Soroban RPC Documentation - getVersionInfo

Return type
GetVersionInfoResponse

Returns
A *GetVersionInfoResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`load_account(account_id)`

Load an account from the server, you can use the returned account object as source account for transactions.

Parameters
account_id (str) – The account ID.

Return type
Account

Returns
An *Account* object.

Raises
AccountNotFoundException - If the account is not found on the network.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`poll_transaction(transaction_hash, max_attempts=30, sleep_strategy=<function BasicSleepStrategy>)`

Poll for a particular transaction with certain parameters.

After submitting a transaction, clients can use this to poll for transaction completion and return a definitive state of success or failure.

Parameters

- **transaction_hash** (str) – The hash of the transaction to poll for.
- **max_attempts** (int) – The number of attempts to make before returning the last-seen status, defaults to 30.

- **sleep_strategy** (Callable[[int], int]) – The amount of time to wait for between each attempt, defaults to 1 second between each attempt.

Return type
GetTransactionResponse

Returns
A *GetTransactionResponse* response object after a "found" response, (which may be success or failure) or the last response obtained after polling the maximum number of specified attempts.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

prepare_transaction(*transaction_envelope*, *simulate_transaction_response=None*)

Submit a trial contract invocation, first run a simulation of the contract invocation as defined on the incoming transaction, and apply the results to a new copy of the transaction which is then returned. Setting the ledger footprint and authorization, so the resulting transaction is ready for signing and sending.

The returned transaction will also have an updated fee that is the sum of fee set on incoming transaction with the contract resource fees estimated from simulation. It is advisable to check the fee on returned transaction and validate or take appropriate measures for interaction with user to confirm it is acceptable.

You can call the *simulate_transaction()* method directly first if you want to inspect estimated fees for a given transaction in detail first if that is of importance.

Parameters

- **transaction_envelope** (*TransactionEnvelope*) – The transaction to prepare. It should include exactly one operation, which must be one of *RestoreFootprint*, *ExtendFootprintTTL*, or *InvokeHostFunction*. Any provided footprint will be ignored. You can use stellar_sdk.Transaction.is_soroban_transaction() to check if a transaction is a Soroban transaction. Any provided footprint will be overwritten. However, if your operation has existing auth entries, they will be preferred over ALL auth entries from the simulation. In other words, if you include auth entries, you don't care about the auth returned from the simulation. Other fields (footprint, etc.) will be filled as normal.
- **simulate_transaction_response** (*SimulateTransactionResponse* | None) – The response of the simulation of the transaction, typically you don't need to pass this parameter, it will be automatically called if you don't pass it.

Return type
TransactionEnvelope

Returns
A copy of the *TransactionEnvelope*, with the expected authorizations (in the case of invocation) and ledger footprint added. The transaction fee will also automatically be padded with the contract's minimum resource fees discovered from the simulation.

send_transaction(*transaction_envelope*)

Submit a real transaction to the Stellar network. This is the only way to make changes "on-chain".

See Soroban RPC Documentation - sendTransaction

Parameters
transaction_envelope (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – The transaction to send.

Return type
SendTransactionResponse

Returns
A *SendTransactionResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

simulate_transaction(*transaction_envelope*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

Submit a trial contract invocation to get back return values, expected ledger footprint, and expected costs.

See Soroban RPC Documentation - simulateTransaction

Parameters

- **transaction_envelope** (*TransactionEnvelope*) – The transaction to simulate. It should include exactly one operation, which must be one of RestoreFootprint, *InvokeHostFunction* or *ExtendFootprintTTL* operation. Any provided footprint will be ignored.
- **addl_resources** (ResourceLeeway | None) – Additional resource include in the simulation.
- **auth_mode** (*AuthMode* | None) – Explicitly allows users to opt-in to non-root authorization in recording mode.
- **auth_v2** (bool) – Request ADDRESS_V2 credentials instead of legacy ADDRESS credentials in returned auth entries. Set this to True when targeting Protocol 27 RPC behavior and callers need V2 auth entries. Older RPC servers may silently ignore this flag; inspect the returned credential arm to confirm the response type.

Return type
SimulateTransactionResponse

Returns
A *SimulateTransactionResponse* object contains the cost, footprint, result/auth requirements (if applicable), and error of the transaction.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

2.1.31 SorobanServerAsync

class stellar_sdk.**SorobanServerAsync**(*server_url='https://soroban-testnet.stellar.org:443'*, *client=None*)

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

Parameters

- **server_url** (str) – Soroban RPC server URL. (ex. `https://soroban-testnet.stellar.org:443`)
- **client** (*BaseAsyncClient* | None) – A client instance that will be used to make requests.

async close()

Close underlying connector, and release all acquired resources.

Return type
None

async get_contract_data(*contract_id*, *key*, *durability=Durability.PERSISTENT*)

Reads the current value of contract data ledger entries directly.

Parameters

- **contract_id** (str) – The contract ID containing the data to load. Encoded as Stellar Contract Address, for example: `"CCJZ5DGASBWQXR5MPFCJXMBI333XE5U3FSJTNQU7RIKE3P5GN2K2WYD5"`
- **key** (*SCVal*) – The key of the contract data to load.
- **durability** (Durability) – The "durability keyspace" that this ledger key belongs to, which is either `Durability.TEMPORARY` or `Durability.PERSISTENT`. Defaults to `Durability.PERSISTENT`.

Return type
LedgerEntryResult | None

Returns
A *LedgerEntryResult* object contains the ledger entry result or `None` if not found.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async get_contract_info(*contract_id*)

Fetch and parse a contract's metadata, spec, and environment metadata.

Parameters
contract_id (str) – The contract ID encoded as a Stellar contract address.

Return type
ContractInfo

Returns
The contract metadata, interface specification, and environment metadata.

async get_contract_meta(*contract_id*)

Fetch and parse a contract's SEP-46 metadata.

Parameters
contract_id (str) – The contract ID encoded as a Stellar contract address.

Return type
ContractMeta

Returns
The contract metadata.

async get_contract_spec(*contract_id*)

Fetch and parse a contract's SEP-48 interface specification.

Parameters
contract_id (str) – The contract ID encoded as a Stellar contract address.

Return type
ContractSpec

Returns
The contract interface specification.

async get_contract_wasm(*contract_id*)

Fetch a contract's Wasm code.

Parameters
contract_id (str) – The contract ID encoded as a Stellar contract address.

Return type
bytes

Returns
The raw Wasm code.

Raises
ContractInstanceNotFoundError - If the contract instance ledger entry is not found.

Raises
SACHasNoWasmError - If the contract is a Stellar Asset Contract.

Raises
ContractCodeNotFoundError - If the contract code ledger entry is not found or archived.

Raises
ContractWasmRetrievalError - If the executable kind is not supported.

async get_contract_wasm_by_hash(*wasm_hash*)

Fetch raw contract Wasm code by its 32-byte hash.

Parameters
wasm_hash (bytes) – The 32-byte Wasm hash.

Return type
bytes

Returns
The raw Wasm code.

Raises

- **ValueError** – If wasm_hash is not 32 bytes or is all zero bytes.
- **TypeError** – If wasm_hash is not bytes.

Raises
ContractCodeNotFoundError - If the contract code ledger entry is not found or archived.

Raises
ContractWasmRetrievalError - If the ledger entry response does not contain contract code.

async get_events(*start_ledger=None*, *end_ledger=None*, *filters=None*, *cursor=None*, *limit=None*)

Fetch a list of events that occurred in the ledger range.

See Soroban RPC Documentation - getEvents

Parameters

- **start_ledger** (int | None) – Ledger sequence number to start fetching responses from (inclusive). This method will return an error if startLedger is less than the oldest ledger stored in this node, or greater than the latest ledger seen by this node. If a cursor is included in the request, startLedger must be omitted.
- **end_ledger** (int | None) – Ledger sequence number represents the end of search window (exclusive). If a cursor is included in the request, this must be omitted.
- **filters** (Sequence[*EventFilter*] | None) – A list of filters to apply to the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetEventsResponse

Returns
A *GetEventsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async get_fee_stats()

General info about the fee stats.

See Soroban RPC Documentation - getFeeStats

Return type
GetFeeStatsResponse

Returns
A *GetFeeStatsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async get_health()

General node health check.

See Soroban RPC Documentation - getHealth

Return type
GetHealthResponse

Returns
A *GetHealthResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`async get_latest_ledger()`

Fetches the latest ledger meta info from network which Soroban-RPC is connected to.

See Soroban RPC Documentation - getLatestLedger

Return type
GetLatestLedgerResponse

Returns
A *GetLatestLedgerResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`async get_ledger_entries(keys)`

For reading the current value of ledger entries directly.

Allows you to directly inspect the current state of a contract, a contract's code, or any other ledger entry. This is a backup way to access your contract data which may not be available via events or simulateTransaction.

See Soroban RPC Documentation - getLedgerEntries

Parameters
keys (list[*LedgerKey*]) – The ledger keys to fetch.

Return type
GetLedgerEntriesResponse

Returns
A *GetLedgerEntriesResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`async get_ledgers(start_ledger=None, cursor=None, limit=None)`

Fetch a detailed list of ledgers starting from the user specified starting point that you can paginate as long as the pages fall within the history retention of their corresponding RPC provider.

See Soroban RPC Documentation - getLedgers

Parameters

- **start_ledger** (int | None) – The first ledger to include in the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetLedgersResponse

Returns
A *GetLedgersResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`async get_network()`

General info about the currently configured network.

See Soroban RPC Documentation - getNetwork

Return type
GetNetworkResponse

Returns
A *GetNetworkResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

`async get_sac_balance(contract_id, sac, network_passphrase=None)`

Returns a contract's balance of a particular SAC asset, if any.

This is a convenience wrapper around *SorobanServerAsync.get_ledger_entries()*.

Parameters

- **contract_id** (str) – The contract ID whose balance of *sac* you want to know.
- **sac** (*Asset*) – The build-in SAC token that you are querying from the given contract.
- **network_passphrase** (str | None) – The network passphrase to use for the contract ID. If not provided, it will use the network passphrase of the current network. We suggest you set it to enhance performance.

Return type
GetSACBalanceResponse

Returns
A *GetSACBalanceResponse* which will contain the balance entry details if and only if the request returned a valid balance ledger entry. If it doesn't, the *balance_entry* field will not exist.

`async get_transaction(transaction_hash)`

Fetch the specified transaction.

See Soroban RPC Documentation - getTransaction

Parameters
transaction_hash (str) – The hash of the transaction to fetch.

Return type
GetTransactionResponse

Returns
A *GetTransactionResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async get_transactions(*start_ledger=None*, *cursor=None*, *limit=None*)

Fetch a detailed list of transactions starting from the user specified starting point that you can paginate as long as the pages fall within the history retention of their corresponding RPC provider.

See Soroban RPC Documentation - getTransactions

Parameters
- **start_ledger** (int | None) – The first ledger to include in the results.
- **cursor** (str | None) – A cursor value for use in pagination.
- **limit** (int | None) – The maximum number of records to return.

Return type
GetTransactionsResponse

Returns
A *GetTransactionsResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async get_version_info()

Version information about the RPC and Captive core.

See Soroban RPC Documentation - getVersionInfo

Return type
GetVersionInfoResponse

Returns
A *GetVersionInfoResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async load_account(*account_id*)

Load an account from the server, you can use the returned account object as source account for transactions.

Parameters
account_id (str) – The account ID.

Return type
Account

Returns
An *Account* object.

Raises
AccountNotFoundException - If the account is not found on the network.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async poll_transaction(*transaction_hash*, *max_attempts=30*, *sleep_strategy=<function BasicSleepStrategy>*)

Poll for a particular transaction with certain parameters.

After submitting a transaction, clients can use this to poll for transaction completion and return a definitive state of success or failure.

Parameters

- **transaction_hash** (str) – The hash of the transaction to poll for.
- **max_attempts** (int) – The number of attempts to make before returning the last-seen status, defaults to 30.
- **sleep_strategy** (Callable[[int], int]) – The amount of time to wait for between each attempt, defaults to 1 second between each attempt.

Return type
GetTransactionResponse

Returns
A *GetTransactionResponse* response object after a "found" response, (which may be success or failure) or the last response obtained after polling the maximum number of specified attempts.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async prepare_transaction(*transaction_envelope*, *simulate_transaction_response=None*)

Submit a trial contract invocation, first run a simulation of the contract invocation as defined on the incoming transaction, and apply the results to a new copy of the transaction which is then returned. Setting the ledger footprint and authorization, so the resulting transaction is ready for signing and sending.

The returned transaction will also have an updated fee that is the sum of fee set on incoming transaction with the contract resource fees estimated from simulation. It is advisable to check the fee on returned transaction and validate or take appropriate measures for interaction with user to confirm it is acceptable.

You can call the *simulate_transaction()* method directly first if you want to inspect estimated fees for a given transaction in detail first if that is of importance.

Parameters

- **transaction_envelope** (*TransactionEnvelope*) – The transaction to prepare. It should include exactly one operation, which must be one of *RestoreFootprint*, *ExtendFootprintTTL*, or *InvokeHostFunction*. Any provided footprint will be

ignored. You can use `stellar_sdk.Transaction.is_soroban_transaction()` to check if a transaction is a Soroban transaction. Any provided footprint will be overwritten. However, if your operation has existing auth entries, they will be preferred over ALL auth entries from the simulation. In other words, if you include auth entries, you don't care about the auth returned from the simulation. Other fields (footprint, etc.) will be filled as normal.

- **simulate_transaction_response** (*SimulateTransactionResponse* | None) – The response of the simulation of the transaction, typically you don't need to pass this parameter, it will be automatically called if you don't pass it.

Return type
TransactionEnvelope

Returns
A copy of the *TransactionEnvelope*, with the expected authorizations (in the case of invocation) and ledger footprint added. The transaction fee will also automatically be padded with the contract's minimum resource fees discovered from the simulation.

async send_transaction(*transaction_envelope*)

Submit a real transaction to the Stellar network. This is the only way to make changes "on-chain".

See Soroban RPC Documentation - sendTransaction

Parameters
transaction_envelope (*TransactionEnvelope* | *FeeBumpTransactionEnvelope* | str) – The transaction to send.

Return type
SendTransactionResponse

Returns
A *SendTransactionResponse* object.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

async simulate_transaction(*transaction_envelope*, *addl_resources=None*, *auth_mode=None*, *auth_v2=False*)

Submit a trial contract invocation to get back return values, expected ledger footprint, and expected costs.

See Soroban RPC Documentation - simulateTransaction

Parameters

- **transaction_envelope** (*TransactionEnvelope*) – The transaction to simulate. It should include exactly one operation, which must be one of `RestoreFootprint`, *InvokeHostFunction* or *ExtendFootprintTTL* operation. Any provided footprint will be ignored.
- **addl_resources** (ResourceLeeway | None) – Additional resource include in the simulation.
- **auth_mode** (*AuthMode* | None) – Explicitly allows users to opt-in to non-root authorization in recording mode.
- **auth_v2** (bool) – Request `ADDRESS_V2` credentials instead of legacy `ADDRESS` credentials in returned auth entries. Set this to `True` when targeting Protocol 27 RPC

behavior and callers need V2 auth entries. Older RPC servers may silently ignore this flag; inspect the returned credential arm to confirm the response type.

Return type
SimulateTransactionResponse

Returns
A *SimulateTransactionResponse* object contains the cost, footprint, result/auth requirements (if applicable), and error of the transaction.

Raises
SorobanRpcErrorResponse - If the Soroban-RPC instance returns an error response.

Raises
BadResponseError - If a non-JSON error response is returned, possibly by a CDN or reverse proxy.

2.1.32 Soroban RPC Definitions

class stellar_sdk.soroban_rpc.**AuthMode**(**values*)

AuthMode represents the authentication mode for transaction simulation.

ENFORCE = 'enforce'

Always enforce mode, even with an empty list.

RECORD = 'record'

Always recording mode, failing if any auth exists.

RECORD_ALL_NOROOT = 'record_allow_nonroot'

Like *RECORD* but allowing non-root authorization.

stellar_sdk.soroban_rpc.**BasicSleepStrategy**(*_iteration*)

A strategy that will sleep 1 second each time.

Return type
int

class stellar_sdk.soroban_rpc.**Error**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**EventFilter**(***data*)

model_config: ClassVar[ConfigDict] = {'populate_by_name': True, 'validate_by_alias': True, 'validate_by_name': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**EventFilterType**(**values*)

class stellar_sdk.soroban_rpc.**EventInfo**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**Events**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**FeeDistribution**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetEventsRequest**(***data*)

Response for JSON-RPC method getEvents.

See getEvents documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetEventsResponse**(***data*)

Response for JSON-RPC method getEvents.

See getEvents documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetFeeStatsResponse**(***data*)

Response for JSON-RPC method getFeeStats.

See getFeeStats documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetHealthResponse**(***data*)

Response for JSON-RPC method getHealth.

See getHealth documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetLatestLedgerResponse**(***data*)

Response for JSON-RPC method getLatestLedger.

See getLatestLedger documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetLedgerEntriesRequest**(***data*)

Response for JSON-RPC method getLedgerEntries.

See getLedgerEntries documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetLedgerEntriesResponse**(***data*)

Response for JSON-RPC method getLedgerEntries.

See getLedgerEntries documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetLedgersRequest**(***data*)

Request for JSON-RPC method getLedgers.

See getLedgers documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetLedgersResponse**(***data*)

Response for JSON-RPC method getLedgers.

See getLedgers documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetNetworkResponse**(***data*)

Response for JSON-RPC method getNetwork.

See getNetwork documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetSACBalanceResponse**(***data*)

Response for *stellar_sdk.SorobanServer.get_sac_balance()* and *stellar_sdk.SorobanServerAsync.get_sac_balance()* methods.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetTransactionRequest**(***data*)

Response for JSON-RPC method getTransaction.

See getTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetTransactionResponse**(***data*)

Response for JSON-RPC method getTransaction.

See getTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetTransactionStatus**(**values*)

FAILED = 'FAILED'

TransactionStatusFailed indicates the transaction was included in the ledger and it was executed with an error.

NOT_FOUND = 'NOT_FOUND'

indicates the transaction was not found in Soroban-RPC's transaction store.

SUCCESS = 'SUCCESS'

indicates the transaction was included in the ledger and it was executed without errors.

class stellar_sdk.soroban_rpc.**GetTransactionsRequest**(***data*)

Request for JSON-RPC method getTransactions.

See getTransactions documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetTransactionsResponse**(***data*)

Response for JSON-RPC method getTransactions.

See getTransactions documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**GetVersionInfoResponse**(***data*)

Response for JSON-RPC method getVersionInfo.

See getVersionInfo documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**LedgerEntryChange**(***data*)

LedgerEntryChange designates a change in a ledger entry. Before and After cannot be omitted at the same time. If Before is omitted, it constitutes a creation, if After is omitted, it constitutes a deletion.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**LedgerEntryResult**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**LedgerInfo**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

stellar_sdk.soroban_rpc.**LinearSleepStrategy**(*iteration*)

A strategy that will sleep 1 second longer on each attempt.

Return type
int

class stellar_sdk.soroban_rpc.**PaginationOptions**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**Request**(***data*)

Represent the request sent to Soroban-RPC.

See JSON-RPC 2.0 Specification - Request object for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**ResourceConfig**(***data*)

ResourceConfig represents the additional resource leeways for transaction simulation.

model_config: ClassVar[ConfigDict] = {'populate_by_name': True, 'validate_by_alias': True, 'validate_by_name': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**Response**(***data*)

Represent the response returned from Soroban-RPC.

See JSON-RPC 2.0 Specification - Response object for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**RestorePreamble**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SACBalanceEntry**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SendTransactionRequest**(***data*)

Response for JSON-RPC method sendTransaction.

See sendTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SendTransactionResponse**(***data*)

Response for JSON-RPC method sendTransaction.

See sendTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SendTransactionStatus**(**values*)

DUPLICATE = 'DUPLICATE'

represents the status value returned by stellar-core when a submitted transaction is a duplicate

ERROR = 'ERROR'

represents the status value returned by stellar-core when an error occurred from submitting a transaction

PENDING = 'PENDING'

represents the status value returned by stellar-core when a transaction has been accepted for processing

TRY_AGAIN_LATER = 'TRY_AGAIN_LATER'

represents the status value returned by stellar-core when a submitted transaction was not included in the previous 4 ledgers and get banned for being added in the next few ledgers.

class stellar_sdk.soroban_rpc.**SimulateHostFunctionResult**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SimulateTransactionCost**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SimulateTransactionRequest**(***data*)

Response for JSON-RPC method simulateTransaction.

> **Note**
>
> The simulation response will have different model representations with different members present or absent depending on type of response that it is conveying. For example, the simulation response for invoke host function, could be one of three types: error, success, or restore operation needed.

See simulateTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {'populate_by_name': True, 'validate_by_alias': True, 'validate_by_name': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SimulateTransactionResponse**(***data*)

Response for JSON-RPC method simulateTransaction.

See simulateTransaction documentation for more information.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**SimulateTransactionResult**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

stellar_sdk.soroban_rpc.**SleepStrategy**

A function for *stellar_sdk.SorobanServer.poll_transaction()* and *stellar_sdk.SorobanServerAsync.poll_transaction()* that returns the number of _seconds_ to sleep on a given *iteration*.

alias of Callable[[int], int]

class stellar_sdk.soroban_rpc.**Transaction**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

class stellar_sdk.soroban_rpc.**TransactionResponseError**(***data*)

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

2.1.33 scval

stellar_sdk.scval.**to_native**(*sc_val*)

Given a stellar_xdr.SCVal value, attempt to convert it to a native Python type.

Possible conversions include:

- SCV_VOID -> *None*
- SCV_I32, SCV_U32 -> *int*
- SCV_I64, SCV_U64, SCV_I128, SCV_U128, SCV_I256, SCV_U256 -> *int*
- SCV_TIMEPOINT, SCV_DURATION -> *int*
- SCV_VEC -> *list* of any of the above (via recursion)
- SCV_MAP -> *dict* with keys and values of any of the above (via recursion)
- SCV_BOOL -> *bool*
- SCV_BYTES -> *bytes*

- SCV_SYMBOL -> *str*
- SCV_STRING -> *str* if the underlying buffer can be decoded as UTF-8, *bytes* of the raw contents in any error case
- SCV_ADDRESS -> *stellar_sdk.address.Address*

If no viable conversion can be determined, this function returns the original `stellar_xdr.SCVal` object.

Parameters
sc_val (*SCVal* | bytes | str) – The `stellar_sdk.xdr.SCVal` XDR object to convert. It can also be an `stellar_sdk.xdr.SCVal` expressed in base64 or bytes.

Return type
bool | None | int | str | bytes | *Address* | *SCVal* | list[Any] | dict[Any, Any]

Returns
The native Python type.

stellar_sdk.scval.**to_address**(*data*)

Creates a new `stellar_sdk.xdr.SCVal` XDR object from an *stellar_sdk.address.Address* object.

Parameters
data (*Address* | str) – The *stellar_sdk.address.Address* object to convert.

Return type
SCVal

Returns
A new `stellar_sdk.xdr.SCVal` XDR object with type `stellar_sdk.xdr.SCValType.SCV_ADDRESS`.

stellar_sdk.scval.**from_address**(*sc_val*)

Creates an *stellar_sdk.address.Address* object from a `stellar_sdk.xdr.SCVal` XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The `stellar_sdk.xdr.SCVal` XDR object to convert. It can also be an `stellar_sdk.xdr.SCVal` expressed in base64 or bytes.

Return type
Address

Returns
An *stellar_sdk.address.Address* object.

Raises
ValueError if `sc_val` is not of type `stellar_sdk.xdr.SCValType.SCV_ADDRESS`.

stellar_sdk.scval.**to_bool**(*data*)

Creates a new `stellar_sdk.xdr.SCVal` XDR object from a bool value.

Parameters
data (bool) – The bool value to convert.

Return type
SCVal

Returns
A new `stellar_sdk.xdr.SCVal` XDR object with type `stellar_sdk.xdr.SCValType.SCV_BOOL`.

stellar_sdk.scval.**from_bool**(*sc_val*)

Creates a bool value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
bool

Returns
A bool value.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_BOOL.

stellar_sdk.scval.**to_bytes**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from a bytes value.

Parameters
data (bytes) – The bytes value to convert.

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_BYTES.

stellar_sdk.scval.**from_bytes**(*sc_val*)

Creates a bytes value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
bytes

Returns
A bytes value.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_BYTES.

stellar_sdk.scval.**to_duration**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The duration. (uint64)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_DURATION.

Raises
ValueError if value is out of uint64 range.

stellar_sdk.scval.**from_duration**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
The duration. (uint64)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_DURATION.

stellar_sdk.scval.**to_int32**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (int32)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_I32.

Raises
ValueError if value is out of int32 range.

stellar_sdk.scval.**from_int32**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (int32)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_I32.

stellar_sdk.scval.**to_int64**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (int64)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_I64.

Raises
ValueError if value is out of int64 range.

stellar_sdk.scval.**from_int64**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (int64)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_I64.

stellar_sdk.scval.**to_int128**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (int128)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_I128.

Raises
ValueError if value is out of int128 range.

stellar_sdk.scval.**from_int128**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (int128)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_I128.

stellar_sdk.scval.**to_int256**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (int256)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_I256.

Raises
ValueError if value is out of int256 range.

stellar_sdk.scval.**from_int256**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (int256)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_I256.

stellar_sdk.scval.**to_map**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from a dict value.

The entries are sorted by key following Soroban runtime ordering rules, as the network requires ScMap keys to be in ascending order.

Parameters
data (dict[*SCVal*, *SCVal*]) – The dict value to convert.

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_MAP.

stellar_sdk.scval.**from_map**(*sc_val*)

Creates a dict value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
dict[*SCVal*, *SCVal*]

Returns
The map value.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_MAP.

stellar_sdk.scval.**to_string**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from a string value.

Parameters
data (str | bytes) – The string value to convert.

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_STRING.

stellar_sdk.scval.**from_string**(*sc_val*)

Creates a string value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
bytes

Returns
A string value in bytes.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_STRING.

stellar_sdk.scval.**to_symbol**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from a symbol value.

Parameters
data (str) – The symbol value to convert.

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_SYMBOL.

stellar_sdk.scval.**from_symbol**(*sc_val*)

Creates a symbol value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
str

Returns
A symbol value.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_SYMBOL.

stellar_sdk.scval.**to_timepoint**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The time point. (uint64)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_TIME_POINT.

Raises
ValueError if value is out of uint64 range.

stellar_sdk.scval.**from_timepoint**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
The time point. (uint64)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_TIMEPOINT.

stellar_sdk.scval.**to_uint32**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (uint32)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_U32.

Raises
ValueError if value is out of uint32 range.

stellar_sdk.scval.**from_uint32**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (uint32)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_U32.

stellar_sdk.scval.**to_uint64**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (uint64)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_U64.

Raises
ValueError if value is out of uint64 range.

stellar_sdk.scval.**from_uint64**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (uint64)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_U64.

stellar_sdk.scval.**to_uint128**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (uint128)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_U128.

Raises
ValueError if value is out of uint128 range.

stellar_sdk.scval.**from_uint128**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
An int value. (uint128)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_U128.

stellar_sdk.scval.**to_uint256**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from an int value.

Parameters
data (int) – The int to convert. (uint256)

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_U256.

Raises
ValueError if value is out of uint256 range.

stellar_sdk.scval.**from_uint256**(*sc_val*)

Creates an int value from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
int

Returns
The value. (uint256)

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_U256.

stellar_sdk.scval.**to_vec**(*data*)

Creates a new stellar_sdk.xdr.SCVal XDR object from a list of stellar_sdk.xdr.SCVal XDR objects.

Parameters
data (Sequence[*SCVal*]) – The list of stellar_sdk.xdr.SCVal XDR objects.

Return type
SCVal

Returns
A new stellar_sdk.xdr.SCVal XDR object with type stellar_sdk.xdr.SCValType.SCV_VEC.

stellar_sdk.scval.**from_vec**(*sc_val*)

Creates a list of stellar_sdk.xdr.SCVal XDR objects from a stellar_sdk.xdr.SCVal XDR object.

Parameters
sc_val (*SCVal* | bytes | str) – The stellar_sdk.xdr.SCVal XDR object to convert. It can also be an stellar_sdk.xdr.SCVal expressed in base64 or bytes.

Return type
list[*SCVal*]

Returns
The list of stellar_sdk.xdr.SCVal XDR objects.

Raises
ValueError if sc_val is not of type stellar_sdk.xdr.SCValType.SCV_VEC.

stellar_sdk.scval.**to_enum**(*key*, *data*)

Creates a stellar_sdk.xdr.SCVal XDR object corresponding to the Enum in the Rust SDK.

Warning

Please note that this API is experimental and may be removed at any time. I recommend using the *from_vec()* to implement it.

Parameters
- **key** (str) – The key of the Enum.
- **data** (*SCVal* | list[*SCVal*] | None) – The data of the Enum.

Return type
SCVal

Returns
A new `stellar_sdk.xdr.SCVal` XDR object.

`stellar_sdk.scval.`**from_enum**(*sc_val*)

Creates a tuple corresponding to the Enum in the Rust SDK.

> **Warning**
>
> Please note that this API is experimental and may be removed at any time. I recommend using the *from_vec()* and *from_symbol()* to implement it.

Parameters
sc_val (*SCVal* | `bytes` | `str`) – The `stellar_sdk.xdr.SCVal` XDR object to convert. It can also be an `stellar_sdk.xdr.SCVal` expressed in base64 or bytes.

Return type
`tuple`[`str`, *SCVal* | `list`[*SCVal*] | `None`]

Returns
A tuple corresponding to the Enum in the Rust SDK.

`stellar_sdk.scval.`**to_tuple_struct**(*data*)

Creates a new `stellar_sdk.xdr.SCVal` XDR object corresponding to the Tuple Struct in the Rust SDK.

> **Warning**
>
> Please note that this API is experimental and may be removed at any time. I recommend using the *to_vec()* to implement it.

Parameters
data (`Sequence`[*SCVal*]) – The fields of the Tuple Struct.

Return type
SCVal

Returns
A new `stellar_sdk.xdr.SCVal` XDR object.

`stellar_sdk.scval.`**from_tuple_struct**(*sc_val*)

Creates a list corresponding to the Tuple Struct in the Rust SDK.

> **Warning**
>
> Please note that this API is experimental and may be removed at any time. I recommend using the *from_vec()* to implement it.

Parameters
sc_val (*SCVal* | `bytes` | `str`) – The `stellar_sdk.xdr.SCVal` XDR object to convert. It can also be an `stellar_sdk.xdr.SCVal` expressed in base64 or bytes.

Return type
list[*SCVal*]

Returns
A list corresponding to the Tuple Struct in the Rust SDK.

stellar_sdk.scval.**to_struct**(*data*)

Creates a new `stellar_sdk.xdr.SCVal` XDR object corresponding to the Struct in the Rust SDK.

> **Warning**
>
> Please note that this API is experimental and may be removed at any time. I recommend using the *to_map()* and *to_symbol()* to implement it.

Parameters
data (dict[str, *SCVal*]) – The dict value to convert.

Return type
SCVal

Returns
A new `stellar_sdk.xdr.SCVal` XDR object.

stellar_sdk.scval.**from_struct**(*sc_val*)

Creates a dict corresponding to the Struct in the Rust SDK.

> **Warning**
>
> Please note that this API is experimental and may be removed at any time. I recommend using the *from_map()* and *from_symbol()* to implement it.

Parameters
sc_val (*SCVal* | bytes | str) – The `stellar_sdk.xdr.SCVal` XDR object to convert. It can also be an `stellar_sdk.xdr.SCVal` expressed in base64 or bytes.

Return type
dict[str, *SCVal*]

Returns
A dict corresponding to the Struct in the Rust SDK.

2.1.34 Auth

stellar_sdk.auth.**authorize_entry**(*entry*, *signer*, *valid_until_ledger_sequence*, *network_passphrase*)

Sign an existing Soroban authorization entry, returning a signed copy.

"Fills out" the authorization with the credentials, expiration ledger, and a signature shaped for the account at the entry's address — be it the default Stellar Account (when `signer` is a `Keypair`) or any custom account contract (when `signer` is an *AuthorizationSigner* callable that returns the contract-defined signature `SCVal`).

Source-account credentials are returned unchanged.

Default account example:

```
signed = authorize_entry(entry, keypair, valid_until, passphrase)
```

Custom account example (BLS, WebAuthn, threshold, ...):

```
from stellar_sdk import scval, xdr
from stellar_sdk.auth import authorization_payload_hash, authorize_entry

def bls_signer(preimage: xdr.HashIDPreimage) -> xdr.SCVal:
    payload = authorization_payload_hash(preimage)
    return scval.to_bytes(my_bls_sign(payload))  # whatever shape the contract␣
↪expects

signed = authorize_entry(entry, bls_signer, valid_until, passphrase)
```

Parameters

- **entry** (*SorobanAuthorizationEntry* | str) – Unsigned Soroban authorization entry, either a stellar_xdr.SorobanAuthorizationEntry or its base64 XDR string.
- **signer** (*Keypair* | Callable[[*HashIDPreimage*], *SCVal*]) – Either a Keypair (uses the default Stellar Account signature shape) or an *AuthorizationSigner* callable. The signer must produce a signature accepted by the account at entry.credentials.address.
- **valid_until_ledger_sequence** (int) – Ledger sequence through which this authorization entry should remain valid (the entry is invalid starting at validUntil + 1).
- **network_passphrase** (str) – Network passphrase incorporated into the signature (see stellar_sdk.Network for options).

Return type

SorobanAuthorizationEntry

Returns

A signed Soroban authorization entry.

Raises

ValueError: if the entry's credential address is not a classic account (G...) or contract (C...) address.

stellar_sdk.auth.**authorize_invocation**(*signer*, *address*, *valid_until_ledger_sequence*, *invocation*, *network_passphrase*)

Build a fresh Soroban authorization entry from scratch and sign it.

Expresses authorization as a function of:

- a particular identity — a signing Keypair, an account contract, or any other custom signer
- approving the execution of an invocation tree (typically a simulation-acquired stellar_xdr.SorobanAuthorizedInvocation)
- on a particular network (uniquely identified by its passphrase, see stellar_sdk.Network)
- until a particular ledger sequence is reached

This is the "build" counterpart of *authorize_entry()*, which signs an existing entry "in place".

Parameters

- **signer** (*Keypair* | Callable[[*HashIDPreimage*], *SCVal*]) – Either a Keypair or an *AuthorizationSigner* callable. See *authorize_entry()* for details.
- **address** (*Address* | str | None) – The address being authorized. Must be a classic G... account address or a C... contract address, or an Address instance of one of those types. When signer is a Keypair, may be omitted (defaults to the keypair's public key); otherwise required.
- **valid_until_ledger_sequence** (int) – Ledger sequence through which this authorization entry should remain valid.
- **invocation** (*SorobanAuthorizedInvocation*) – Invocation tree being authorized (typically from transaction simulation).
- **network_passphrase** (str) – Network passphrase incorporated into the signature.

Return type
SorobanAuthorizationEntry

Returns
A signed Soroban authorization entry.

Raises
ValueError: if address is omitted with a non-Keypair signer, or if address is not a classic account (G...) or contract (C...) address.

stellar_sdk.auth.**authorization_payload_hash**(*preimage*)

Return the 32-byte payload that account contracts receive in __check_auth.

Use this inside a custom *AuthorizationSigner* to obtain the bytes the host hashes from the authorization preimage and asks the account contract to verify.

Parameters
preimage (*HashIDPreimage*) – The Soroban authorization preimage.

Return type
bytes

Returns
SHA-256 hash of the preimage XDR bytes.

stellar_sdk.auth.**build_authorization_preimage**(*entry*, *valid_until_ledger_sequence*, *network_passphrase*)

Build the signature preimage for a Soroban address authorization entry.

Parameters

- **entry** (*SorobanAuthorizationEntry*) – Soroban authorization entry to be authorized.
- **valid_until_ledger_sequence** (int) – Ledger sequence through which this authorization entry should remain valid.
- **network_passphrase** (str) – Network passphrase incorporated into the signature.

Return type
HashIDPreimage

Returns
A stellar_sdk.xdr.HashIDPreimage for the authorization.

Raises
ValueError: if entry does not use address credentials, or if the credential address is not a classic account (G...) or contract (C...) address.

stellar_sdk.auth.**AuthorizationSigner**

Type alias for a custom Soroban authorization signer.

Receives the authorization preimage and returns the signature `SCVal` accepted by the account contract at the entry's address. Use *authorization_payload_hash()* to obtain the same 32-byte payload that the account's `__check_auth` would receive.

alias of `Callable`[[*HashIDPreimage*], *SCVal*]

2.1.35 Helpers

stellar_sdk.helpers.**parse_transaction_envelope_from_xdr**(*xdr*, *network_passphrase*)

When you are not sure whether your XDR belongs to *TransactionEnvelope* or *FeeBumpTransactionEnvelope*, you can use this helper function.

An example:

```
from stellar_sdk import Network
from stellar_sdk.helpers import parse_transaction_envelope_from_xdr

xdr = "AAAAAgAAAADHJNEDn33/C1uDkDfzDfKVq/
↪4XE9IxDfGiLCfoV7riZQAAA+gCI4TVABpRPgAAAAAAAAAAAAAAAQAAAAAAAAADAAAAAUxpcmEAAAAAabIaDgm0ypyJpsVfEj
↪kEB2Z4UL20y536evnwmmSc4c2FnxlvUcPZl5jgWHcNwY8LTpFhdrUN9TZWciCRp/JCZYa0SJh8cYB"
te = parse_transaction_envelope_from_xdr(xdr, Network.PUBLIC_NETWORK_PASSPHRASE)
print(te)
```

Parameters

- **xdr** (`str`) – Transaction envelope XDR
- **network_passphrase** (`str`) – The network to connect to for verifying and retrieving additional attributes from. (ex. `"Public Global Stellar Network ; September 2015"`)

Raises

`ValueError` - XDR is neither *TransactionEnvelope* nor *FeeBumpTransactionEnvelope*

Return type

TransactionEnvelope | *FeeBumpTransactionEnvelope*

2.1.36 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 recommended that you **always** use HTTPS.
- **client** (*BaseSyncClient* | `None`) – Http Client used to send the request.

Return type
MutableMapping[str, Any]

Returns
The stellar.toml file as an object via `toml.loads()`.

Raises
StellarTomlNotFoundError: if the Stellar toml file could not be found.

Raises
`ContentSizeLimitExceededError`: if the response size exceeds the maximum allowed size.

async stellar_sdk.sep.stellar_toml.**fetch_stellar_toml_async**(*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 recommended that you **always** use HTTPS.
- **client** (*BaseAsyncClient* | None) – Http client used to send the request.

Return type
MutableMapping[str, Any]

Returns
The stellar.toml file as a dict object.

Raises
StellarTomlNotFoundError: if the Stellar toml file could not be found.

Raises
`ContentSizeLimitExceededError`: if the response size exceeds the maximum allowed size.

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** (*BaseSyncClient* | None) – Http Client used to send the request.
- **federation_url** (str | None) – 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 recommended that you **always** use HTTPS.

Return type
FederationRecord

Returns
Federation record.

async stellar_sdk.sep.federation.**resolve_stellar_address_async**(*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** (*BaseAsyncClient* | None) – Http Client used to send the request.
- **federation_url** (str | None) – 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 recommended that you **always** use HTTPS.

Return type
FederationRecord

Returns
Federation record.

async stellar_sdk.sep.federation.**resolve_account_id_async**(*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. "GBYNR2QJXLBCBTRN44MRORCMI4YO7FZPFBCNOKTOBCAAFC7KC3LNPRYS"
- **domain** (str | None) – Get *federation_url* from the domain, you don't need to set this value if *federation_url* is set.
- **federation_url** (str | None) – The federation server URL (ex. "https://stellar.org/federation").
- **client** (*BaseAsyncClient* | 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 recommended that you **always** use HTTPS.

Return type
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. "GBYNR2QJXLBCBTRN44MRORCMI4YO7FZPFBCNOKTOBCAAFC7KC3LNPRYS"
- **domain** (str | None) – Get *federation_url* from the domain, you don't need to set this value if *federation_url* is set.

- **federation_url** (str | None) – The federation server URL (ex. "https://stellar.org/federation").
- **client** (*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 recommended that you **always** use HTTPS.

Return type
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*)

Please use *stellar_sdk.keypair.Keypair.generate_mnemonic_phrase()* and *stellar_sdk.keypair.Keypair.from_mnemonic_phrase()*

static derive(*seed*, *index*)

Derive an ED25519 key from a BIP-39 seed.

Return type
bytes

generate(*strength=128*)

Create a new mnemonic using a random generated number as entropy.

As defined in BIP39, the entropy must be a multiple of 32 bits, and its size must be between 128 and 256 bits. Therefore the possible values for *strength* are 128, 160, 192, 224 and 256.

If not provided, the default entropy length will be set to 128 bits.

The return is a list of words that encodes the generated entropy.

Parameters
strength (int) – Number of bytes used as entropy

Returns
A randomly generated mnemonic

Return type
str

to_bip39_seed(*mnemonic*, *passphrase=''*)

Derive a BIP-39 key from a mnemonic.

Return type
bytes

to_seed(*mnemonic*, *passphrase=''*, *index=0*)

Derive an ED25519 key from a mnemonic.

Return type
bytes

class stellar_sdk.sep.mnemonic.**Language**(**values*)

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 0007: URI Scheme to facilitate delegated signing

class stellar_sdk.sep.stellar_uri.**PayStellarUri**(*destination*, *amount=None*, *asset=None*, *memo=None*, *callback=None*, *message=None*, *network_passphrase=None*, *origin_domain=None*, *signature=None*)

A request for a payment to be signed.

See SEP-0007

Parameters

- **destination** (str) – A valid account ID or payment address.
- **amount** (str | Decimal | None) – Amount that destination will receive.
- **asset** (*Asset* | None) – Asset destination will receive.
- **memo** (*Memo* | None) – A memo to attach to the transaction.
- **callback** (str | None) – The uri to post the transaction to after signing.
- **message** (str | None) – An message for displaying to the user.
- **network_passphrase** (str | None) – The passphrase of the target network.
- **origin_domain** (str | None) – A fully qualified domain name that specifies the originating domain of the URI request.
- **signature** (str | None) – A base64 encode signature of the hash of the URI request.

classmethod from_uri(*uri*)

Parse Stellar Pay URI and generate *PayStellarUri* object.

Parameters
uri (str) – Stellar Pay URI.

Return type
PayStellarUri

Returns
PayStellarUri object from uri.

sign(*signer*)

Sign the URI.

Parameters
signer (*Keypair* | str) – The account used to sign this transaction, it should be the secret key of *URI_REQUEST_SIGNING_KEY*.

Return type
None

to_uri()

Generate the request URI.

Return type
str

Returns
Stellar Pay URI.

class stellar_sdk.sep.stellar_uri.**TransactionStellarUri**(*transaction_envelope*, *replace=None*, *callback=None*, *pubkey=None*, *message=None*, *network_passphrase=None*, *origin_domain=None*, *signature=None*)

A request for a transaction to be signed.

See SEP-0007

Parameters

- **transaction_envelope** (*TransactionEnvelope* | *FeeBumpTransactionEnvelope*) – Transaction waiting to be signed.
- **replace** (list[*Replacement*] | None) – A value that identifies the fields to be replaced in the xdr using the Txrep (SEP-0011) representation.
- **callback** (str | None) – The uri to post the transaction to after signing.
- **pubkey** (str | None) – Specify which public key you want the URI handler to sign for.
- **message** (str | None) – An message for displaying to the user.
- **network_passphrase** (str | None) – The passphrase of the target network.
- **origin_domain** (str | None) – A fully qualified domain name that specifies the originating domain of the URI request.
- **signature** (str | None) – A base64 encode signature of the hash of the URI request.

classmethod **from_uri**(*uri*, *network_passphrase*)

Parse Stellar Transaction URI and generate *TransactionStellarUri* object.

Parameters

- **uri** (str) – Stellar Transaction URI.
- **network_passphrase** (str | None) – The network to connect to for verifying and retrieving xdr, If it is set to *None*, the *network_passphrase* in the uri will not be verified.

Return type
TransactionStellarUri

Returns
TransactionStellarUri object from uri.

sign(*signer*)

Sign the URI.

Parameters
signer (*Keypair* | str) – The account used to sign this transaction, it should be the secret key of *URI_REQUEST_SIGNING_KEY*.

Return type
None

to_uri()

Generate the request URI.

Return type
str

Returns
Stellar Transaction URI.

class stellar_sdk.sep.stellar_uri.**Replacement**(*txrep_tx_field_name*, *reference_identifier*, *hint*)

Used to represent a single replacement.

An example:

```
r1 = Replacement("sourceAccount", "X", "account on which to create the trustline")
r2 = Replacement("seqNum", "Y", "sequence for sourceAccount")
replacements = [r1, r2]
```

See SEP-0007

Parameters

- **txrep_tx_field_name** (str) – Txrep tx field name.
- **reference_identifier** (str) – Reference identifier.
- **hint** (str) – A brief and clear explanation of the context for the *reference_identifier*.

SEP 0010: Stellar Web Authentication

stellar_sdk.sep.stellar_web_authentication.**build_challenge_transaction**(*server_secret*, *client_account_id*, *home_domain*, *web_auth_domain*, *network_passphrase*, *timeout=900*, *client_domain=None*, *client_signing_key=None*, *memo=None*)

Returns a valid SEP0010 challenge transaction which you can use for Stellar Web Authentication.

Parameters

- **server_secret** (str) – secret key for server's stellar.toml *SIGNING_KEY*.
- **client_account_id** (str) – The stellar account (`G...`) or muxed account (`M...`) that the wallet wishes to authenticate with the server.
- **home_domain** (str) – The fully qualified domain name of the service requiring authentication (ex. `"example.com"`).

- **web_auth_domain** (str) – The fully qualified domain name of the service issuing the challenge.
- **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).
- **client_domain** (str | None) – The domain of the client application requesting authentication
- **client_signing_key** (str | None) – The stellar account listed as the SIGNING_KEY on the client domain's TOML file
- **memo** (int | None) – The ID memo to attach to the transaction. Not permitted if *client_account_id* is a muxed account

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*, *home_domains*, *web_auth_domain*, *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.
- **home_domains** (str | Iterable[str]) – The home domain that is expected to be included in the first Manage Data operation's string key. If a list is provided, one of the domain names in the array must match.
- **web_auth_domain** (str) – The home domain that is expected to be included as the value of the Manage Data operation with the 'web_auth_domain' key. If no such operation is included, this parameter is not used.
- **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
ChallengeTransaction

stellar_sdk.sep.stellar_web_authentication.**verify_challenge_transaction_threshold**(*challenge_transaction*, *server_account_id*, *home_domains*, *web_auth_domain*, *network_passphrase*, *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.
- **home_domains** (str | Iterable[str]) – The home domain that is expected to be included in the first Manage Data operation's string key. If a list is provided, one of the domain names in the array must match.
- **web_auth_domain** (str) – The home domain that is expected to be included as the value of the Manage Data operation with the 'web_auth_domain' key. If no such operation is included, this parameter is not used.
- **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** (Sequence[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_master_key**(*chal serv hom web net- work*

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.
- **home_domains** (str | Iterable[str]) – The home domain that is expected to be included in the first Manage Data operation's string key. If a list is provided, one of the domain names in the array must match.
- **web_auth_domain** (str) – The home domain that is expected to be included as the value of the Manage Data operation with the 'web_auth_domain' key. If no such operation is included, this parameter is not used.
- **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_transaction*, *server_account_id*, *home_domains*, *web_auth_domain*, *network_passphrase*, *signers*)

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.
- **home_domains** (str | Iterable[str]) – The home domain that is expected to be included in the first Manage Data operation's string key. If a list is provided, one of the domain names in the array must match.
- **web_auth_domain** (str) – The home domain that is expected to be included as the value of the Manage Data operation with the 'web_auth_domain' key, if present.
- **network_passphrase** (str) – The network to connect to for verifying and retrieving additional attributes from. (ex. "Public Global Stellar Network ; September 2015")
- **signers** (Sequence[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*, *home_domains*, *web_auth_domain*, *network_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.
- **home_domains** (str | Iterable[str]) – The home domain that is expected to be included in the first Manage Data operation's string key. If a list is provided, one of the domain names in the array must match.
- **web_auth_domain** (str) – The home domain that is expected to be included as the value of the Manage Data operation with the *web_auth_domain* key, if present.
- **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

class stellar_sdk.sep.stellar_web_authentication.**ChallengeTransaction**(*transaction*, *client_account_id*, *matched_home_domain*, *memo=None*)

Used to store the results produced by *stellar_sdk.sep.stellar_web_authentication.read_challenge_transaction()*.

Parameters

- **transaction** (*TransactionEnvelope*) – The TransactionEnvelope parsed from challenge xdr.
- **client_account_id** (str) – The stellar account that the wallet wishes to authenticate with the server.
- **matched_home_domain** (str) – The domain name that has been matched.

- **memo** (int | None) – The ID memo attached to the transaction

SEP 0011: Txrep: human-readable low-level representation of Stellar transactions

stellar_sdk.sep.txrep.**to_txrep**(*transaction_envelope*)

Generate a human-readable format for Stellar transactions.

MuxAccount is currently not supported.

Txrep is a human-readable representation of Stellar transactions that functions like an assembly language for XDR.

See SEP-0011

Parameters
transaction_envelope (*TransactionEnvelope* | *FeeBumpTransactionEnvelope*) – Transaction envelope object.

Return type
str

Returns
A human-readable format for Stellar transactions.

stellar_sdk.sep.txrep.**from_txrep**(*txrep*, *network_passphrase*)

Parse txrep and generate transaction envelope object.

MuxAccount is currently not supported.

Txrep is a human-readable representation of Stellar transactions that functions like an assembly language for XDR.

See SEP-0011

Parameters

- **txrep** (str) – a human-readable format for Stellar transactions.
- **network_passphrase** (str) – The network to connect, you do not need to set this value at this time, it is reserved for future use.

Return type
TransactionEnvelope | *FeeBumpTransactionEnvelope*

Returns
A human-readable format for Stellar transactions.

SEP 0035: Operation IDs

class stellar_sdk.sep.toid.**TOID**(*ledger_sequence*, *transaction_order*, *operation_order*)

TOID represents the total order of Ledgers, Transactions and Operations. This is an implementation of SEP-35: https://github.com/stellar/stellar-protocol/blob/master/ecosystem/sep-0035.md

Operations within the stellar network have a total order, expressed by three pieces of information: the ledger sequence the operation was validated in, the order which the operation's containing transaction was applied in that ledger, and the index of the operation within that parent transaction.

Parameters

- **ledger_sequence** (int) – The ledger sequence the operation was validated in.
- **transaction_order** (int) – The order that the transaction was applied within the ledger where it was validated. The application order value starts at 1. The maximum supported number of transactions per operation is 1,048,575.

- **operation_order** (int) – The index of the operation within that parent transaction. The operation index value starts at 1. The maximum supported number of operations per transaction is 4095.

classmethod after_ledger(*ledger_sequence*)

Creates a new toid that represents the ledger time **after** any contents (e.g. transactions, operations) that occur within the specified ledger.

Parameters
ledger_sequence (int) – The ledger sequence.

Return type
TOID

Returns
The TOID instance.

classmethod from_int64(*value*)

Converts a signed 64-bit integer to a TOID.

Parameters
value (int) – The signed 64-bit integer to convert.

Return type
TOID

increment_operation_order()

Increments the operation order by 1, rolling over to the next ledger if overflow occurs. This allows queries to easily advance a cursor to the next operation.

Return type
None

Returns
The current TOID instance.

static ledger_range_inclusive(*start*, *end*)

The inclusive range representation between two ledgers inclusive. The second value points at the end+1 ledger so when using this value make sure < order is used.

Parameters

- **start** (int) – The start ledger sequence.
- **end** (int) – The end ledger sequence.

Return type
tuple[int, int]

Returns
The inclusive range representation between two ledgers.

to_int64()

Converts the TOID to a signed 64-bit integer.

Return type
int

Returns
The signed 64-bit integer representation of the TOID.

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.**InvalidFederationAddress**

If the federation address is invalid, the exception will be thrown.

class stellar_sdk.sep.exceptions.**FederationServerNotFoundError**

If the federation address is invalid, the exception will be thrown.

class stellar_sdk.sep.exceptions.**BadFederationResponseError**(*response*)

If the federation server does not return a valid response, the exception will be thrown.

Parameters
response – client response

class stellar_sdk.sep.exceptions.**InvalidSep10ChallengeError**

If the SEP 0010 validation fails, the exception will be thrown.

class stellar_sdk.sep.exceptions.**AccountRequiresMemoError**(*message*, *account_id*, *operation_index*)

AccountRequiresMemoError is raised when a transaction is trying to submit an operation to an account which requires a memo.

This error contains two attributes to help you identify the account requiring the memo and the operation where the account is the destination.

See SEP-0029 for more information.

2.1.37 stellar_sdk.xdr

AccountEntry

class stellar_sdk.xdr.account_entry.**AccountEntry**(*account_id*, *balance*, *seq_num*, *num_sub_entries*, *inflation_dest*, *flags*, *home_domain*, *thresholds*, *signers*, *ext*)

XDR Source Code:

```
struct AccountEntry
{
    AccountID accountID;      // master public key for this account
    int64 balance;            // in stroops
    SequenceNumber seqNum;    // last sequence number used for this account
    uint32 numSubEntries;     // number of sub-entries this account has
                              // drives the reserve
    AccountID* inflationDest; // Account to vote for during inflation
    uint32 flags;             // see AccountFlags

    string32 homeDomain; // can be used for reverse federation and memo lookup

    // fields used for signatures
    // thresholds stores unsigned bytes: [weight of master|low|medium|high]
    Thresholds thresholds;

    Signer signers<MAX_SIGNERS>; // possible signers for this account

    // reserved for future use
```

(continues on next page)

(continued from previous page)

```
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        AccountEntryExtensionV1 v1;
    }
    ext;
};
```

AccountEntryExt

class stellar_sdk.xdr.account_entry_ext.**AccountEntryExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        AccountEntryExtensionV1 v1;
    }
```

AccountEntryExtensionV1

class stellar_sdk.xdr.account_entry_extension_v1.**AccountEntryExtensionV1**(*liabilities*, *ext*)

XDR Source Code:

```
struct AccountEntryExtensionV1
{
    Liabilities liabilities;

    union switch (int v)
    {
    case 0:
        void;
    case 2:
        AccountEntryExtensionV2 v2;
    }
    ext;
};
```

AccountEntryExtensionV1Ext

class stellar_sdk.xdr.account_entry_extension_v1_ext.**AccountEntryExtensionV1Ext**(*v*, *v2=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
```

(continues on next page)

(continued from previous page)

```
        void;
    case 2:
        AccountEntryExtensionV2 v2;
    }
```

AccountEntryExtensionV2

class stellar_sdk.xdr.account_entry_extension_v2.**AccountEntryExtensionV2**(*num_sponsored*, *num_sponsoring*, *signer_sponsoring_i_ds*, *ext*)

XDR Source Code:

```
struct AccountEntryExtensionV2
{
    uint32 numSponsored;
    uint32 numSponsoring;
    SponsorshipDescriptor signerSponsoringIDs<MAX_SIGNERS>;

    union switch (int v)
    {
    case 0:
        void;
    case 3:
        AccountEntryExtensionV3 v3;
    }
    ext;
};
```

AccountEntryExtensionV2Ext

class stellar_sdk.xdr.account_entry_extension_v2_ext.**AccountEntryExtensionV2Ext**(*v*, *v3=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 3:
        AccountEntryExtensionV3 v3;
    }
```

AccountEntryExtensionV3

class stellar_sdk.xdr.account_entry_extension_v3.**AccountEntryExtensionV3**(*ext*, *seq_ledger*, *seq_time*)

XDR Source Code:

```
struct AccountEntryExtensionV3
{
```

(continues on next page)

(continued from previous page)

```
    // We can use this to add more fields, or because it is first, to
    // change AccountEntryExtensionV3 into a union.
    ExtensionPoint ext;

    // Ledger number at which `seqNum` took on its present value.
    uint32 seqLedger;

    // Time at which `seqNum` took on its present value.
    TimePoint seqTime;
};
```

AccountFlags

class stellar_sdk.xdr.account_flags.**AccountFlags**(**values*)

XDR Source Code:

```
enum AccountFlags
{ // masks for each flag

    // Flags set on issuer accounts
    // TrustLines are created with authorized set to "false" requiring
    // the issuer to set it for each TrustLine
    AUTH_REQUIRED_FLAG = 0x1,
    // If set, the authorized flag in TrustLines can be cleared
    // otherwise, authorization cannot be revoked
    AUTH_REVOCABLE_FLAG = 0x2,
    // Once set, causes all AUTH_* flags to be read-only
    AUTH_IMMUTABLE_FLAG = 0x4,
    // Trustlines are created with clawback enabled set to "true",
    // and claimable balances created from those trustlines are created
    // with clawback enabled set to "true"
    AUTH_CLAWBACK_ENABLED_FLAG = 0x8
};
```

AccountID

class stellar_sdk.xdr.account_id.**AccountID**(*account_id*)

XDR Source Code:

```
typedef PublicKey AccountID;
```

AccountMergeResult

class stellar_sdk.xdr.account_merge_result.**AccountMergeResult**(*code*, *source_account_balance=None*)

XDR Source Code:

```
union AccountMergeResult switch (AccountMergeResultCode code)
{
case ACCOUNT_MERGE_SUCCESS:
    int64 sourceAccountBalance; // how much got transferred from source account
```

(continues on next page)

(continued from previous page)

```
case ACCOUNT_MERGE_MALFORMED:
case ACCOUNT_MERGE_NO_ACCOUNT:
case ACCOUNT_MERGE_IMMUTABLE_SET:
case ACCOUNT_MERGE_HAS_SUB_ENTRIES:
case ACCOUNT_MERGE_SEQNUM_TOO_FAR:
case ACCOUNT_MERGE_DEST_FULL:
case ACCOUNT_MERGE_IS_SPONSOR:
    void;
};
```

AccountMergeResultCode

class stellar_sdk.xdr.account_merge_result_code.**AccountMergeResultCode**(**values*)

XDR Source Code:

```
enum AccountMergeResultCode
{
    // codes considered as "success" for the operation
    ACCOUNT_MERGE_SUCCESS = 0,
    // codes considered as "failure" for the operation
    ACCOUNT_MERGE_MALFORMED = -1,       // can't merge onto itself
    ACCOUNT_MERGE_NO_ACCOUNT = -2,      // destination does not exist
    ACCOUNT_MERGE_IMMUTABLE_SET = -3,   // source account has AUTH_IMMUTABLE set
    ACCOUNT_MERGE_HAS_SUB_ENTRIES = -4, // account has trust lines/offers
    ACCOUNT_MERGE_SEQNUM_TOO_FAR = -5,  // sequence number is over max allowed
    ACCOUNT_MERGE_DEST_FULL = -6,       // can't add source balance to
                                        // destination balance
    ACCOUNT_MERGE_IS_SPONSOR = -7       // can't merge account that is a sponsor
};
```

AllowTrustOp

class stellar_sdk.xdr.allow_trust_op.**AllowTrustOp**(*trustor*, *asset*, *authorize*)

XDR Source Code:

```
struct AllowTrustOp
{
    AccountID trustor;
    AssetCode asset;

    // One of 0, AUTHORIZED_FLAG, or AUTHORIZED_TO_MAINTAIN_LIABILITIES_FLAG
    uint32 authorize;
};
```

AllowTrustResult

class stellar_sdk.xdr.allow_trust_result.**AllowTrustResult**(*code*)

XDR Source Code:

```
union AllowTrustResult switch (AllowTrustResultCode code)
{
```

(continues on next page)

(continued from previous page)

```
case ALLOW_TRUST_SUCCESS:
    void;
case ALLOW_TRUST_MALFORMED:
case ALLOW_TRUST_NO_TRUST_LINE:
case ALLOW_TRUST_TRUST_NOT_REQUIRED:
case ALLOW_TRUST_CANT_REVOKE:
case ALLOW_TRUST_SELF_NOT_ALLOWED:
case ALLOW_TRUST_LOW_RESERVE:
    void;
};
```

AllowTrustResultCode

class stellar_sdk.xdr.allow_trust_result_code.**AllowTrustResultCode**(**values*)

XDR Source Code:

```
enum AllowTrustResultCode
{
    // codes considered as "success" for the operation
    ALLOW_TRUST_SUCCESS = 0,
    // codes considered as "failure" for the operation
    ALLOW_TRUST_MALFORMED = -1,     // asset is not ASSET_TYPE_ALPHANUM
    ALLOW_TRUST_NO_TRUST_LINE = -2, // trustor does not have a trustline
                                    // source account does not require trust
    ALLOW_TRUST_TRUST_NOT_REQUIRED = -3,
    ALLOW_TRUST_CANT_REVOKE = -4,       // source account can't revoke trust,
    ALLOW_TRUST_SELF_NOT_ALLOWED = -5, // trusting self is not allowed
    ALLOW_TRUST_LOW_RESERVE = -6        // claimable balances can't be created
                                        // on revoke due to low reserves
};
```

AlphaNum12

class stellar_sdk.xdr.alpha_num12.**AlphaNum12**(*asset_code*, *issuer*)

XDR Source Code:

```
struct AlphaNum12
{
    AssetCode12 assetCode;
    AccountID issuer;
};
```

AlphaNum4

class stellar_sdk.xdr.alpha_num4.**AlphaNum4**(*asset_code*, *issuer*)

XDR Source Code:

```
struct AlphaNum4
{
    AssetCode4 assetCode;
    AccountID issuer;
};
```

Asset

class stellar_sdk.xdr.asset.**Asset**(*type*, *alpha_num4=None*, *alpha_num12=None*)

XDR Source Code:

```
union Asset switch (AssetType type)
{
case ASSET_TYPE_NATIVE: // Not credit
    void;

case ASSET_TYPE_CREDIT_ALPHANUM4:
    AlphaNum4 alphaNum4;

case ASSET_TYPE_CREDIT_ALPHANUM12:
    AlphaNum12 alphaNum12;

    // add other asset types here in the future
};
```

AssetCode

class stellar_sdk.xdr.asset_code.**AssetCode**(*type*, *asset_code4=None*, *asset_code12=None*)

XDR Source Code:

```
union AssetCode switch (AssetType type)
{
case ASSET_TYPE_CREDIT_ALPHANUM4:
    AssetCode4 assetCode4;

case ASSET_TYPE_CREDIT_ALPHANUM12:
    AssetCode12 assetCode12;

    // add other asset types here in the future
};
```

AssetCode12

class stellar_sdk.xdr.asset_code12.**AssetCode12**(*asset_code12*)

XDR Source Code:

```
typedef opaque AssetCode12[12];
```

AssetCode4

class stellar_sdk.xdr.asset_code4.**AssetCode4**(*asset_code4*)

XDR Source Code:

```
typedef opaque AssetCode4[4];
```

AssetType

class stellar_sdk.xdr.asset_type.**AssetType**(**values*)

XDR Source Code:

```
enum AssetType
{
    ASSET_TYPE_NATIVE = 0,
    ASSET_TYPE_CREDIT_ALPHANUM4 = 1,
    ASSET_TYPE_CREDIT_ALPHANUM12 = 2,
    ASSET_TYPE_POOL_SHARE = 3
};
```

Auth

class stellar_sdk.xdr.auth.**Auth**(*flags*)

XDR Source Code:

```
struct Auth
{
    int flags;
};
```

AuthCert

class stellar_sdk.xdr.auth_cert.**AuthCert**(*pubkey*, *expiration*, *sig*)

XDR Source Code:

```
struct AuthCert
{
    Curve25519Public pubkey;
    uint64 expiration;
    Signature sig;
};
```

AuthenticatedMessage

class stellar_sdk.xdr.authenticated_message.**AuthenticatedMessage**(*v*, *v0=None*)

XDR Source Code:

```
union AuthenticatedMessage switch (uint32 v)
{
case 0:
    struct
    {
        uint64 sequence;
        StellarMessage message;
        HmacSha256Mac mac;
    } v0;
};
```

AuthenticatedMessageV0

class stellar_sdk.xdr.authenticated_message_v0.**AuthenticatedMessageV0**(*sequence*, *message*, *mac*)

XDR Source Code:

```
struct
    {
        uint64 sequence;
        StellarMessage message;
        HmacSha256Mac mac;
    }
```

BeginSponsoringFutureReservesOp

class stellar_sdk.xdr.begin_sponsoring_future_reserves_op.**BeginSponsoringFutureReservesOp**(*sponsored_id*)

XDR Source Code:

```
struct BeginSponsoringFutureReservesOp
{
    AccountID sponsoredID;
};
```

BeginSponsoringFutureReservesResult

class stellar_sdk.xdr.begin_sponsoring_future_reserves_result.**BeginSponsoringFutureReservesResult**(*code*)

XDR Source Code:

```
union BeginSponsoringFutureReservesResult switch (
    BeginSponsoringFutureReservesResultCode code)
{
case BEGIN_SPONSORING_FUTURE_RESERVES_SUCCESS:
    void;
case BEGIN_SPONSORING_FUTURE_RESERVES_MALFORMED:
case BEGIN_SPONSORING_FUTURE_RESERVES_ALREADY_SPONSORED:
case BEGIN_SPONSORING_FUTURE_RESERVES_RECURSIVE:
    void;
};
```

BeginSponsoringFutureReservesResultCode

class stellar_sdk.xdr.begin_sponsoring_future_reserves_result_code.**BeginSponsoringFutureReservesResultC**

XDR Source Code:

```
enum BeginSponsoringFutureReservesResultCode
{
    // codes considered as "success" for the operation
    BEGIN_SPONSORING_FUTURE_RESERVES_SUCCESS = 0,

    // codes considered as "failure" for the operation
    BEGIN_SPONSORING_FUTURE_RESERVES_MALFORMED = -1,
    BEGIN_SPONSORING_FUTURE_RESERVES_ALREADY_SPONSORED = -2,
    BEGIN_SPONSORING_FUTURE_RESERVES_RECURSIVE = -3
};
```

BinaryFuseFilterType

class stellar_sdk.xdr.binary_fuse_filter_type.**BinaryFuseFilterType**(**values*)

XDR Source Code:

```
enum BinaryFuseFilterType
{
    BINARY_FUSE_FILTER_8_BIT = 0,
    BINARY_FUSE_FILTER_16_BIT = 1,
    BINARY_FUSE_FILTER_32_BIT = 2
};
```

Boolean

class stellar_sdk.xdr.base.**Boolean**(*value*)

BucketEntry

class stellar_sdk.xdr.bucket_entry.**BucketEntry**(*type*, *live_entry=None*, *dead_entry=None*, *meta_entry=None*)

XDR Source Code:

```
union BucketEntry switch (BucketEntryType type)
{
case LIVEENTRY:
case INITENTRY:
    LedgerEntry liveEntry;

case DEADENTRY:
    LedgerKey deadEntry;
case METAENTRY:
    BucketMetadata metaEntry;
};
```

BucketEntryType

class stellar_sdk.xdr.bucket_entry_type.**BucketEntryType**(**values*)

XDR Source Code:

```
enum BucketEntryType
{
    METAENTRY =
        -1, // At-and-after protocol 11: bucket metadata, should come first.
    LIVEENTRY = 0, // Before protocol 11: created-or-updated;
                   // At-and-after protocol 11: only updated.
    DEADENTRY = 1,
    INITENTRY = 2 // At-and-after protocol 11: only created.
};
```

BucketListType

class stellar_sdk.xdr.bucket_list_type.**BucketListType**(**values*)

XDR Source Code:

```
enum BucketListType
{
    LIVE = 0,
    HOT_ARCHIVE = 1
};
```

BucketMetadata

class stellar_sdk.xdr.bucket_metadata.**BucketMetadata**(*ledger_version*, *ext*)

XDR Source Code:

```
struct BucketMetadata
{
    // Indicates the protocol version used to create / merge this bucket.
    uint32 ledgerVersion;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        BucketListType bucketListType;
    }
    ext;
};
```

BucketMetadataExt

class stellar_sdk.xdr.bucket_metadata_ext.**BucketMetadataExt**(*v*, *bucket_list_type=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        BucketListType bucketListType;
    }
```

BumpSequenceOp

class stellar_sdk.xdr.bump_sequence_op.**BumpSequenceOp**(*bump_to*)

XDR Source Code:

```
struct BumpSequenceOp
{
    SequenceNumber bumpTo;
};
```

BumpSequenceResult

class stellar_sdk.xdr.bump_sequence_result.**BumpSequenceResult**(*code*)

XDR Source Code:

```
union BumpSequenceResult switch (BumpSequenceResultCode code)
{
case BUMP_SEQUENCE_SUCCESS:
    void;
case BUMP_SEQUENCE_BAD_SEQ:
    void;
};
```

BumpSequenceResultCode

class stellar_sdk.xdr.bump_sequence_result_code.**BumpSequenceResultCode**(**values*)

XDR Source Code:

```
enum BumpSequenceResultCode
{
    // codes considered as "success" for the operation
    BUMP_SEQUENCE_SUCCESS = 0,
    // codes considered as "failure" for the operation
    BUMP_SEQUENCE_BAD_SEQ = -1 // `bumpTo` is not within bounds
};
```

ChangeTrustAsset

class stellar_sdk.xdr.change_trust_asset.**ChangeTrustAsset**(*type*, *alpha_num4=None*, *alpha_num12=None*, *liquidity_pool=None*)

XDR Source Code:

```
union ChangeTrustAsset switch (AssetType type)
{
case ASSET_TYPE_NATIVE: // Not credit
    void;

case ASSET_TYPE_CREDIT_ALPHANUM4:
    AlphaNum4 alphaNum4;

case ASSET_TYPE_CREDIT_ALPHANUM12:
    AlphaNum12 alphaNum12;

case ASSET_TYPE_POOL_SHARE:
    LiquidityPoolParameters liquidityPool;

    // add other asset types here in the future
};
```

ChangeTrustOp

class stellar_sdk.xdr.change_trust_op.**ChangeTrustOp**(*line*, *limit*)

XDR Source Code:

```
struct ChangeTrustOp
{
    ChangeTrustAsset line;

    // if limit is set to 0, deletes the trust line
    int64 limit;
};
```

ChangeTrustResult

class stellar_sdk.xdr.change_trust_result.**ChangeTrustResult**(*code*)

XDR Source Code:

```
union ChangeTrustResult switch (ChangeTrustResultCode code)
{
case CHANGE_TRUST_SUCCESS:
    void;
case CHANGE_TRUST_MALFORMED:
case CHANGE_TRUST_NO_ISSUER:
case CHANGE_TRUST_INVALID_LIMIT:
case CHANGE_TRUST_LOW_RESERVE:
case CHANGE_TRUST_SELF_NOT_ALLOWED:
case CHANGE_TRUST_TRUST_LINE_MISSING:
case CHANGE_TRUST_CANNOT_DELETE:
case CHANGE_TRUST_NOT_AUTH_MAINTAIN_LIABILITIES:
    void;
};
```

ChangeTrustResultCode

class stellar_sdk.xdr.change_trust_result_code.**ChangeTrustResultCode**(**values*)

XDR Source Code:

```
enum ChangeTrustResultCode
{
    // codes considered as "success" for the operation
    CHANGE_TRUST_SUCCESS = 0,
    // codes considered as "failure" for the operation
    CHANGE_TRUST_MALFORMED = -1,     // bad input
    CHANGE_TRUST_NO_ISSUER = -2,     // could not find issuer
    CHANGE_TRUST_INVALID_LIMIT = -3, // cannot drop limit below balance
                                     // cannot create with a limit of 0
    CHANGE_TRUST_LOW_RESERVE =
        -4, // not enough funds to create a new trust line,
    CHANGE_TRUST_SELF_NOT_ALLOWED = -5,   // trusting self is not allowed
    CHANGE_TRUST_TRUST_LINE_MISSING = -6, // Asset trustline is missing for pool
    CHANGE_TRUST_CANNOT_DELETE =
        -7, // Asset trustline is still referenced in a pool
```

(continues on next page)

(continued from previous page)

```
    CHANGE_TRUST_NOT_AUTH_MAINTAIN_LIABILITIES =
        -8 // Asset trustline is deauthorized
};
```

ClaimAtom

class stellar_sdk.xdr.claim_atom.**ClaimAtom**(*type*, *v0=None*, *order_book=None*, *liquidity_pool=None*)

XDR Source Code:

```
union ClaimAtom switch (ClaimAtomType type)
{
case CLAIM_ATOM_TYPE_V0:
    ClaimOfferAtomV0 v0;
case CLAIM_ATOM_TYPE_ORDER_BOOK:
    ClaimOfferAtom orderBook;
case CLAIM_ATOM_TYPE_LIQUIDITY_POOL:
    ClaimLiquidityAtom liquidityPool;
};
```

ClaimAtomType

class stellar_sdk.xdr.claim_atom_type.**ClaimAtomType**(**values*)

XDR Source Code:

```
enum ClaimAtomType
{
    CLAIM_ATOM_TYPE_V0 = 0,
    CLAIM_ATOM_TYPE_ORDER_BOOK = 1,
    CLAIM_ATOM_TYPE_LIQUIDITY_POOL = 2
};
```

ClaimClaimableBalanceOp

class stellar_sdk.xdr.claim_claimable_balance_op.**ClaimClaimableBalanceOp**(*balance_id*)

XDR Source Code:

```
struct ClaimClaimableBalanceOp
{
    ClaimableBalanceID balanceID;
};
```

ClaimClaimableBalanceResult

class stellar_sdk.xdr.claim_claimable_balance_result.**ClaimClaimableBalanceResult**(*code*)

XDR Source Code:

```
union ClaimClaimableBalanceResult switch (ClaimClaimableBalanceResultCode code)
{
case CLAIM_CLAIMABLE_BALANCE_SUCCESS:
    void;
case CLAIM_CLAIMABLE_BALANCE_DOES_NOT_EXIST:
```

(continues on next page)

(continued from previous page)

```
case CLAIM_CLAIMABLE_BALANCE_CANNOT_CLAIM:
case CLAIM_CLAIMABLE_BALANCE_LINE_FULL:
case CLAIM_CLAIMABLE_BALANCE_NO_TRUST:
case CLAIM_CLAIMABLE_BALANCE_NOT_AUTHORIZED:
case CLAIM_CLAIMABLE_BALANCE_TRUSTLINE_FROZEN:
    void;
};
```

ClaimClaimableBalanceResultCode

class stellar_sdk.xdr.claim_claimable_balance_result_code.**ClaimClaimableBalanceResultCode**(**values*)

XDR Source Code:

```
enum ClaimClaimableBalanceResultCode
{
    CLAIM_CLAIMABLE_BALANCE_SUCCESS = 0,
    CLAIM_CLAIMABLE_BALANCE_DOES_NOT_EXIST = -1,
    CLAIM_CLAIMABLE_BALANCE_CANNOT_CLAIM = -2,
    CLAIM_CLAIMABLE_BALANCE_LINE_FULL = -3,
    CLAIM_CLAIMABLE_BALANCE_NO_TRUST = -4,
    CLAIM_CLAIMABLE_BALANCE_NOT_AUTHORIZED = -5,
    CLAIM_CLAIMABLE_BALANCE_TRUSTLINE_FROZEN = -6
};
```

ClaimLiquidityAtom

class stellar_sdk.xdr.claim_liquidity_atom.**ClaimLiquidityAtom**(*liquidity_pool_id*, *asset_sold*, *amount_sold*, *asset_bought*, *amount_bought*)

XDR Source Code:

```
struct ClaimLiquidityAtom
{
    PoolID liquidityPoolID;

    // amount and asset taken from the pool
    Asset assetSold;
    int64 amountSold;

    // amount and asset sent to the pool
    Asset assetBought;
    int64 amountBought;
};
```

ClaimOfferAtom

class stellar_sdk.xdr.claim_offer_atom.**ClaimOfferAtom**(*seller_id*, *offer_id*, *asset_sold*, *amount_sold*, *asset_bought*, *amount_bought*)

XDR Source Code:

```
struct ClaimOfferAtom
{
    // emitted to identify the offer
    AccountID sellerID; // Account that owns the offer
    int64 offerID;

    // amount and asset taken from the owner
    Asset assetSold;
    int64 amountSold;

    // amount and asset sent to the owner
    Asset assetBought;
    int64 amountBought;
};
```

ClaimOfferAtomV0

class stellar_sdk.xdr.claim_offer_atom_v0.**ClaimOfferAtomV0**(*seller_ed25519*, *offer_id*, *asset_sold*, *amount_sold*, *asset_bought*, *amount_bought*)

XDR Source Code:

```
struct ClaimOfferAtomV0
{
    // emitted to identify the offer
    uint256 sellerEd25519; // Account that owns the offer
    int64 offerID;

    // amount and asset taken from the owner
    Asset assetSold;
    int64 amountSold;

    // amount and asset sent to the owner
    Asset assetBought;
    int64 amountBought;
};
```

ClaimPredicate

class stellar_sdk.xdr.claim_predicate.**ClaimPredicate**(*type*, *and_predicates=None*, *or_predicates=None*, *not_predicate=None*, *abs_before=None*, *rel_before=None*)

XDR Source Code:

```
union ClaimPredicate switch (ClaimPredicateType type)
{
case CLAIM_PREDICATE_UNCONDITIONAL:
    void;
case CLAIM_PREDICATE_AND:
    ClaimPredicate andPredicates<2>;
case CLAIM_PREDICATE_OR:
    ClaimPredicate orPredicates<2>;
```

(continues on next page)

(continued from previous page)

```
case CLAIM_PREDICATE_NOT:
    ClaimPredicate* notPredicate;
case CLAIM_PREDICATE_BEFORE_ABSOLUTE_TIME:
    int64 absBefore; // Predicate will be true if closeTime < absBefore
case CLAIM_PREDICATE_BEFORE_RELATIVE_TIME:
    int64 relBefore; // Seconds since closeTime of the ledger in which the
                     // ClaimableBalanceEntry was created
};
```

ClaimPredicateType

class stellar_sdk.xdr.claim_predicate_type.**ClaimPredicateType**(**values*)

XDR Source Code:

```
enum ClaimPredicateType
{
    CLAIM_PREDICATE_UNCONDITIONAL = 0,
    CLAIM_PREDICATE_AND = 1,
    CLAIM_PREDICATE_OR = 2,
    CLAIM_PREDICATE_NOT = 3,
    CLAIM_PREDICATE_BEFORE_ABSOLUTE_TIME = 4,
    CLAIM_PREDICATE_BEFORE_RELATIVE_TIME = 5
};
```

ClaimableBalanceEntry

class stellar_sdk.xdr.claimable_balance_entry.**ClaimableBalanceEntry**(*balance_id*, *claimants*, *asset*, *amount*, *ext*)

XDR Source Code:

```
struct ClaimableBalanceEntry
{
    // Unique identifier for this ClaimableBalanceEntry
    ClaimableBalanceID balanceID;

    // List of claimants with associated predicate
    Claimant claimants<10>;

    // Any asset including native
    Asset asset;

    // Amount of asset
    int64 amount;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        ClaimableBalanceEntryExtensionV1 v1;
    }
```

(continues on next page)

(continued from previous page)

```
    ext;
};
```

ClaimableBalanceEntryExt

class stellar_sdk.xdr.claimable_balance_entry_ext.**ClaimableBalanceEntryExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        ClaimableBalanceEntryExtensionV1 v1;
    }
```

ClaimableBalanceEntryExtensionV1

class stellar_sdk.xdr.claimable_balance_entry_extension_v1.**ClaimableBalanceEntryExtensionV1**(*ext*, *flags*)

XDR Source Code:

```
struct ClaimableBalanceEntryExtensionV1
{
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;

    uint32 flags; // see ClaimableBalanceFlags
};
```

ClaimableBalanceEntryExtensionV1Ext

class stellar_sdk.xdr.claimable_balance_entry_extension_v1_ext.**ClaimableBalanceEntryExtensionV1Ext**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

ClaimableBalanceFlags

class stellar_sdk.xdr.claimable_balance_flags.**ClaimableBalanceFlags**(**values*)

XDR Source Code:

```
enum ClaimableBalanceFlags
{
    // If set, the issuer account of the asset held by the claimable balance may
    // clawback the claimable balance
    CLAIMABLE_BALANCE_CLAWBACK_ENABLED_FLAG = 0x1
};
```

ClaimableBalanceID

class stellar_sdk.xdr.claimable_balance_id.**ClaimableBalanceID**(*type*, *v0=None*)

XDR Source Code:

```
union ClaimableBalanceID switch (ClaimableBalanceIDType type)
{
case CLAIMABLE_BALANCE_ID_TYPE_V0:
    Hash v0;
};
```

ClaimableBalanceIDType

class stellar_sdk.xdr.claimable_balance_id_type.**ClaimableBalanceIDType**(**values*)

XDR Source Code:

```
enum ClaimableBalanceIDType
{
    CLAIMABLE_BALANCE_ID_TYPE_V0 = 0
};
```

Claimant

class stellar_sdk.xdr.claimant.**Claimant**(*type*, *v0=None*)

XDR Source Code:

```
union Claimant switch (ClaimantType type)
{
case CLAIMANT_TYPE_V0:
    struct
    {
        AccountID destination;    // The account that can use this condition
        ClaimPredicate predicate; // Claimable if predicate is true
    } v0;
};
```

ClaimantType

class stellar_sdk.xdr.claimant_type.**ClaimantType**(**values*)

XDR Source Code:

```
enum ClaimantType
{
    CLAIMANT_TYPE_V0 = 0
};
```

ClaimantV0

class stellar_sdk.xdr.claimant_v0.**ClaimantV0**(*destination*, *predicate*)

XDR Source Code:

```
struct
    {
        AccountID destination;    // The account that can use this condition
        ClaimPredicate predicate; // Claimable if predicate is true
    }
```

ClawbackClaimableBalanceOp

class stellar_sdk.xdr.clawback_claimable_balance_op.**ClawbackClaimableBalanceOp**(*balance_id*)

XDR Source Code:

```
struct ClawbackClaimableBalanceOp
{
    ClaimableBalanceID balanceID;
};
```

ClawbackClaimableBalanceResult

class stellar_sdk.xdr.clawback_claimable_balance_result.**ClawbackClaimableBalanceResult**(*code*)

XDR Source Code:

```
union ClawbackClaimableBalanceResult switch (
    ClawbackClaimableBalanceResultCode code)
{
case CLAWBACK_CLAIMABLE_BALANCE_SUCCESS:
    void;
case CLAWBACK_CLAIMABLE_BALANCE_DOES_NOT_EXIST:
case CLAWBACK_CLAIMABLE_BALANCE_NOT_ISSUER:
case CLAWBACK_CLAIMABLE_BALANCE_NOT_CLAWBACK_ENABLED:
    void;
};
```

ClawbackClaimableBalanceResultCode

class stellar_sdk.xdr.clawback_claimable_balance_result_code.**ClawbackClaimableBalanceResultCode**(**values*)

XDR Source Code:

```
enum ClawbackClaimableBalanceResultCode
{
    // codes considered as "success" for the operation
    CLAWBACK_CLAIMABLE_BALANCE_SUCCESS = 0,

    // codes considered as "failure" for the operation
    CLAWBACK_CLAIMABLE_BALANCE_DOES_NOT_EXIST = -1,
    CLAWBACK_CLAIMABLE_BALANCE_NOT_ISSUER = -2,
    CLAWBACK_CLAIMABLE_BALANCE_NOT_CLAWBACK_ENABLED = -3
};
```

ClawbackOp

class stellar_sdk.xdr.clawback_op.**ClawbackOp**(*asset*, *from_*, *amount*)

XDR Source Code:

```
struct ClawbackOp
{
    Asset asset;
    MuxedAccount from_;
    int64 amount;
};
```

ClawbackResult

class stellar_sdk.xdr.clawback_result.**ClawbackResult**(*code*)

XDR Source Code:

```
union ClawbackResult switch (ClawbackResultCode code)
{
case CLAWBACK_SUCCESS:
    void;
case CLAWBACK_MALFORMED:
case CLAWBACK_NOT_CLAWBACK_ENABLED:
case CLAWBACK_NO_TRUST:
case CLAWBACK_UNDERFUNDED:
    void;
};
```

ClawbackResultCode

class stellar_sdk.xdr.clawback_result_code.**ClawbackResultCode**(**values*)

XDR Source Code:

```
enum ClawbackResultCode
{
    // codes considered as "success" for the operation
    CLAWBACK_SUCCESS = 0,

    // codes considered as "failure" for the operation
    CLAWBACK_MALFORMED = -1,
    CLAWBACK_NOT_CLAWBACK_ENABLED = -2,
    CLAWBACK_NO_TRUST = -3,
    CLAWBACK_UNDERFUNDED = -4
};
```

ConfigSettingContractBandwidthV0

class stellar_sdk.xdr.config_setting_contract_bandwidth_v0.**ConfigSettingContractBandwidthV0**(*ledger_max_tx*
tx_max_size_b
fee_tx_size1_k

XDR Source Code:

```
struct ConfigSettingContractBandwidthV0
{
    // Maximum sum of all transaction sizes in the ledger in bytes
    uint32 ledgerMaxTxsSizeBytes;
    // Maximum size in bytes for a transaction
    uint32 txMaxSizeBytes;

    // Fee for 1 KB of transaction size
    int64 feeTxSize1KB;
};
```

ConfigSettingContractComputeV0

class stellar_sdk.xdr.config_setting_contract_compute_v0.**ConfigSettingContractComputeV0**(*ledger_max_instruc*
tx_max_instructions
fee_rate_per_instruc
tx_memory_limit)

XDR Source Code:

```
struct ConfigSettingContractComputeV0
{
    // Maximum instructions per ledger
    int64 ledgerMaxInstructions;
    // Maximum instructions per transaction
    int64 txMaxInstructions;
    // Cost of 10000 instructions
    int64 feeRatePerInstructionsIncrement;

    // Memory limit per transaction. Unlike instructions, there is no fee
    // for memory, just the limit.
    uint32 txMemoryLimit;
};
```

ConfigSettingContractEventsV0

class stellar_sdk.xdr.config_setting_contract_events_v0.**ConfigSettingContractEventsV0**(*tx_max_contract_even*
*fee_contract_events1_*

XDR Source Code:

```
struct ConfigSettingContractEventsV0
{
    // Maximum size of events that a contract call can emit.
    uint32 txMaxContractEventsSizeBytes;
    // Fee for generating 1KB of contract events.
    int64 feeContractEvents1KB;
};
```

ConfigSettingContractExecutionLanesV0

class stellar_sdk.xdr.config_setting_contract_execution_lanes_v0.**ConfigSettingContractExecutionLanesV0**(

XDR Source Code:

```
struct ConfigSettingContractExecutionLanesV0
{
    // maximum number of Soroban transactions per ledger
    uint32 ledgerMaxTxCount;
};
```

ConfigSettingContractHistoricalDataV0

class stellar_sdk.xdr.config_setting_contract_historical_data_v0.**ConfigSettingContractHistoricalDataV0**(

XDR Source Code:

```
struct ConfigSettingContractHistoricalDataV0
{
    int64 feeHistorical1KB; // Fee for storing 1KB in archives
};
```

ConfigSettingContractLedgerCostExtV0

class stellar_sdk.xdr.config_setting_contract_ledger_cost_ext_v0.**ConfigSettingContractLedgerCostExtV0**(*tx*
fe

XDR Source Code:

```
struct ConfigSettingContractLedgerCostExtV0
{
    // Maximum number of RO+RW entries in the transaction footprint.
    uint32 txMaxFootprintEntries;
    // Fee per 1 KB of data written to the ledger.
    // Unlike the rent fee, this is a flat fee that is charged for any ledger
    // write, independent of the type of the entry being written.
    int64 feeWrite1KB;
};
```

ConfigSettingContractLedgerCostV0

class stellar_sdk.xdr.config_setting_contract_ledger_cost_v0.**ConfigSettingContractLedgerCostV0**(*ledger_ma*
ledger_ma
ledger_ma
ledger_ma
tx_max_di
tx_max_di
tx_max_wr
tx_max_wr
fee_disk_re
*fee_write_*
fee_disk_re
soroban_st
*rent_fee1_*
*rent_fee1_*
soroban_st

XDR Source Code:

```
struct ConfigSettingContractLedgerCostV0
{
    // Maximum number of disk entry read operations per ledger
    uint32 ledgerMaxDiskReadEntries;
    // Maximum number of bytes of disk reads that can be performed per ledger
    uint32 ledgerMaxDiskReadBytes;
    // Maximum number of ledger entry write operations per ledger
    uint32 ledgerMaxWriteLedgerEntries;
    // Maximum number of bytes that can be written per ledger
    uint32 ledgerMaxWriteBytes;

    // Maximum number of disk entry read operations per transaction
    uint32 txMaxDiskReadEntries;
    // Maximum number of bytes of disk reads that can be performed per transaction
    uint32 txMaxDiskReadBytes;
    // Maximum number of ledger entry write operations per transaction
    uint32 txMaxWriteLedgerEntries;
    // Maximum number of bytes that can be written per transaction
    uint32 txMaxWriteBytes;

    int64 feeDiskReadLedgerEntry;  // Fee per disk ledger entry read
    int64 feeWriteLedgerEntry;     // Fee per ledger entry write

    int64 feeDiskRead1KB;          // Fee for reading 1KB disk

    // The following parameters determine the write fee per 1KB.
    // Rent fee grows linearly until soroban state reaches this size
    int64 sorobanStateTargetSizeBytes;
    // Fee per 1KB rent when the soroban state is empty
    int64 rentFee1KBSorobanStateSizeLow;
    // Fee per 1KB rent when the soroban state has reached␣
↪`sorobanStateTargetSizeBytes`
    int64 rentFee1KBSorobanStateSizeHigh;
    // Rent fee multiplier for any additional data past the first␣
↪`sorobanStateTargetSizeBytes`
    uint32 sorobanStateRentFeeGrowthFactor;
};
```

ConfigSettingContractParallelComputeV0

class stellar_sdk.xdr.config_setting_contract_parallel_compute_v0.**ConfigSettingContractParallelComputeV**

XDR Source Code:

```
struct ConfigSettingContractParallelComputeV0
{
    // Maximum number of clusters with dependent transactions allowed in a
    // stage of parallel tx set component.
    // This effectively sets the lower bound on the number of physical threads
    // necessary to effectively apply transaction sets in parallel.
    uint32 ledgerMaxDependentTxClusters;
};
```

ConfigSettingEntry

class stellar_sdk.xdr.config_setting_entry.**ConfigSettingEntry**(*config_setting_id*, *contract_max_size_bytes=None*, *contract_compute=None*, *contract_ledger_cost=None*, *contract_historical_data=None*, *contract_events=None*, *contract_bandwidth=None*, *contract_cost_params_cpu_insns=None*, *contract_cost_params_mem_bytes=None*, *contract_data_key_size_bytes=None*, *contract_data_entry_size_bytes=None*, *state_archival_settings=None*, *contract_execution_lanes=None*, *live_soroban_state_size_window=None*, *eviction_iterator=None*, *contract_parallel_compute=None*, *contract_ledger_cost_ext=None*, *contract_scp_timing=None*, *frozen_ledger_keys=None*, *frozen_ledger_keys_delta=None*, *freeze_bypass_txs=None*, *freeze_bypass_txs_delta=None*)

XDR Source Code:

```
union ConfigSettingEntry switch (ConfigSettingID configSettingID)
{
case CONFIG_SETTING_CONTRACT_MAX_SIZE_BYTES:
    uint32 contractMaxSizeBytes;
case CONFIG_SETTING_CONTRACT_COMPUTE_V0:
    ConfigSettingContractComputeV0 contractCompute;
case CONFIG_SETTING_CONTRACT_LEDGER_COST_V0:
    ConfigSettingContractLedgerCostV0 contractLedgerCost;
case CONFIG_SETTING_CONTRACT_HISTORICAL_DATA_V0:
    ConfigSettingContractHistoricalDataV0 contractHistoricalData;
case CONFIG_SETTING_CONTRACT_EVENTS_V0:
    ConfigSettingContractEventsV0 contractEvents;
case CONFIG_SETTING_CONTRACT_BANDWIDTH_V0:
    ConfigSettingContractBandwidthV0 contractBandwidth;
case CONFIG_SETTING_CONTRACT_COST_PARAMS_CPU_INSTRUCTIONS:
    ContractCostParams contractCostParamsCpuInsns;
case CONFIG_SETTING_CONTRACT_COST_PARAMS_MEMORY_BYTES:
    ContractCostParams contractCostParamsMemBytes;
case CONFIG_SETTING_CONTRACT_DATA_KEY_SIZE_BYTES:
    uint32 contractDataKeySizeBytes;
case CONFIG_SETTING_CONTRACT_DATA_ENTRY_SIZE_BYTES:
    uint32 contractDataEntrySizeBytes;
case CONFIG_SETTING_STATE_ARCHIVAL:
    StateArchivalSettings stateArchivalSettings;
case CONFIG_SETTING_CONTRACT_EXECUTION_LANES:
```

(continues on next page)

(continued from previous page)

```
    ConfigSettingContractExecutionLanesV0 contractExecutionLanes;
case CONFIG_SETTING_LIVE_SOROBAN_STATE_SIZE_WINDOW:
    uint64 liveSorobanStateSizeWindow<>;
case CONFIG_SETTING_EVICTION_ITERATOR:
    EvictionIterator evictionIterator;
case CONFIG_SETTING_CONTRACT_PARALLEL_COMPUTE_V0:
    ConfigSettingContractParallelComputeV0 contractParallelCompute;
case CONFIG_SETTING_CONTRACT_LEDGER_COST_EXT_V0:
    ConfigSettingContractLedgerCostExtV0 contractLedgerCostExt;
case CONFIG_SETTING_SCP_TIMING:
    ConfigSettingSCPTiming contractSCPTiming;
case CONFIG_SETTING_FROZEN_LEDGER_KEYS:
    FrozenLedgerKeys frozenLedgerKeys;
case CONFIG_SETTING_FROZEN_LEDGER_KEYS_DELTA:
    FrozenLedgerKeysDelta frozenLedgerKeysDelta;
case CONFIG_SETTING_FREEZE_BYPASS_TXS:
    FreezeBypassTxs freezeBypassTxs;
case CONFIG_SETTING_FREEZE_BYPASS_TXS_DELTA:
    FreezeBypassTxsDelta freezeBypassTxsDelta;
};
```

ConfigSettingID

class stellar_sdk.xdr.config_setting_id.**ConfigSettingID**(**values*)

XDR Source Code:

```
enum ConfigSettingID
{
    CONFIG_SETTING_CONTRACT_MAX_SIZE_BYTES = 0,
    CONFIG_SETTING_CONTRACT_COMPUTE_V0 = 1,
    CONFIG_SETTING_CONTRACT_LEDGER_COST_V0 = 2,
    CONFIG_SETTING_CONTRACT_HISTORICAL_DATA_V0 = 3,
    CONFIG_SETTING_CONTRACT_EVENTS_V0 = 4,
    CONFIG_SETTING_CONTRACT_BANDWIDTH_V0 = 5,
    CONFIG_SETTING_CONTRACT_COST_PARAMS_CPU_INSTRUCTIONS = 6,
    CONFIG_SETTING_CONTRACT_COST_PARAMS_MEMORY_BYTES = 7,
    CONFIG_SETTING_CONTRACT_DATA_KEY_SIZE_BYTES = 8,
    CONFIG_SETTING_CONTRACT_DATA_ENTRY_SIZE_BYTES = 9,
    CONFIG_SETTING_STATE_ARCHIVAL = 10,
    CONFIG_SETTING_CONTRACT_EXECUTION_LANES = 11,
    CONFIG_SETTING_LIVE_SOROBAN_STATE_SIZE_WINDOW = 12,
    CONFIG_SETTING_EVICTION_ITERATOR = 13,
    CONFIG_SETTING_CONTRACT_PARALLEL_COMPUTE_V0 = 14,
    CONFIG_SETTING_CONTRACT_LEDGER_COST_EXT_V0 = 15,
    CONFIG_SETTING_SCP_TIMING = 16,
    CONFIG_SETTING_FROZEN_LEDGER_KEYS = 17,
    CONFIG_SETTING_FROZEN_LEDGER_KEYS_DELTA = 18,
    CONFIG_SETTING_FREEZE_BYPASS_TXS = 19,
    CONFIG_SETTING_FREEZE_BYPASS_TXS_DELTA = 20
};
```

ConfigSettingSCPTiming

class stellar_sdk.xdr.config_setting_scp_timing.**ConfigSettingSCPTiming**(*ledger_target_close_time_milliseconds*, *nomination_timeout_initial_milliseconds*, *nomination_timeout_increment_milliseconds*, *ballot_timeout_initial_milliseconds*, *ballot_timeout_increment_milliseconds*)

XDR Source Code:

```
struct ConfigSettingSCPTiming {
    uint32 ledgerTargetCloseTimeMilliseconds;
    uint32 nominationTimeoutInitialMilliseconds;
    uint32 nominationTimeoutIncrementMilliseconds;
    uint32 ballotTimeoutInitialMilliseconds;
    uint32 ballotTimeoutIncrementMilliseconds;
};
```

ConfigUpgradeSet

class stellar_sdk.xdr.config_upgrade_set.**ConfigUpgradeSet**(*updated_entry*)

XDR Source Code:

```
struct ConfigUpgradeSet {
    ConfigSettingEntry updatedEntry<>;
};
```

ConfigUpgradeSetKey

class stellar_sdk.xdr.config_upgrade_set_key.**ConfigUpgradeSetKey**(*contract_id*, *content_hash*)

XDR Source Code:

```
struct ConfigUpgradeSetKey {
    ContractID contractID;
    Hash contentHash;
};
```

ContractCodeCostInputs

class stellar_sdk.xdr.contract_code_cost_inputs.**ContractCodeCostInputs**(*ext*, *n_instructions*, *n_functions*, *n_globals*, *n_table_entries*, *n_types*, *n_data_segments*, *n_elem_segments*, *n_imports*, *n_exports*, *n_data_segment_bytes*)

XDR Source Code:

```
struct ContractCodeCostInputs {
    ExtensionPoint ext;
    uint32 nInstructions;
    uint32 nFunctions;
    uint32 nGlobals;
    uint32 nTableEntries;
    uint32 nTypes;
    uint32 nDataSegments;
    uint32 nElemSegments;
    uint32 nImports;
    uint32 nExports;
    uint32 nDataSegmentBytes;
};
```

ContractCodeEntry

class stellar_sdk.xdr.contract_code_entry.**ContractCodeEntry**(*ext*, *hash*, *code*)

XDR Source Code:

```
struct ContractCodeEntry {
    union switch (int v)
    {
        case 0:
            void;
        case 1:
            struct
            {
                ExtensionPoint ext;
                ContractCodeCostInputs costInputs;
            } v1;
    } ext;

    Hash hash;
    opaque code<>;
};
```

ContractCodeEntryExt

class stellar_sdk.xdr.contract_code_entry_ext.**ContractCodeEntryExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
        case 0:
            void;
        case 1:
            struct
            {
                ExtensionPoint ext;
                ContractCodeCostInputs costInputs;
            } v1;
    }
```

ContractCodeEntryV1

class stellar_sdk.xdr.contract_code_entry_v1.**ContractCodeEntryV1**(*ext*, *cost_inputs*)

XDR Source Code:

```
struct
            {
                ExtensionPoint ext;
                ContractCodeCostInputs costInputs;
            }
```

ContractCostParamEntry

class stellar_sdk.xdr.contract_cost_param_entry.**ContractCostParamEntry**(*ext*, *const_term*, *linear_term*)

XDR Source Code:

```
struct ContractCostParamEntry {
    // use `ext` to add more terms (e.g. higher order polynomials) in the future
    ExtensionPoint ext;

    int64 constTerm;
    int64 linearTerm;
};
```

ContractCostParams

class stellar_sdk.xdr.contract_cost_params.**ContractCostParams**(*contract_cost_params*)

XDR Source Code:

```
typedef ContractCostParamEntry ContractCostParams<CONTRACT_COST_COUNT_LIMIT>;
```

ContractCostType

class stellar_sdk.xdr.contract_cost_type.**ContractCostType**(**values*)

XDR Source Code:

```
enum ContractCostType {
    // Cost of running 1 wasm instruction
    WasmInsnExec = 0,
    // Cost of allocating a slice of memory (in bytes)
    MemAlloc = 1,
    // Cost of copying a slice of bytes into a pre-allocated memory
    MemCpy = 2,
    // Cost of comparing two slices of memory
    MemCmp = 3,
    // Cost of a host function dispatch, not including the actual work done by
    // the function nor the cost of VM invocation machinary
    DispatchHostFunction = 4,
    // Cost of visiting a host object from the host object storage. Exists to
    // make sure some baseline cost coverage, i.e. repeatly visiting objects
    // by the guest will always incur some charges.
    VisitObject = 5,
```

(continues on next page)

(continued from previous page)

```
    // Cost of serializing an xdr object to bytes
    ValSer = 6,
    // Cost of deserializing an xdr object from bytes
    ValDeser = 7,
    // Cost of computing the sha256 hash from bytes
    ComputeSha256Hash = 8,
    // Cost of computing the ed25519 pubkey from bytes
    ComputeEd25519PubKey = 9,
    // Cost of verifying ed25519 signature of a payload.
    VerifyEd25519Sig = 10,
    // Cost of instantiation a VM from wasm bytes code.
    VmInstantiation = 11,
    // Cost of instantiation a VM from a cached state.
    VmCachedInstantiation = 12,
    // Cost of invoking a function on the VM. If the function is a host function,
    // additional cost will be covered by `DispatchHostFunction`.
    InvokeVmFunction = 13,
    // Cost of computing a keccak256 hash from bytes.
    ComputeKeccak256Hash = 14,
    // Cost of decoding an ECDSA signature computed from a 256-bit prime modulus
    // curve (e.g. secp256k1 and secp256r1)
    DecodeEcdsaCurve256Sig = 15,
    // Cost of recovering an ECDSA secp256k1 key from a signature.
    RecoverEcdsaSecp256k1Key = 16,
    // Cost of int256 addition (`+`) and subtraction (`-`) operations
    Int256AddSub = 17,
    // Cost of int256 multiplication (`*`) operation
    Int256Mul = 18,
    // Cost of int256 division (`/`) operation
    Int256Div = 19,
    // Cost of int256 power (`exp`) operation
    Int256Pow = 20,
    // Cost of int256 shift (`shl`, `shr`) operation
    Int256Shift = 21,
    // Cost of drawing random bytes using a ChaCha20 PRNG
    ChaCha20DrawBytes = 22,

    // Cost of parsing wasm bytes that only encode instructions.
    ParseWasmInstructions = 23,
    // Cost of parsing a known number of wasm functions.
    ParseWasmFunctions = 24,
    // Cost of parsing a known number of wasm globals.
    ParseWasmGlobals = 25,
    // Cost of parsing a known number of wasm table entries.
    ParseWasmTableEntries = 26,
    // Cost of parsing a known number of wasm types.
    ParseWasmTypes = 27,
    // Cost of parsing a known number of wasm data segments.
    ParseWasmDataSegments = 28,
    // Cost of parsing a known number of wasm element segments.
    ParseWasmElemSegments = 29,
    // Cost of parsing a known number of wasm imports.
```

(continues on next page)

(continued from previous page)

```
    ParseWasmImports = 30,
    // Cost of parsing a known number of wasm exports.
    ParseWasmExports = 31,
    // Cost of parsing a known number of data segment bytes.
    ParseWasmDataSegmentBytes = 32,

    // Cost of instantiating wasm bytes that only encode instructions.
    InstantiateWasmInstructions = 33,
    // Cost of instantiating a known number of wasm functions.
    InstantiateWasmFunctions = 34,
    // Cost of instantiating a known number of wasm globals.
    InstantiateWasmGlobals = 35,
    // Cost of instantiating a known number of wasm table entries.
    InstantiateWasmTableEntries = 36,
    // Cost of instantiating a known number of wasm types.
    InstantiateWasmTypes = 37,
    // Cost of instantiating a known number of wasm data segments.
    InstantiateWasmDataSegments = 38,
    // Cost of instantiating a known number of wasm element segments.
    InstantiateWasmElemSegments = 39,
    // Cost of instantiating a known number of wasm imports.
    InstantiateWasmImports = 40,
    // Cost of instantiating a known number of wasm exports.
    InstantiateWasmExports = 41,
    // Cost of instantiating a known number of data segment bytes.
    InstantiateWasmDataSegmentBytes = 42,

    // Cost of decoding a bytes array representing an uncompressed SEC-1 encoded
    // point on a 256-bit elliptic curve
    Sec1DecodePointUncompressed = 43,
    // Cost of verifying an ECDSA Secp256r1 signature
    VerifyEcdsaSecp256r1Sig = 44,

    // Cost of encoding a BLS12-381 Fp (base field element)
    Bls12381EncodeFp = 45,
    // Cost of decoding a BLS12-381 Fp (base field element)
    Bls12381DecodeFp = 46,
    // Cost of checking a G1 point lies on the curve
    Bls12381G1CheckPointOnCurve = 47,
    // Cost of checking a G1 point belongs to the correct subgroup
    Bls12381G1CheckPointInSubgroup = 48,
    // Cost of checking a G2 point lies on the curve
    Bls12381G2CheckPointOnCurve = 49,
    // Cost of checking a G2 point belongs to the correct subgroup
    Bls12381G2CheckPointInSubgroup = 50,
    // Cost of converting a BLS12-381 G1 point from projective to affine coordinates
    Bls12381G1ProjectiveToAffine = 51,
    // Cost of converting a BLS12-381 G2 point from projective to affine coordinates
    Bls12381G2ProjectiveToAffine = 52,
    // Cost of performing BLS12-381 G1 point addition
    Bls12381G1Add = 53,
    // Cost of performing BLS12-381 G1 scalar multiplication
```

(continues on next page)

(continued from previous page)

```
    Bls12381G1Mul = 54,
    // Cost of performing BLS12-381 G1 multi-scalar multiplication (MSM)
    Bls12381G1Msm = 55,
    // Cost of mapping a BLS12-381 Fp field element to a G1 point
    Bls12381MapFpToG1 = 56,
    // Cost of hashing to a BLS12-381 G1 point
    Bls12381HashToG1 = 57,
    // Cost of performing BLS12-381 G2 point addition
    Bls12381G2Add = 58,
    // Cost of performing BLS12-381 G2 scalar multiplication
    Bls12381G2Mul = 59,
    // Cost of performing BLS12-381 G2 multi-scalar multiplication (MSM)
    Bls12381G2Msm = 60,
    // Cost of mapping a BLS12-381 Fp2 field element to a G2 point
    Bls12381MapFp2ToG2 = 61,
    // Cost of hashing to a BLS12-381 G2 point
    Bls12381HashToG2 = 62,
    // Cost of performing BLS12-381 pairing operation
    Bls12381Pairing = 63,
    // Cost of converting a BLS12-381 scalar element from U256
    Bls12381FrFromU256 = 64,
    // Cost of converting a BLS12-381 scalar element to U256
    Bls12381FrToU256 = 65,
    // Cost of performing BLS12-381 scalar element addition/subtraction
    Bls12381FrAddSub = 66,
    // Cost of performing BLS12-381 scalar element multiplication
    Bls12381FrMul = 67,
    // Cost of performing BLS12-381 scalar element exponentiation
    Bls12381FrPow = 68,
    // Cost of performing BLS12-381 scalar element inversion
    Bls12381FrInv = 69,

    // Cost of encoding a BN254 Fp (base field element)
    Bn254EncodeFp = 70,
    // Cost of decoding a BN254 Fp (base field element)
    Bn254DecodeFp = 71,
    // Cost of checking a G1 point lies on the curve
    Bn254G1CheckPointOnCurve = 72,
    // Cost of checking a G2 point lies on the curve
    Bn254G2CheckPointOnCurve = 73,
    // Cost of checking a G2 point belongs to the correct subgroup
    Bn254G2CheckPointInSubgroup = 74,
    // Cost of converting a BN254 G1 point from projective to affine coordinates
    Bn254G1ProjectiveToAffine = 75,
    // Cost of performing BN254 G1 point addition
    Bn254G1Add = 76,
    // Cost of performing BN254 G1 scalar multiplication
    Bn254G1Mul = 77,
    // Cost of performing BN254 pairing operation
    Bn254Pairing = 78,
    // Cost of converting a BN254 scalar element from U256
    Bn254FrFromU256 = 79,
```

(continues on next page)

(continued from previous page)

```
    // Cost of converting a BN254 scalar element to U256
    Bn254FrToU256 = 80,
    // // Cost of performing BN254 scalar element addition/subtraction
    Bn254FrAddSub = 81,
    // Cost of performing BN254 scalar element multiplication
    Bn254FrMul = 82,
    // Cost of performing BN254 scalar element exponentiation
    Bn254FrPow = 83,
     // Cost of performing BN254 scalar element inversion
    Bn254FrInv = 84,
    // Cost of performing BN254 G1 multi-scalar multiplication (MSM)
    Bn254G1Msm = 85
};
```

ContractDataDurability

class stellar_sdk.xdr.contract_data_durability.**ContractDataDurability**(**values*)

XDR Source Code:

```
enum ContractDataDurability {
    TEMPORARY = 0,
    PERSISTENT = 1
};
```

ContractDataEntry

class stellar_sdk.xdr.contract_data_entry.**ContractDataEntry**(*ext*, *contract*, *key*, *durability*, *val*)

XDR Source Code:

```
struct ContractDataEntry {
    ExtensionPoint ext;

    SCAddress contract;
    SCVal key;
    ContractDataDurability durability;
    SCVal val;
};
```

ContractEvent

class stellar_sdk.xdr.contract_event.**ContractEvent**(*ext*, *contract_id*, *type*, *body*)

XDR Source Code:

```
struct ContractEvent
{
    // We can use this to add more fields, or because it
    // is first, to change ContractEvent into a union.
    ExtensionPoint ext;

    ContractID* contractID;
    ContractEventType type;
```

(continues on next page)

(continued from previous page)

```
    union switch (int v)
    {
    case 0:
        struct
        {
            SCVal topics<>;
            SCVal data;
        } v0;
    }
    body;
};
```

ContractEventBody

class stellar_sdk.xdr.contract_event_body.**ContractEventBody**(*v*, *v0=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        struct
        {
            SCVal topics<>;
            SCVal data;
        } v0;
    }
```

ContractEventType

class stellar_sdk.xdr.contract_event_type.**ContractEventType**(**values*)

XDR Source Code:

```
enum ContractEventType
{
    SYSTEM = 0,
    CONTRACT = 1,
    DIAGNOSTIC = 2
};
```

ContractEventV0

class stellar_sdk.xdr.contract_event_v0.**ContractEventV0**(*topics*, *data*)

XDR Source Code:

```
struct
        {
            SCVal topics<>;
            SCVal data;
        }
```

ContractExecutable

class stellar_sdk.xdr.contract_executable.**ContractExecutable**(*type*, *wasm_hash=None*)

XDR Source Code:

```
union ContractExecutable switch (ContractExecutableType type)
{
case CONTRACT_EXECUTABLE_WASM:
    Hash wasm_hash;
case CONTRACT_EXECUTABLE_STELLAR_ASSET:
    void;
};
```

ContractExecutableType

class stellar_sdk.xdr.contract_executable_type.**ContractExecutableType**(**values*)

XDR Source Code:

```
enum ContractExecutableType
{
    CONTRACT_EXECUTABLE_WASM = 0,
    CONTRACT_EXECUTABLE_STELLAR_ASSET = 1
};
```

ContractID

class stellar_sdk.xdr.contract_id.**ContractID**(*contract_id*)

XDR Source Code:

```
typedef Hash ContractID;
```

ContractIDPreimage

class stellar_sdk.xdr.contract_id_preimage.**ContractIDPreimage**(*type*, *from_address=None*, *from_asset=None*)

XDR Source Code:

```
union ContractIDPreimage switch (ContractIDPreimageType type)
{
case CONTRACT_ID_PREIMAGE_FROM_ADDRESS:
    struct
    {
        SCAddress address;
        uint256 salt;
    } fromAddress;
case CONTRACT_ID_PREIMAGE_FROM_ASSET:
    Asset fromAsset;
};
```

ContractIDPreimageFromAddress

class stellar_sdk.xdr.contract_id_preimage_from_address.**ContractIDPreimageFromAddress**(*address*, *salt*)

XDR Source Code:

```
struct
    {
        SCAddress address;
        uint256 salt;
    }
```

ContractIDPreimageType

class stellar_sdk.xdr.contract_id_preimage_type.**ContractIDPreimageType**(**values*)

XDR Source Code:

```
enum ContractIDPreimageType
{
    CONTRACT_ID_PREIMAGE_FROM_ADDRESS = 0,
    CONTRACT_ID_PREIMAGE_FROM_ASSET = 1
};
```

CreateAccountOp

class stellar_sdk.xdr.create_account_op.**CreateAccountOp**(*destination*, *starting_balance*)

XDR Source Code:

```
struct CreateAccountOp
{
    AccountID destination; // account to create
    int64 startingBalance; // amount they end up with
};
```

CreateAccountResult

class stellar_sdk.xdr.create_account_result.**CreateAccountResult**(*code*)

XDR Source Code:

```
union CreateAccountResult switch (CreateAccountResultCode code)
{
case CREATE_ACCOUNT_SUCCESS:
    void;
case CREATE_ACCOUNT_MALFORMED:
case CREATE_ACCOUNT_UNDERFUNDED:
case CREATE_ACCOUNT_LOW_RESERVE:
case CREATE_ACCOUNT_ALREADY_EXIST:
    void;
};
```

CreateAccountResultCode

class stellar_sdk.xdr.create_account_result_code.**CreateAccountResultCode**(**values*)

XDR Source Code:

```
enum CreateAccountResultCode
{
    // codes considered as "success" for the operation
    CREATE_ACCOUNT_SUCCESS = 0, // account was created

    // codes considered as "failure" for the operation
    CREATE_ACCOUNT_MALFORMED = -1,   // invalid destination
    CREATE_ACCOUNT_UNDERFUNDED = -2, // not enough funds in source account
    CREATE_ACCOUNT_LOW_RESERVE =
        -3, // would create an account below the min reserve
    CREATE_ACCOUNT_ALREADY_EXIST = -4 // account already exists
};
```

CreateClaimableBalanceOp

class stellar_sdk.xdr.create_claimable_balance_op.**CreateClaimableBalanceOp**(*asset*, *amount*, *claimants*)

XDR Source Code:

```
struct CreateClaimableBalanceOp
{
    Asset asset;
    int64 amount;
    Claimant claimants<10>;
};
```

CreateClaimableBalanceResult

class stellar_sdk.xdr.create_claimable_balance_result.**CreateClaimableBalanceResult**(*code*, *balance_id=None*)

XDR Source Code:

```
union CreateClaimableBalanceResult switch (
    CreateClaimableBalanceResultCode code)
{
case CREATE_CLAIMABLE_BALANCE_SUCCESS:
    ClaimableBalanceID balanceID;
case CREATE_CLAIMABLE_BALANCE_MALFORMED:
case CREATE_CLAIMABLE_BALANCE_LOW_RESERVE:
case CREATE_CLAIMABLE_BALANCE_NO_TRUST:
case CREATE_CLAIMABLE_BALANCE_NOT_AUTHORIZED:
case CREATE_CLAIMABLE_BALANCE_UNDERFUNDED:
    void;
};
```

CreateClaimableBalanceResultCode

class stellar_sdk.xdr.create_claimable_balance_result_code.**CreateClaimableBalanceResultCode**(**values*)

XDR Source Code:

```
enum CreateClaimableBalanceResultCode
{
    CREATE_CLAIMABLE_BALANCE_SUCCESS = 0,
    CREATE_CLAIMABLE_BALANCE_MALFORMED = -1,
    CREATE_CLAIMABLE_BALANCE_LOW_RESERVE = -2,
    CREATE_CLAIMABLE_BALANCE_NO_TRUST = -3,
    CREATE_CLAIMABLE_BALANCE_NOT_AUTHORIZED = -4,
    CREATE_CLAIMABLE_BALANCE_UNDERFUNDED = -5
};
```

CreateContractArgs

class stellar_sdk.xdr.create_contract_args.**CreateContractArgs**(*contract_id_preimage*, *executable*)

XDR Source Code:

```
struct CreateContractArgs
{
    ContractIDPreimage contractIDPreimage;
    ContractExecutable executable;
};
```

CreateContractArgsV2

class stellar_sdk.xdr.create_contract_args_v2.**CreateContractArgsV2**(*contract_id_preimage*, *executable*, *constructor_args*)

XDR Source Code:

```
struct CreateContractArgsV2
{
    ContractIDPreimage contractIDPreimage;
    ContractExecutable executable;
    // Arguments of the contract's constructor.
    SCVal constructorArgs<>;
};
```

CreatePassiveSellOfferOp

class stellar_sdk.xdr.create_passive_sell_offer_op.**CreatePassiveSellOfferOp**(*selling*, *buying*, *amount*, *price*)

XDR Source Code:

```
struct CreatePassiveSellOfferOp
{
    Asset selling; // A
    Asset buying;  // B
    int64 amount;  // amount taker gets
```

(continues on next page)

(continued from previous page)

```
    Price price;   // cost of A in terms of B
};
```

CryptoKeyType

class stellar_sdk.xdr.crypto_key_type.**CryptoKeyType**(**values*)

XDR Source Code:

```
enum CryptoKeyType
{
    KEY_TYPE_ED25519 = 0,
    KEY_TYPE_PRE_AUTH_TX = 1,
    KEY_TYPE_HASH_X = 2,
    KEY_TYPE_ED25519_SIGNED_PAYLOAD = 3,
    // MUXED enum values for supported type are derived from the enum values
    // above by ORing them with 0x100
    KEY_TYPE_MUXED_ED25519 = 0x100
};
```

Curve25519Public

class stellar_sdk.xdr.curve25519_public.**Curve25519Public**(*key*)

XDR Source Code:

```
struct Curve25519Public
{
    opaque key[32];
};
```

Curve25519Secret

class stellar_sdk.xdr.curve25519_secret.**Curve25519Secret**(*key*)

XDR Source Code:

```
struct Curve25519Secret
{
    opaque key[32];
};
```

DataEntry

class stellar_sdk.xdr.data_entry.**DataEntry**(*account_id*, *data_name*, *data_value*, *ext*)

XDR Source Code:

```
struct DataEntry
{
    AccountID accountID; // account this data belongs to
    string64 dataName;
    DataValue dataValue;

    // reserved for future use
```

(continues on next page)

(continued from previous page)

```
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

DataEntryExt

class stellar_sdk.xdr.data_entry_ext.**DataEntryExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

DataValue

class stellar_sdk.xdr.data_value.**DataValue**(*data_value*)

XDR Source Code:

```
typedef opaque DataValue<64>;
```

DecoratedSignature

class stellar_sdk.xdr.decorated_signature.**DecoratedSignature**(*hint*, *signature*)

XDR Source Code:

```
struct DecoratedSignature
{
    SignatureHint hint;  // last 4 bytes of the public key, used as a hint
    Signature signature; // actual signature
};
```

DependentTxCluster

class stellar_sdk.xdr.dependent_tx_cluster.**DependentTxCluster**(*dependent_tx_cluster*)

XDR Source Code:

```
typedef TransactionEnvelope DependentTxCluster<>;
```

DiagnosticEvent

class stellar_sdk.xdr.diagnostic_event.**DiagnosticEvent**(*in_successful_contract_call*, *event*)

XDR Source Code:

```
struct DiagnosticEvent
{
    bool inSuccessfulContractCall;
    ContractEvent event;
};
```

DontHave

class stellar_sdk.xdr.dont_have.**DontHave**(*type*, *req_hash*)

XDR Source Code:

```
struct DontHave
{
    MessageType type;
    uint256 reqHash;
};
```

Double

class stellar_sdk.xdr.base.**Double**(*value*)

Duration

class stellar_sdk.xdr.duration.**Duration**(*duration*)

XDR Source Code:

```
typedef uint64 Duration;
```

EncodedLedgerKey

class stellar_sdk.xdr.encoded_ledger_key.**EncodedLedgerKey**(*encoded_ledger_key*)

XDR Source Code:

```
typedef opaque EncodedLedgerKey<>;
```

EncryptedBody

class stellar_sdk.xdr.encrypted_body.**EncryptedBody**(*encrypted_body*)

XDR Source Code:

```
typedef opaque EncryptedBody<64000>;
```

EndSponsoringFutureReservesResult

class stellar_sdk.xdr.end_sponsoring_future_reserves_result.**EndSponsoringFutureReservesResult**(*code*)

XDR Source Code:

```
union EndSponsoringFutureReservesResult switch (
    EndSponsoringFutureReservesResultCode code)
{
case END_SPONSORING_FUTURE_RESERVES_SUCCESS:
    void;
```

(continues on next page)

(continued from previous page)

```
case END_SPONSORING_FUTURE_RESERVES_NOT_SPONSORED:
    void;
};
```

EndSponsoringFutureReservesResultCode

class stellar_sdk.xdr.end_sponsoring_future_reserves_result_code.**EndSponsoringFutureReservesResultCode**(

XDR Source Code:

```
enum EndSponsoringFutureReservesResultCode
{
    // codes considered as "success" for the operation
    END_SPONSORING_FUTURE_RESERVES_SUCCESS = 0,

    // codes considered as "failure" for the operation
    END_SPONSORING_FUTURE_RESERVES_NOT_SPONSORED = -1
};
```

EnvelopeType

class stellar_sdk.xdr.envelope_type.**EnvelopeType**(**values*)

XDR Source Code:

```
enum EnvelopeType
{
    ENVELOPE_TYPE_TX_V0 = 0,
    ENVELOPE_TYPE_SCP = 1,
    ENVELOPE_TYPE_TX = 2,
    ENVELOPE_TYPE_AUTH = 3,
    ENVELOPE_TYPE_SCPVALUE = 4,
    ENVELOPE_TYPE_TX_FEE_BUMP = 5,
    ENVELOPE_TYPE_OP_ID = 6,
    ENVELOPE_TYPE_POOL_REVOKE_OP_ID = 7,
    ENVELOPE_TYPE_CONTRACT_ID = 8,
    ENVELOPE_TYPE_SOROBAN_AUTHORIZATION = 9,
    ENVELOPE_TYPE_SOROBAN_AUTHORIZATION_WITH_ADDRESS = 10
};
```

Error

class stellar_sdk.xdr.error.**Error**(*code*, *msg*)

XDR Source Code:

```
struct Error
{
    ErrorCode code;
    string msg<100>;
};
```

ErrorCode

class stellar_sdk.xdr.error_code.**ErrorCode**(**values*)

XDR Source Code:

```
enum ErrorCode
{
    ERR_MISC = 0, // Unspecific error
    ERR_DATA = 1, // Malformed data
    ERR_CONF = 2, // Misconfiguration error
    ERR_AUTH = 3, // Authentication failure
    ERR_LOAD = 4  // System overloaded
};
```

EvictionIterator

class stellar_sdk.xdr.eviction_iterator.**EvictionIterator**(*bucket_list_level*, *is_curr_bucket*, *bucket_file_offset*)

XDR Source Code:

```
struct EvictionIterator {
    uint32 bucketListLevel;
    bool isCurrBucket;
    uint64 bucketFileOffset;
};
```

ExtendFootprintTTLOp

class stellar_sdk.xdr.extend_footprint_ttl_op.**ExtendFootprintTTLOp**(*ext*, *extend_to*)

XDR Source Code:

```
struct ExtendFootprintTTLOp
{
    ExtensionPoint ext;
    uint32 extendTo;
};
```

ExtendFootprintTTLResult

class stellar_sdk.xdr.extend_footprint_ttl_result.**ExtendFootprintTTLResult**(*code*)

XDR Source Code:

```
union ExtendFootprintTTLResult switch (ExtendFootprintTTLResultCode code)
{
case EXTEND_FOOTPRINT_TTL_SUCCESS:
    void;
case EXTEND_FOOTPRINT_TTL_MALFORMED:
case EXTEND_FOOTPRINT_TTL_RESOURCE_LIMIT_EXCEEDED:
case EXTEND_FOOTPRINT_TTL_INSUFFICIENT_REFUNDABLE_FEE:
    void;
};
```

ExtendFootprintTTLResultCode

class stellar_sdk.xdr.extend_footprint_ttl_result_code.**ExtendFootprintTTLResultCode**(**values*)

XDR Source Code:

```
enum ExtendFootprintTTLResultCode
{
    // codes considered as "success" for the operation
    EXTEND_FOOTPRINT_TTL_SUCCESS = 0,

    // codes considered as "failure" for the operation
    EXTEND_FOOTPRINT_TTL_MALFORMED = -1,
    EXTEND_FOOTPRINT_TTL_RESOURCE_LIMIT_EXCEEDED = -2,
    EXTEND_FOOTPRINT_TTL_INSUFFICIENT_REFUNDABLE_FEE = -3
};
```

ExtensionPoint

class stellar_sdk.xdr.extension_point.**ExtensionPoint**(*v*)

XDR Source Code:

```
union ExtensionPoint switch (int v)
{
case 0:
    void;
};
```

FeeBumpTransaction

class stellar_sdk.xdr.fee_bump_transaction.**FeeBumpTransaction**(*fee_source*, *fee*, *inner_tx*, *ext*)

XDR Source Code:

```
struct FeeBumpTransaction
{
    MuxedAccount feeSource;
    int64 fee;
    union switch (EnvelopeType type)
    {
    case ENVELOPE_TYPE_TX:
        TransactionV1Envelope v1;
    }
    innerTx;
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

FeeBumpTransactionEnvelope

class stellar_sdk.xdr.fee_bump_transaction_envelope.**FeeBumpTransactionEnvelope**(*tx*, *signatures*)

XDR Source Code:

```
struct FeeBumpTransactionEnvelope
{
    FeeBumpTransaction tx;
    /* Each decorated signature is a signature over the SHA256 hash of
     * a TransactionSignaturePayload */
    DecoratedSignature signatures<20>;
};
```

FeeBumpTransactionExt

class stellar_sdk.xdr.fee_bump_transaction_ext.**FeeBumpTransactionExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

FeeBumpTransactionInnerTx

class stellar_sdk.xdr.fee_bump_transaction_inner_tx.**FeeBumpTransactionInnerTx**(*type*, *v1=None*)

XDR Source Code:

```
union switch (EnvelopeType type)
    {
    case ENVELOPE_TYPE_TX:
        TransactionV1Envelope v1;
    }
```

Float

class stellar_sdk.xdr.base.**Float**(*value*)

FloodAdvert

class stellar_sdk.xdr.flood_advert.**FloodAdvert**(*tx_hashes*)

XDR Source Code:

```
struct FloodAdvert
{
    TxAdvertVector txHashes;
};
```

FloodDemand

class stellar_sdk.xdr.flood_demand.**FloodDemand**(*tx_hashes*)

XDR Source Code:

```
struct FloodDemand
{
    TxDemandVector txHashes;
};
```

FreezeBypassTxs

class stellar_sdk.xdr.freeze_bypass_txs.**FreezeBypassTxs**(*tx_hashes*)

XDR Source Code:

```
struct FreezeBypassTxs {
    Hash txHashes<>;
};
```

FreezeBypassTxsDelta

class stellar_sdk.xdr.freeze_bypass_txs_delta.**FreezeBypassTxsDelta**(*add_txs*, *remove_txs*)

XDR Source Code:

```
struct FreezeBypassTxsDelta {
    Hash addTxs<>;
    Hash removeTxs<>;
};
```

FrozenLedgerKeys

class stellar_sdk.xdr.frozen_ledger_keys.**FrozenLedgerKeys**(*keys*)

XDR Source Code:

```
struct FrozenLedgerKeys {
    EncodedLedgerKey keys<>;
};
```

FrozenLedgerKeysDelta

class stellar_sdk.xdr.frozen_ledger_keys_delta.**FrozenLedgerKeysDelta**(*keys_to_freeze*, *keys_to_unfreeze*)

XDR Source Code:

```
struct FrozenLedgerKeysDelta {
    EncodedLedgerKey keysToFreeze<>;
    EncodedLedgerKey keysToUnfreeze<>;
};
```

GeneralizedTransactionSet

class stellar_sdk.xdr.generalized_transaction_set.**GeneralizedTransactionSet**(*v*, *v1_tx_set=None*)

XDR Source Code:

```
union GeneralizedTransactionSet switch (int v)
{
// We consider the legacy TransactionSet to be v0.
case 1:
    TransactionSetV1 v1TxSet;
};
```

Hash

class stellar_sdk.xdr.hash.**Hash**(*hash*)

XDR Source Code:

```
typedef opaque Hash[32];
```

HashIDPreimage

class stellar_sdk.xdr.hash_id_preimage.**HashIDPreimage**(*type*, *operation_id=None*, *revoke_id=None*, *contract_id=None*, *soroban_authorization=None*, *soroban_authorization_with_address=None*)

XDR Source Code:

```
union HashIDPreimage switch (EnvelopeType type)
{
case ENVELOPE_TYPE_OP_ID:
    struct
    {
        AccountID sourceAccount;
        SequenceNumber seqNum;
        uint32 opNum;
    } operationID;
case ENVELOPE_TYPE_POOL_REVOKE_OP_ID:
    struct
    {
        AccountID sourceAccount;
        SequenceNumber seqNum;
        uint32 opNum;
        PoolID liquidityPoolID;
        Asset asset;
    } revokeID;
case ENVELOPE_TYPE_CONTRACT_ID:
    struct
    {
        Hash networkID;
        ContractIDPreimage contractIDPreimage;
    } contractID;
case ENVELOPE_TYPE_SOROBAN_AUTHORIZATION:
```

(continues on next page)

(continued from previous page)

```
    struct
    {
        Hash networkID;
        int64 nonce;
        uint32 signatureExpirationLedger;
        SorobanAuthorizedInvocation invocation;
    } sorobanAuthorization;
case ENVELOPE_TYPE_SOROBAN_AUTHORIZATION_WITH_ADDRESS:
    struct
    {
        Hash networkID;
        int64 nonce;
        uint32 signatureExpirationLedger;
        SCAddress address;
        SorobanAuthorizedInvocation invocation;
    } sorobanAuthorizationWithAddress;
};
```

HashIDPreimageContractID

class stellar_sdk.xdr.hash_id_preimage_contract_id.**HashIDPreimageContractID**(*network_id*, *contract_id_preimage*)

XDR Source Code:

```
struct
    {
        Hash networkID;
        ContractIDPreimage contractIDPreimage;
    }
```

HashIDPreimageOperationID

class stellar_sdk.xdr.hash_id_preimage_operation_id.**HashIDPreimageOperationID**(*source_account*, *seq_num*, *op_num*)

XDR Source Code:

```
struct
    {
        AccountID sourceAccount;
        SequenceNumber seqNum;
        uint32 opNum;
    }
```

HashIDPreimageRevokeID

class stellar_sdk.xdr.hash_id_preimage_revoke_id.**HashIDPreimageRevokeID**(*source_account*, *seq_num*, *op_num*, *liquidity_pool_id*, *asset*)

XDR Source Code:

```
struct
    {
        AccountID sourceAccount;
        SequenceNumber seqNum;
        uint32 opNum;
        PoolID liquidityPoolID;
        Asset asset;
    }
```

HashIDPreimageSorobanAuthorization

class stellar_sdk.xdr.hash_id_preimage_soroban_authorization.**HashIDPreimageSorobanAuthorization**(*network_i nonce, signature_expi invocation*)

XDR Source Code:

```
struct
    {
        Hash networkID;
        int64 nonce;
        uint32 signatureExpirationLedger;
        SorobanAuthorizedInvocation invocation;
    }
```

HashIDPreimageSorobanAuthorizationWithAddress

class stellar_sdk.xdr.hash_id_preimage_soroban_authorization_with_address.**HashIDPreimageSorobanAuthoriza**

XDR Source Code:

```
struct
    {
        Hash networkID;
        int64 nonce;
        uint32 signatureExpirationLedger;
        SCAddress address;
```

(continues on next page)

(continued from previous page)

```
        SorobanAuthorizedInvocation invocation;
    }
```

Hello

class stellar_sdk.xdr.hello.**Hello**(*ledger_version*, *overlay_version*, *overlay_min_version*, *network_id*, *version_str*, *listening_port*, *peer_id*, *cert*, *nonce*)

XDR Source Code:

```
struct Hello
{
    uint32 ledgerVersion;
    uint32 overlayVersion;
    uint32 overlayMinVersion;
    Hash networkID;
    string versionStr<100>;
    int listeningPort;
    NodeID peerID;
    AuthCert cert;
    uint256 nonce;
};
```

HmacSha256Key

class stellar_sdk.xdr.hmac_sha256_key.**HmacSha256Key**(*key*)

XDR Source Code:

```
struct HmacSha256Key
{
    opaque key[32];
};
```

HmacSha256Mac

class stellar_sdk.xdr.hmac_sha256_mac.**HmacSha256Mac**(*mac*)

XDR Source Code:

```
struct HmacSha256Mac
{
    opaque mac[32];
};
```

HostFunction

class stellar_sdk.xdr.host_function.**HostFunction**(*type*, *invoke_contract=None*, *create_contract=None*, *wasm=None*, *create_contract_v2=None*)

XDR Source Code:

```
union HostFunction switch (HostFunctionType type)
{
case HOST_FUNCTION_TYPE_INVOKE_CONTRACT:
```

(continues on next page)

(continued from previous page)

```
    InvokeContractArgs invokeContract;
case HOST_FUNCTION_TYPE_CREATE_CONTRACT:
    CreateContractArgs createContract;
case HOST_FUNCTION_TYPE_UPLOAD_CONTRACT_WASM:
    opaque wasm<>;
case HOST_FUNCTION_TYPE_CREATE_CONTRACT_V2:
    CreateContractArgsV2 createContractV2;
};
```

HostFunctionType

class stellar_sdk.xdr.host_function_type.**HostFunctionType**(**values*)

XDR Source Code:

```
enum HostFunctionType
{
    HOST_FUNCTION_TYPE_INVOKE_CONTRACT = 0,
    HOST_FUNCTION_TYPE_CREATE_CONTRACT = 1,
    HOST_FUNCTION_TYPE_UPLOAD_CONTRACT_WASM = 2,
    HOST_FUNCTION_TYPE_CREATE_CONTRACT_V2 = 3
};
```

HotArchiveBucketEntry

class stellar_sdk.xdr.hot_archive_bucket_entry.**HotArchiveBucketEntry**(*type*, *archived_entry=None*, *key=None*, *meta_entry=None*)

XDR Source Code:

```
union HotArchiveBucketEntry switch (HotArchiveBucketEntryType type)
{
case HOT_ARCHIVE_ARCHIVED:
    LedgerEntry archivedEntry;

case HOT_ARCHIVE_LIVE:
    LedgerKey key;
case HOT_ARCHIVE_METAENTRY:
    BucketMetadata metaEntry;
};
```

HotArchiveBucketEntryType

class stellar_sdk.xdr.hot_archive_bucket_entry_type.**HotArchiveBucketEntryType**(**values*)

XDR Source Code:

```
enum HotArchiveBucketEntryType
{
    HOT_ARCHIVE_METAENTRY = -1, // Bucket metadata, should come first.
    HOT_ARCHIVE_ARCHIVED = 0,   // Entry is Archived
    HOT_ARCHIVE_LIVE = 1        // Entry was previously HOT_ARCHIVE_ARCHIVED, but
```

(continues on next page)

(continued from previous page)

```
                                // has been added back to the live BucketList.
                                // Does not need to be persisted.
};
```

Hyper

class stellar_sdk.xdr.base.**Hyper**(*value*)

IPAddrType

class stellar_sdk.xdr.ip_addr_type.**IPAddrType**(**values*)

XDR Source Code:

```
enum IPAddrType
{
    IPv4 = 0,
    IPv6 = 1
};
```

InflationPayout

class stellar_sdk.xdr.inflation_payout.**InflationPayout**(*destination*, *amount*)

XDR Source Code:

```
struct InflationPayout // or use PaymentResultAtom to limit types?
{
    AccountID destination;
    int64 amount;
};
```

InflationResult

class stellar_sdk.xdr.inflation_result.**InflationResult**(*code*, *payouts=None*)

XDR Source Code:

```
union InflationResult switch (InflationResultCode code)
{
case INFLATION_SUCCESS:
    InflationPayout payouts<>;
case INFLATION_NOT_TIME:
    void;
};
```

InflationResultCode

class stellar_sdk.xdr.inflation_result_code.**InflationResultCode**(**values*)

XDR Source Code:

```
enum InflationResultCode
{
    // codes considered as "success" for the operation
```

(continues on next page)

(continued from previous page)

```
    INFLATION_SUCCESS = 0,
    // codes considered as "failure" for the operation
    INFLATION_NOT_TIME = -1
};
```

InnerTransactionResult

class stellar_sdk.xdr.inner_transaction_result.**InnerTransactionResult**(*fee_charged*, *result*, *ext*)

XDR Source Code:

```
struct InnerTransactionResult
{
    // Always 0. Here for binary compatibility.
    int64 feeCharged;

    union switch (TransactionResultCode code)
    {
    // txFEE_BUMP_INNER_SUCCESS is not included
    case txSUCCESS:
    case txFAILED:
        OperationResult results<>;
    case txTOO_EARLY:
    case txTOO_LATE:
    case txMISSING_OPERATION:
    case txBAD_SEQ:
    case txBAD_AUTH:
    case txINSUFFICIENT_BALANCE:
    case txNO_ACCOUNT:
    case txINSUFFICIENT_FEE:
    case txBAD_AUTH_EXTRA:
    case txINTERNAL_ERROR:
    case txNOT_SUPPORTED:
    // txFEE_BUMP_INNER_FAILED is not included
    case txBAD_SPONSORSHIP:
    case txBAD_MIN_SEQ_AGE_OR_GAP:
    case txMALFORMED:
    case txSOROBAN_INVALID:
    case txFROZEN_KEY_ACCESSED:
        void;
    }
    result;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

InnerTransactionResultExt

class stellar_sdk.xdr.inner_transaction_result_ext.**InnerTransactionResultExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

InnerTransactionResultPair

class stellar_sdk.xdr.inner_transaction_result_pair.**InnerTransactionResultPair**(*transaction_hash*, *result*)

XDR Source Code:

```
struct InnerTransactionResultPair
{
    Hash transactionHash;          // hash of the inner transaction
    InnerTransactionResult result; // result for the inner transaction
};
```

InnerTransactionResultResult

class stellar_sdk.xdr.inner_transaction_result_result.**InnerTransactionResultResult**(*code*, *results=None*)

XDR Source Code:

```
union switch (TransactionResultCode code)
    {
    // txFEE_BUMP_INNER_SUCCESS is not included
    case txSUCCESS:
    case txFAILED:
        OperationResult results<>;
    case txTOO_EARLY:
    case txTOO_LATE:
    case txMISSING_OPERATION:
    case txBAD_SEQ:
    case txBAD_AUTH:
    case txINSUFFICIENT_BALANCE:
    case txNO_ACCOUNT:
    case txINSUFFICIENT_FEE:
    case txBAD_AUTH_EXTRA:
    case txINTERNAL_ERROR:
    case txNOT_SUPPORTED:
    // txFEE_BUMP_INNER_FAILED is not included
    case txBAD_SPONSORSHIP:
    case txBAD_MIN_SEQ_AGE_OR_GAP:
    case txMALFORMED:
    case txSOROBAN_INVALID:
    case txFROZEN_KEY_ACCESSED:
```

(continues on next page)

(continued from previous page)

```
        void;
    }
```

Int128Parts

class stellar_sdk.xdr.int128_parts.**Int128Parts**(*hi*, *lo*)

XDR Source Code:

```
struct Int128Parts {
    int64 hi;
    uint64 lo;
};
```

Int256Parts

class stellar_sdk.xdr.int256_parts.**Int256Parts**(*hi_hi*, *hi_lo*, *lo_hi*, *lo_lo*)

XDR Source Code:

```
struct Int256Parts {
    int64 hi_hi;
    uint64 hi_lo;
    uint64 lo_hi;
    uint64 lo_lo;
};
```

Int32

class stellar_sdk.xdr.int32.**Int32**(*int32*)

XDR Source Code:

```
typedef int int32;
```

Int64

class stellar_sdk.xdr.int64.**Int64**(*int64*)

XDR Source Code:

```
typedef hyper int64;
```

Integer

class stellar_sdk.xdr.base.**Integer**(*value*)

InvokeContractArgs

class stellar_sdk.xdr.invoke_contract_args.**InvokeContractArgs**(*contract_address*, *function_name*, *args*)

XDR Source Code:

```
struct InvokeContractArgs {
    SCAddress contractAddress;
    SCSymbol functionName;
    SCVal args<>;
};
```

InvokeHostFunctionOp

class stellar_sdk.xdr.invoke_host_function_op.**InvokeHostFunctionOp**(*host_function*, *auth*)

XDR Source Code:

```
struct InvokeHostFunctionOp
{
    // Host function to invoke.
    HostFunction hostFunction;
    // Per-address authorizations for this host function.
    SorobanAuthorizationEntry auth<>;
};
```

InvokeHostFunctionResult

class stellar_sdk.xdr.invoke_host_function_result.**InvokeHostFunctionResult**(*code*, *success=None*)

XDR Source Code:

```
union InvokeHostFunctionResult switch (InvokeHostFunctionResultCode code)
{
case INVOKE_HOST_FUNCTION_SUCCESS:
    Hash success; // sha256(InvokeHostFunctionSuccessPreImage)
case INVOKE_HOST_FUNCTION_MALFORMED:
case INVOKE_HOST_FUNCTION_TRAPPED:
case INVOKE_HOST_FUNCTION_RESOURCE_LIMIT_EXCEEDED:
case INVOKE_HOST_FUNCTION_ENTRY_ARCHIVED:
case INVOKE_HOST_FUNCTION_INSUFFICIENT_REFUNDABLE_FEE:
    void;
};
```

InvokeHostFunctionResultCode

class stellar_sdk.xdr.invoke_host_function_result_code.**InvokeHostFunctionResultCode**(**values*)

XDR Source Code:

```
enum InvokeHostFunctionResultCode
{
    // codes considered as "success" for the operation
    INVOKE_HOST_FUNCTION_SUCCESS = 0,

    // codes considered as "failure" for the operation
    INVOKE_HOST_FUNCTION_MALFORMED = -1,
    INVOKE_HOST_FUNCTION_TRAPPED = -2,
    INVOKE_HOST_FUNCTION_RESOURCE_LIMIT_EXCEEDED = -3,
    INVOKE_HOST_FUNCTION_ENTRY_ARCHIVED = -4,
```

(continues on next page)

(continued from previous page)

```
    INVOKE_HOST_FUNCTION_INSUFFICIENT_REFUNDABLE_FEE = -5
};
```

InvokeHostFunctionSuccessPreImage

class stellar_sdk.xdr.invoke_host_function_success_pre_image.**InvokeHostFunctionSuccessPreImage**(*return_val*
events)

XDR Source Code:

```
struct InvokeHostFunctionSuccessPreImage
{
    SCVal returnValue;
    ContractEvent events<>;
};
```

LedgerBounds

class stellar_sdk.xdr.ledger_bounds.**LedgerBounds**(*min_ledger*, *max_ledger*)

XDR Source Code:

```
struct LedgerBounds
{
    uint32 minLedger;
    uint32 maxLedger; // 0 here means no maxLedger
};
```

LedgerCloseMeta

class stellar_sdk.xdr.ledger_close_meta.**LedgerCloseMeta**(*v*, *v0=None*, *v1=None*, *v2=None*)

XDR Source Code:

```
union LedgerCloseMeta switch (int v)
{
case 0:
    LedgerCloseMetaV0 v0;
case 1:
    LedgerCloseMetaV1 v1;
case 2:
    LedgerCloseMetaV2 v2;
};
```

LedgerCloseMetaBatch

class stellar_sdk.xdr.ledger_close_meta_batch.**LedgerCloseMetaBatch**(*start_sequence*,
end_sequence,
ledger_close_metas)

XDR Source Code:

```
struct LedgerCloseMetaBatch
{
    // starting ledger sequence number in the batch
```

(continues on next page)

(continued from previous page)

```
    uint32 startSequence;

    // ending ledger sequence number in the batch
    uint32 endSequence;

    // Ledger close meta for each ledger within the batch
    LedgerCloseMeta ledgerCloseMetas<>;
};
```

LedgerCloseMetaExt

class stellar_sdk.xdr.ledger_close_meta_ext.**LedgerCloseMetaExt**(*v*, *v1=None*)

XDR Source Code:

```
union LedgerCloseMetaExt switch (int v)
{
case 0:
    void;
case 1:
    LedgerCloseMetaExtV1 v1;
};
```

LedgerCloseMetaExtV1

class stellar_sdk.xdr.ledger_close_meta_ext_v1.**LedgerCloseMetaExtV1**(*ext*, *soroban_fee_write1_kb*)

XDR Source Code:

```
struct LedgerCloseMetaExtV1
{
    ExtensionPoint ext;
    int64 sorobanFeeWrite1KB;
};
```

LedgerCloseMetaV0

class stellar_sdk.xdr.ledger_close_meta_v0.**LedgerCloseMetaV0**(*ledger_header*, *tx_set*, *tx_processing*, *upgrades_processing*, *scp_info*)

XDR Source Code:

```
struct LedgerCloseMetaV0
{
    LedgerHeaderHistoryEntry ledgerHeader;
    // NB: txSet is sorted in "Hash order"
    TransactionSet txSet;

    // NB: transactions are sorted in apply order here
    // fees for all transactions are processed first
    // followed by applying transactions
    TransactionResultMeta txProcessing<>;
```

(continues on next page)

(continued from previous page)

```
    // upgrades are applied last
    UpgradeEntryMeta upgradesProcessing<>;

    // other misc information attached to the ledger close
    SCPHistoryEntry scpInfo<>;
};
```

LedgerCloseMetaV1

class stellar_sdk.xdr.ledger_close_meta_v1.**LedgerCloseMetaV1**(*ext*, *ledger_header*, *tx_set*, *tx_processing*, *upgrades_processing*, *scp_info*, *total_byte_size_of_live_soroban_state*, *evicted_keys*, *unused*)

XDR Source Code:

```
struct LedgerCloseMetaV1
{
    LedgerCloseMetaExt ext;

    LedgerHeaderHistoryEntry ledgerHeader;

    GeneralizedTransactionSet txSet;

    // NB: transactions are sorted in apply order here
    // fees for all transactions are processed first
    // followed by applying transactions
    TransactionResultMeta txProcessing<>;

    // upgrades are applied last
    UpgradeEntryMeta upgradesProcessing<>;

    // other misc information attached to the ledger close
    SCPHistoryEntry scpInfo<>;

    // Size in bytes of live Soroban state, to support downstream
    // systems calculating storage fees correctly.
    uint64 totalByteSizeOfLiveSorobanState;

    // TTL and data/code keys that have been evicted at this ledger.
    LedgerKey evictedKeys<>;

    // Maintained for backwards compatibility, should never be populated.
    LedgerEntry unused<>;
};
```

LedgerCloseMetaV2

class stellar_sdk.xdr.ledger_close_meta_v2.**LedgerCloseMetaV2**(*ext*, *ledger_header*, *tx_set*, *tx_processing*, *upgrades_processing*, *scp_info*, *total_byte_size_of_live_soroban_state*, *evicted_keys*)

XDR Source Code:

```
struct LedgerCloseMetaV2
{
    LedgerCloseMetaExt ext;

    LedgerHeaderHistoryEntry ledgerHeader;

    GeneralizedTransactionSet txSet;

    // NB: transactions are sorted in apply order here
    // fees for all transactions are processed first
    // followed by applying transactions
    TransactionResultMetaV1 txProcessing<>;

    // upgrades are applied last
    UpgradeEntryMeta upgradesProcessing<>;

    // other misc information attached to the ledger close
    SCPHistoryEntry scpInfo<>;

    // Size in bytes of live Soroban state, to support downstream
    // systems calculating storage fees correctly.
    uint64 totalByteSizeOfLiveSorobanState;

    // TTL and data/code keys that have been evicted at this ledger.
    LedgerKey evictedKeys<>;
};
```

LedgerCloseValueSignature

class stellar_sdk.xdr.ledger_close_value_signature.**LedgerCloseValueSignature**(*node_id*, *signature*)

XDR Source Code:

```
struct LedgerCloseValueSignature
{
    NodeID nodeID;       // which node introduced the value
    Signature signature; // nodeID's signature
};
```

LedgerEntry

class stellar_sdk.xdr.ledger_entry.**LedgerEntry**(*last_modified_ledger_seq*, *data*, *ext*)

XDR Source Code:

```
struct LedgerEntry
{
    uint32 lastModifiedLedgerSeq; // ledger the LedgerEntry was last changed

    union switch (LedgerEntryType type)
    {
```

(continues on next page)

(continued from previous page)

```
    case ACCOUNT:
        AccountEntry account;
    case TRUSTLINE:
        TrustLineEntry trustLine;
    case OFFER:
        OfferEntry offer;
    case DATA:
        DataEntry data;
    case CLAIMABLE_BALANCE:
        ClaimableBalanceEntry claimableBalance;
    case LIQUIDITY_POOL:
        LiquidityPoolEntry liquidityPool;
    case CONTRACT_DATA:
        ContractDataEntry contractData;
    case CONTRACT_CODE:
        ContractCodeEntry contractCode;
    case CONFIG_SETTING:
        ConfigSettingEntry configSetting;
    case TTL:
        TTLEntry ttl;
    }
    data;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        LedgerEntryExtensionV1 v1;
    }
    ext;
};
```

LedgerEntryChange

class stellar_sdk.xdr.ledger_entry_change.**LedgerEntryChange**(*type*, *created=None*, *updated=None*, *removed=None*, *state=None*, *restored=None*)

XDR Source Code:

```
union LedgerEntryChange switch (LedgerEntryChangeType type)
{
case LEDGER_ENTRY_CREATED:
    LedgerEntry created;
case LEDGER_ENTRY_UPDATED:
    LedgerEntry updated;
case LEDGER_ENTRY_REMOVED:
    LedgerKey removed;
case LEDGER_ENTRY_STATE:
    LedgerEntry state;
case LEDGER_ENTRY_RESTORED:
```

(continues on next page)

(continued from previous page)

```
    LedgerEntry restored;
};
```

LedgerEntryChangeType

class stellar_sdk.xdr.ledger_entry_change_type.**LedgerEntryChangeType**(**values*)

XDR Source Code:

```
enum LedgerEntryChangeType
{
    LEDGER_ENTRY_CREATED = 0, // entry was added to the ledger
    LEDGER_ENTRY_UPDATED = 1, // entry was modified in the ledger
    LEDGER_ENTRY_REMOVED = 2, // entry was removed from the ledger
    LEDGER_ENTRY_STATE    = 3, // value of the entry
    LEDGER_ENTRY_RESTORED = 4  // archived entry was restored in the ledger
};
```

LedgerEntryChanges

class stellar_sdk.xdr.ledger_entry_changes.**LedgerEntryChanges**(*ledger_entry_changes*)

XDR Source Code:

```
typedef LedgerEntryChange LedgerEntryChanges<>;
```

LedgerEntryData

class stellar_sdk.xdr.ledger_entry_data.**LedgerEntryData**(*type*, *account=None*, *trust_line=None*, *offer=None*, *data=None*, *claimable_balance=None*, *liquidity_pool=None*, *contract_data=None*, *contract_code=None*, *config_setting=None*, *ttl=None*)

XDR Source Code:

```
union switch (LedgerEntryType type)
    {
    case ACCOUNT:
        AccountEntry account;
    case TRUSTLINE:
        TrustLineEntry trustLine;
    case OFFER:
        OfferEntry offer;
    case DATA:
        DataEntry data;
    case CLAIMABLE_BALANCE:
        ClaimableBalanceEntry claimableBalance;
    case LIQUIDITY_POOL:
        LiquidityPoolEntry liquidityPool;
    case CONTRACT_DATA:
        ContractDataEntry contractData;
    case CONTRACT_CODE:
```

(continues on next page)

(continued from previous page)

```
        ContractCodeEntry contractCode;
    case CONFIG_SETTING:
        ConfigSettingEntry configSetting;
    case TTL:
        TTLEntry ttl;
    }
```

LedgerEntryExt

class stellar_sdk.xdr.ledger_entry_ext.**LedgerEntryExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        LedgerEntryExtensionV1 v1;
    }
```

LedgerEntryExtensionV1

class stellar_sdk.xdr.ledger_entry_extension_v1.**LedgerEntryExtensionV1**(*sponsoring_id*, *ext*)

XDR Source Code:

```
struct LedgerEntryExtensionV1
{
    SponsorshipDescriptor sponsoringID;

    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

LedgerEntryExtensionV1Ext

class stellar_sdk.xdr.ledger_entry_extension_v1_ext.**LedgerEntryExtensionV1Ext**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

LedgerEntryType

class stellar_sdk.xdr.ledger_entry_type.**LedgerEntryType**(**values*)

XDR Source Code:

```
enum LedgerEntryType
{
    ACCOUNT = 0,
    TRUSTLINE = 1,
    OFFER = 2,
    DATA = 3,
    CLAIMABLE_BALANCE = 4,
    LIQUIDITY_POOL = 5,
    CONTRACT_DATA = 6,
    CONTRACT_CODE = 7,
    CONFIG_SETTING = 8,
    TTL = 9
};
```

LedgerFootprint

class stellar_sdk.xdr.ledger_footprint.**LedgerFootprint**(*read_only*, *read_write*)

XDR Source Code:

```
struct LedgerFootprint
{
    LedgerKey readOnly<>;
    LedgerKey readWrite<>;
};
```

LedgerHeader

class stellar_sdk.xdr.ledger_header.**LedgerHeader**(*ledger_version*, *previous_ledger_hash*, *scp_value*, *tx_set_result_hash*, *bucket_list_hash*, *ledger_seq*, *total_coins*, *fee_pool*, *inflation_seq*, *id_pool*, *base_fee*, *base_reserve*, *max_tx_set_size*, *skip_list*, *ext*)

XDR Source Code:

```
struct LedgerHeader
{
    uint32 ledgerVersion;    // the protocol version of the ledger
    Hash previousLedgerHash; // hash of the previous ledger header
    StellarValue scpValue;   // what consensus agreed to
    Hash txSetResultHash;    // the TransactionResultSet that led to this ledger
    Hash bucketListHash;     // hash of the ledger state

    uint32 ledgerSeq; // sequence number of this ledger

    int64 totalCoins; // total number of stroops in existence.
                      // 10,000,000 stroops in 1 XLM

    int64 feePool;       // fees burned since last inflation run
```

(continues on next page)

(continued from previous page)

```
    uint32 inflationSeq; // inflation sequence number

    uint64 idPool; // last used global ID, used for generating objects

    uint32 baseFee;     // base fee per operation in stroops
    uint32 baseReserve; // account base reserve in stroops

    uint32 maxTxSetSize; // maximum size a transaction set can be

    Hash skipList[4]; // hashes of ledgers in the past. allows you to jump back
                      // in time without walking the chain back ledger by ledger
                      // each slot contains the oldest ledger that is mod of
                      // either 50  5000  50000 or 500000 depending on index
                      // skipList[0] mod(50), skipList[1] mod(5000), etc

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        LedgerHeaderExtensionV1 v1;
    }
    ext;
};
```

LedgerHeaderExt

class stellar_sdk.xdr.ledger_header_ext.**LedgerHeaderExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        LedgerHeaderExtensionV1 v1;
    }
```

LedgerHeaderExtensionV1

class stellar_sdk.xdr.ledger_header_extension_v1.**LedgerHeaderExtensionV1**(*flags*, *ext*)

XDR Source Code:

```
struct LedgerHeaderExtensionV1
{
    uint32 flags; // LedgerHeaderFlags

    union switch (int v)
    {
    case 0:
        void;
```

(continues on next page)

(continued from previous page)

```
    }
    ext;
};
```

LedgerHeaderExtensionV1Ext

class stellar_sdk.xdr.ledger_header_extension_v1_ext.**LedgerHeaderExtensionV1Ext**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

LedgerHeaderFlags

class stellar_sdk.xdr.ledger_header_flags.**LedgerHeaderFlags**(**values*)

XDR Source Code:

```
enum LedgerHeaderFlags
{
    DISABLE_LIQUIDITY_POOL_TRADING_FLAG = 0x1,
    DISABLE_LIQUIDITY_POOL_DEPOSIT_FLAG = 0x2,
    DISABLE_LIQUIDITY_POOL_WITHDRAWAL_FLAG = 0x4
};
```

LedgerHeaderHistoryEntry

class stellar_sdk.xdr.ledger_header_history_entry.**LedgerHeaderHistoryEntry**(*hash*, *header*, *ext*)

XDR Source Code:

```
struct LedgerHeaderHistoryEntry
{
    Hash hash;
    LedgerHeader header;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

LedgerHeaderHistoryEntryExt

class stellar_sdk.xdr.ledger_header_history_entry_ext.**LedgerHeaderHistoryEntryExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

LedgerKey

class stellar_sdk.xdr.ledger_key.**LedgerKey**(*type*, *account=None*, *trust_line=None*, *offer=None*, *data=None*, *claimable_balance=None*, *liquidity_pool=None*, *contract_data=None*, *contract_code=None*, *config_setting=None*, *ttl=None*)

XDR Source Code:

```
union LedgerKey switch (LedgerEntryType type)
{
case ACCOUNT:
    struct
    {
        AccountID accountID;
    } account;

case TRUSTLINE:
    struct
    {
        AccountID accountID;
        TrustLineAsset asset;
    } trustLine;

case OFFER:
    struct
    {
        AccountID sellerID;
        int64 offerID;
    } offer;

case DATA:
    struct
    {
        AccountID accountID;
        string64 dataName;
    } data;

case CLAIMABLE_BALANCE:
    struct
    {
        ClaimableBalanceID balanceID;
    } claimableBalance;

case LIQUIDITY_POOL:
    struct
    {
```

(continues on next page)

(continued from previous page)

```
        PoolID liquidityPoolID;
    } liquidityPool;
case CONTRACT_DATA:
    struct
    {
        SCAddress contract;
        SCVal key;
        ContractDataDurability durability;
    } contractData;
case CONTRACT_CODE:
    struct
    {
        Hash hash;
    } contractCode;
case CONFIG_SETTING:
    struct
    {
        ConfigSettingID configSettingID;
    } configSetting;
case TTL:
    struct
    {
        // Hash of the LedgerKey that is associated with this TTLEntry
        Hash keyHash;
    } ttl;
};
```

LedgerKeyAccount

class stellar_sdk.xdr.ledger_key_account.**LedgerKeyAccount**(*account_id*)

XDR Source Code:

```
struct
    {
        AccountID accountID;
    }
```

LedgerKeyClaimableBalance

class stellar_sdk.xdr.ledger_key_claimable_balance.**LedgerKeyClaimableBalance**(*balance_id*)

XDR Source Code:

```
struct
    {
        ClaimableBalanceID balanceID;
    }
```

LedgerKeyConfigSetting

class stellar_sdk.xdr.ledger_key_config_setting.**LedgerKeyConfigSetting**(*config_setting_id*)

XDR Source Code:

```
struct
    {
        ConfigSettingID configSettingID;
    }
```

LedgerKeyContractCode

class stellar_sdk.xdr.ledger_key_contract_code.**LedgerKeyContractCode**(*hash*)

XDR Source Code:

```
struct
    {
        Hash hash;
    }
```

LedgerKeyContractData

class stellar_sdk.xdr.ledger_key_contract_data.**LedgerKeyContractData**(*contract*, *key*, *durability*)

XDR Source Code:

```
struct
    {
        SCAddress contract;
        SCVal key;
        ContractDataDurability durability;
    }
```

LedgerKeyData

class stellar_sdk.xdr.ledger_key_data.**LedgerKeyData**(*account_id*, *data_name*)

XDR Source Code:

```
struct
    {
        AccountID accountID;
        string64 dataName;
    }
```

LedgerKeyLiquidityPool

class stellar_sdk.xdr.ledger_key_liquidity_pool.**LedgerKeyLiquidityPool**(*liquidity_pool_id*)

XDR Source Code:

```
struct
    {
        PoolID liquidityPoolID;
    }
```

LedgerKeyOffer

class stellar_sdk.xdr.ledger_key_offer.**LedgerKeyOffer**(*seller_id*, *offer_id*)

XDR Source Code:

```
struct
    {
        AccountID sellerID;
        int64 offerID;
    }
```

LedgerKeyTrustLine

class stellar_sdk.xdr.ledger_key_trust_line.**LedgerKeyTrustLine**(*account_id*, *asset*)

XDR Source Code:

```
struct
    {
        AccountID accountID;
        TrustLineAsset asset;
    }
```

LedgerKeyTtl

class stellar_sdk.xdr.ledger_key_ttl.**LedgerKeyTtl**(*key_hash*)

XDR Source Code:

```
struct
    {
        // Hash of the LedgerKey that is associated with this TTLEntry
        Hash keyHash;
    }
```

LedgerSCPMessages

class stellar_sdk.xdr.ledger_scp_messages.**LedgerSCPMessages**(*ledger_seq*, *messages*)

XDR Source Code:

```
struct LedgerSCPMessages
{
    uint32 ledgerSeq;
    SCPEnvelope messages<>;
};
```

LedgerUpgrade

class stellar_sdk.xdr.ledger_upgrade.**LedgerUpgrade**(*type*, *new_ledger_version=None*, *new_base_fee=None*, *new_max_tx_set_size=None*, *new_base_reserve=None*, *new_flags=None*, *new_config=None*, *new_max_soroban_tx_set_size=None*)

XDR Source Code:

```
union LedgerUpgrade switch (LedgerUpgradeType type)
{
case LEDGER_UPGRADE_VERSION:
    uint32 newLedgerVersion; // update ledgerVersion
case LEDGER_UPGRADE_BASE_FEE:
    uint32 newBaseFee; // update baseFee
case LEDGER_UPGRADE_MAX_TX_SET_SIZE:
    uint32 newMaxTxSetSize; // update maxTxSetSize
case LEDGER_UPGRADE_BASE_RESERVE:
    uint32 newBaseReserve; // update baseReserve
case LEDGER_UPGRADE_FLAGS:
    uint32 newFlags; // update flags
case LEDGER_UPGRADE_CONFIG:
    // Update arbitrary `ConfigSetting` entries identified by the key.
    ConfigUpgradeSetKey newConfig;
case LEDGER_UPGRADE_MAX_SOROBAN_TX_SET_SIZE:
    // Update ConfigSettingContractExecutionLanesV0.ledgerMaxTxCount without
    // using `LEDGER_UPGRADE_CONFIG`.
    uint32 newMaxSorobanTxSetSize;
};
```

LedgerUpgradeType

class stellar_sdk.xdr.ledger_upgrade_type.**LedgerUpgradeType**(**values*)

XDR Source Code:

```
enum LedgerUpgradeType
{
    LEDGER_UPGRADE_VERSION = 1,
    LEDGER_UPGRADE_BASE_FEE = 2,
    LEDGER_UPGRADE_MAX_TX_SET_SIZE = 3,
    LEDGER_UPGRADE_BASE_RESERVE = 4,
    LEDGER_UPGRADE_FLAGS = 5,
    LEDGER_UPGRADE_CONFIG = 6,
    LEDGER_UPGRADE_MAX_SOROBAN_TX_SET_SIZE = 7
};
```

Liabilities

class stellar_sdk.xdr.liabilities.**Liabilities**(*buying*, *selling*)

XDR Source Code:

```
struct Liabilities
{
    int64 buying;
    int64 selling;
};
```

LiquidityPoolConstantProductParameters

class stellar_sdk.xdr.liquidity_pool_constant_product_parameters.**LiquidityPoolConstantProductParameters**

XDR Source Code:

```
struct LiquidityPoolConstantProductParameters
{
    Asset assetA; // assetA < assetB
    Asset assetB;
    int32 fee; // Fee is in basis points, so the actual rate is (fee/100)%
};
```

LiquidityPoolDepositOp

class stellar_sdk.xdr.liquidity_pool_deposit_op.**LiquidityPoolDepositOp**(*liquidity_pool_id*, *max_amount_a*, *max_amount_b*, *min_price*, *max_price*)

XDR Source Code:

```
struct LiquidityPoolDepositOp
{
    PoolID liquidityPoolID;
    int64 maxAmountA; // maximum amount of first asset to deposit
    int64 maxAmountB; // maximum amount of second asset to deposit
    Price minPrice;   // minimum depositA/depositB
    Price maxPrice;   // maximum depositA/depositB
};
```

LiquidityPoolDepositResult

class stellar_sdk.xdr.liquidity_pool_deposit_result.**LiquidityPoolDepositResult**(*code*)

XDR Source Code:

```
union LiquidityPoolDepositResult switch (LiquidityPoolDepositResultCode code)
{
case LIQUIDITY_POOL_DEPOSIT_SUCCESS:
    void;
case LIQUIDITY_POOL_DEPOSIT_MALFORMED:
case LIQUIDITY_POOL_DEPOSIT_NO_TRUST:
case LIQUIDITY_POOL_DEPOSIT_NOT_AUTHORIZED:
case LIQUIDITY_POOL_DEPOSIT_UNDERFUNDED:
case LIQUIDITY_POOL_DEPOSIT_LINE_FULL:
case LIQUIDITY_POOL_DEPOSIT_BAD_PRICE:
case LIQUIDITY_POOL_DEPOSIT_POOL_FULL:
case LIQUIDITY_POOL_DEPOSIT_TRUSTLINE_FROZEN:
    void;
};
```

LiquidityPoolDepositResultCode

class stellar_sdk.xdr.liquidity_pool_deposit_result_code.**LiquidityPoolDepositResultCode**(**values*)

XDR Source Code:

```
enum LiquidityPoolDepositResultCode
{
    // codes considered as "success" for the operation
    LIQUIDITY_POOL_DEPOSIT_SUCCESS = 0,

    // codes considered as "failure" for the operation
    LIQUIDITY_POOL_DEPOSIT_MALFORMED = -1,      // bad input
    LIQUIDITY_POOL_DEPOSIT_NO_TRUST = -2,       // no trust line for one of the
                                                // assets
    LIQUIDITY_POOL_DEPOSIT_NOT_AUTHORIZED = -3, // not authorized for one of the
                                                // assets
    LIQUIDITY_POOL_DEPOSIT_UNDERFUNDED = -4,    // not enough balance for one of
                                                // the assets
    LIQUIDITY_POOL_DEPOSIT_LINE_FULL = -5,      // pool share trust line doesn't
                                                // have sufficient limit
    LIQUIDITY_POOL_DEPOSIT_BAD_PRICE = -6,      // deposit price outside bounds
    LIQUIDITY_POOL_DEPOSIT_POOL_FULL = -7,      // pool reserves are full
    LIQUIDITY_POOL_DEPOSIT_TRUSTLINE_FROZEN = -8  // trustline for one of the
                                                  // assets is frozen
};
```

LiquidityPoolEntry

class stellar_sdk.xdr.liquidity_pool_entry.**LiquidityPoolEntry**(*liquidity_pool_id*, *body*)

XDR Source Code:

```
struct LiquidityPoolEntry
{
    PoolID liquidityPoolID;

    union switch (LiquidityPoolType type)
    {
    case LIQUIDITY_POOL_CONSTANT_PRODUCT:
        struct
        {
            LiquidityPoolConstantProductParameters params;

            int64 reserveA;        // amount of A in the pool
            int64 reserveB;        // amount of B in the pool
            int64 totalPoolShares; // total number of pool shares issued
            int64 poolSharesTrustLineCount; // number of trust lines for the
                                            // associated pool shares
        } constantProduct;
    }
    body;
};
```

LiquidityPoolEntryBody

class stellar_sdk.xdr.liquidity_pool_entry_body.**LiquidityPoolEntryBody**(*type*, *constant_product=None*)

XDR Source Code:

```
union switch (LiquidityPoolType type)
    {
    case LIQUIDITY_POOL_CONSTANT_PRODUCT:
        struct
        {
            LiquidityPoolConstantProductParameters params;

            int64 reserveA;        // amount of A in the pool
            int64 reserveB;        // amount of B in the pool
            int64 totalPoolShares; // total number of pool shares issued
            int64 poolSharesTrustLineCount; // number of trust lines for the
                                            // associated pool shares
        } constantProduct;
    }
```

LiquidityPoolEntryConstantProduct

class stellar_sdk.xdr.liquidity_pool_entry_constant_product.**LiquidityPoolEntryConstantProduct**(*params*, *reserve_a*, *reserve_b*, *total_pool_sh pool_shares*

XDR Source Code:

```
struct
        {
            LiquidityPoolConstantProductParameters params;

            int64 reserveA;        // amount of A in the pool
            int64 reserveB;        // amount of B in the pool
            int64 totalPoolShares; // total number of pool shares issued
            int64 poolSharesTrustLineCount; // number of trust lines for the
                                            // associated pool shares
        }
```

LiquidityPoolParameters

class stellar_sdk.xdr.liquidity_pool_parameters.**LiquidityPoolParameters**(*type*, *constant_product=None*)

XDR Source Code:

```
union LiquidityPoolParameters switch (LiquidityPoolType type)
{
case LIQUIDITY_POOL_CONSTANT_PRODUCT:
```

(continues on next page)

(continued from previous page)

```
    LiquidityPoolConstantProductParameters constantProduct;
};
```

LiquidityPoolType

class stellar_sdk.xdr.liquidity_pool_type.**LiquidityPoolType**(**values*)

XDR Source Code:

```
enum LiquidityPoolType
{
    LIQUIDITY_POOL_CONSTANT_PRODUCT = 0
};
```

LiquidityPoolWithdrawOp

class stellar_sdk.xdr.liquidity_pool_withdraw_op.**LiquidityPoolWithdrawOp**(*liquidity_pool_id*, *amount*, *min_amount_a*, *min_amount_b*)

XDR Source Code:

```
struct LiquidityPoolWithdrawOp
{
    PoolID liquidityPoolID;
    int64 amount;     // amount of pool shares to withdraw
    int64 minAmountA; // minimum amount of first asset to withdraw
    int64 minAmountB; // minimum amount of second asset to withdraw
};
```

LiquidityPoolWithdrawResult

class stellar_sdk.xdr.liquidity_pool_withdraw_result.**LiquidityPoolWithdrawResult**(*code*)

XDR Source Code:

```
union LiquidityPoolWithdrawResult switch (LiquidityPoolWithdrawResultCode code)
{
case LIQUIDITY_POOL_WITHDRAW_SUCCESS:
    void;
case LIQUIDITY_POOL_WITHDRAW_MALFORMED:
case LIQUIDITY_POOL_WITHDRAW_NO_TRUST:
case LIQUIDITY_POOL_WITHDRAW_UNDERFUNDED:
case LIQUIDITY_POOL_WITHDRAW_LINE_FULL:
case LIQUIDITY_POOL_WITHDRAW_UNDER_MINIMUM:
case LIQUIDITY_POOL_WITHDRAW_TRUSTLINE_FROZEN:
    void;
};
```

LiquidityPoolWithdrawResultCode

class stellar_sdk.xdr.liquidity_pool_withdraw_result_code.**LiquidityPoolWithdrawResultCode**(**values*)

XDR Source Code:

```
enum LiquidityPoolWithdrawResultCode
{
    // codes considered as "success" for the operation
    LIQUIDITY_POOL_WITHDRAW_SUCCESS = 0,

    // codes considered as "failure" for the operation
    LIQUIDITY_POOL_WITHDRAW_MALFORMED = -1,    // bad input
    LIQUIDITY_POOL_WITHDRAW_NO_TRUST = -2,     // no trust line for one of the
                                               // assets
    LIQUIDITY_POOL_WITHDRAW_UNDERFUNDED = -3,  // not enough balance of the
                                               // pool share
    LIQUIDITY_POOL_WITHDRAW_LINE_FULL = -4,    // would go above limit for one
                                               // of the assets
    LIQUIDITY_POOL_WITHDRAW_UNDER_MINIMUM = -5, // didn't withdraw enough
    LIQUIDITY_POOL_WITHDRAW_TRUSTLINE_FROZEN = -6  // trustline for one of the
                                                   // assets is frozen
};
```

ManageBuyOfferOp

class stellar_sdk.xdr.manage_buy_offer_op.**ManageBuyOfferOp**(*selling*, *buying*, *buy_amount*, *price*, *offer_id*)

XDR Source Code:

```
struct ManageBuyOfferOp
{
    Asset selling;
    Asset buying;
    int64 buyAmount; // amount being bought. if set to 0, delete the offer
    Price price;     // price of thing being bought in terms of what you are
                     // selling

    // 0=create a new offer, otherwise edit an existing offer
    int64 offerID;
};
```

ManageBuyOfferResult

class stellar_sdk.xdr.manage_buy_offer_result.**ManageBuyOfferResult**(*code*, *success=None*)

XDR Source Code:

```
union ManageBuyOfferResult switch (ManageBuyOfferResultCode code)
{
case MANAGE_BUY_OFFER_SUCCESS:
    ManageOfferSuccessResult success;
case MANAGE_BUY_OFFER_MALFORMED:
case MANAGE_BUY_OFFER_SELL_NO_TRUST:
case MANAGE_BUY_OFFER_BUY_NO_TRUST:
```

(continues on next page)

(continued from previous page)

```
case MANAGE_BUY_OFFER_SELL_NOT_AUTHORIZED:
case MANAGE_BUY_OFFER_BUY_NOT_AUTHORIZED:
case MANAGE_BUY_OFFER_LINE_FULL:
case MANAGE_BUY_OFFER_UNDERFUNDED:
case MANAGE_BUY_OFFER_CROSS_SELF:
case MANAGE_BUY_OFFER_SELL_NO_ISSUER:
case MANAGE_BUY_OFFER_BUY_NO_ISSUER:
case MANAGE_BUY_OFFER_NOT_FOUND:
case MANAGE_BUY_OFFER_LOW_RESERVE:
    void;
};
```

ManageBuyOfferResultCode

class stellar_sdk.xdr.manage_buy_offer_result_code.**ManageBuyOfferResultCode**(**values*)

XDR Source Code:

```
enum ManageBuyOfferResultCode
{
    // codes considered as "success" for the operation
    MANAGE_BUY_OFFER_SUCCESS = 0,

    // codes considered as "failure" for the operation
    MANAGE_BUY_OFFER_MALFORMED = -1,     // generated offer would be invalid
    MANAGE_BUY_OFFER_SELL_NO_TRUST = -2, // no trust line for what we're selling
    MANAGE_BUY_OFFER_BUY_NO_TRUST = -3,  // no trust line for what we're buying
    MANAGE_BUY_OFFER_SELL_NOT_AUTHORIZED = -4, // not authorized to sell
    MANAGE_BUY_OFFER_BUY_NOT_AUTHORIZED = -5,  // not authorized to buy
    MANAGE_BUY_OFFER_LINE_FULL = -6,   // can't receive more of what it's buying
    MANAGE_BUY_OFFER_UNDERFUNDED = -7, // doesn't hold what it's trying to sell
    MANAGE_BUY_OFFER_CROSS_SELF = -8, // would cross an offer from the same user
    MANAGE_BUY_OFFER_SELL_NO_ISSUER = -9, // no issuer for what we're selling
    MANAGE_BUY_OFFER_BUY_NO_ISSUER = -10, // no issuer for what we're buying

    // update errors
    MANAGE_BUY_OFFER_NOT_FOUND =
        -11, // offerID does not match an existing offer

    MANAGE_BUY_OFFER_LOW_RESERVE = -12 // not enough funds to create a new Offer
};
```

ManageDataOp

class stellar_sdk.xdr.manage_data_op.**ManageDataOp**(*data_name*, *data_value*)

XDR Source Code:

```
struct ManageDataOp
{
    string64 dataName;
    DataValue* dataValue; // set to null to clear
};
```

ManageDataResult

class stellar_sdk.xdr.manage_data_result.**ManageDataResult**(*code*)

XDR Source Code:

```
union ManageDataResult switch (ManageDataResultCode code)
{
case MANAGE_DATA_SUCCESS:
    void;
case MANAGE_DATA_NOT_SUPPORTED_YET:
case MANAGE_DATA_NAME_NOT_FOUND:
case MANAGE_DATA_LOW_RESERVE:
case MANAGE_DATA_INVALID_NAME:
    void;
};
```

ManageDataResultCode

class stellar_sdk.xdr.manage_data_result_code.**ManageDataResultCode**(**values*)

XDR Source Code:

```
enum ManageDataResultCode
{
    // codes considered as "success" for the operation
    MANAGE_DATA_SUCCESS = 0,
    // codes considered as "failure" for the operation
    MANAGE_DATA_NOT_SUPPORTED_YET =
        -1, // The network hasn't moved to this protocol change yet
    MANAGE_DATA_NAME_NOT_FOUND =
        -2, // Trying to remove a Data Entry that isn't there
    MANAGE_DATA_LOW_RESERVE = -3, // not enough funds to create a new Data Entry
    MANAGE_DATA_INVALID_NAME = -4 // Name not a valid string
};
```

ManageOfferEffect

class stellar_sdk.xdr.manage_offer_effect.**ManageOfferEffect**(**values*)

XDR Source Code:

```
enum ManageOfferEffect
{
    MANAGE_OFFER_CREATED = 0,
    MANAGE_OFFER_UPDATED = 1,
    MANAGE_OFFER_DELETED = 2
};
```

ManageOfferSuccessResult

class stellar_sdk.xdr.manage_offer_success_result.**ManageOfferSuccessResult**(*offers_claimed*, *offer*)

XDR Source Code:

```
struct ManageOfferSuccessResult
{
    // offers that got claimed while creating this offer
    ClaimAtom offersClaimed<>;

    union switch (ManageOfferEffect effect)
    {
    case MANAGE_OFFER_CREATED:
    case MANAGE_OFFER_UPDATED:
        OfferEntry offer;
    case MANAGE_OFFER_DELETED:
        void;
    }
    offer;
};
```

ManageOfferSuccessResultOffer

class stellar_sdk.xdr.manage_offer_success_result_offer.**ManageOfferSuccessResultOffer**(*effect*, *offer=None*)

XDR Source Code:

```
union switch (ManageOfferEffect effect)
    {
    case MANAGE_OFFER_CREATED:
    case MANAGE_OFFER_UPDATED:
        OfferEntry offer;
    case MANAGE_OFFER_DELETED:
        void;
    }
```

ManageSellOfferOp

class stellar_sdk.xdr.manage_sell_offer_op.**ManageSellOfferOp**(*selling*, *buying*, *amount*, *price*, *offer_id*)

XDR Source Code:

```
struct ManageSellOfferOp
{
    Asset selling;
    Asset buying;
    int64 amount; // amount being sold. if set to 0, delete the offer
    Price price;  // price of thing being sold in terms of what you are buying

    // 0=create a new offer, otherwise edit an existing offer
    int64 offerID;
};
```

ManageSellOfferResult

class stellar_sdk.xdr.manage_sell_offer_result.**ManageSellOfferResult**(*code*, *success=None*)

XDR Source Code:

```
union ManageSellOfferResult switch (ManageSellOfferResultCode code)
{
case MANAGE_SELL_OFFER_SUCCESS:
    ManageOfferSuccessResult success;
case MANAGE_SELL_OFFER_MALFORMED:
case MANAGE_SELL_OFFER_SELL_NO_TRUST:
case MANAGE_SELL_OFFER_BUY_NO_TRUST:
case MANAGE_SELL_OFFER_SELL_NOT_AUTHORIZED:
case MANAGE_SELL_OFFER_BUY_NOT_AUTHORIZED:
case MANAGE_SELL_OFFER_LINE_FULL:
case MANAGE_SELL_OFFER_UNDERFUNDED:
case MANAGE_SELL_OFFER_CROSS_SELF:
case MANAGE_SELL_OFFER_SELL_NO_ISSUER:
case MANAGE_SELL_OFFER_BUY_NO_ISSUER:
case MANAGE_SELL_OFFER_NOT_FOUND:
case MANAGE_SELL_OFFER_LOW_RESERVE:
    void;
};
```

ManageSellOfferResultCode

class stellar_sdk.xdr.manage_sell_offer_result_code.**ManageSellOfferResultCode**(**values*)

XDR Source Code:

```
enum ManageSellOfferResultCode
{
    // codes considered as "success" for the operation
    MANAGE_SELL_OFFER_SUCCESS = 0,

    // codes considered as "failure" for the operation
    MANAGE_SELL_OFFER_MALFORMED = -1, // generated offer would be invalid
    MANAGE_SELL_OFFER_SELL_NO_TRUST =
        -2,                               // no trust line for what we're selling
    MANAGE_SELL_OFFER_BUY_NO_TRUST = -3, // no trust line for what we're buying
    MANAGE_SELL_OFFER_SELL_NOT_AUTHORIZED = -4, // not authorized to sell
    MANAGE_SELL_OFFER_BUY_NOT_AUTHORIZED = -5,  // not authorized to buy
    MANAGE_SELL_OFFER_LINE_FULL = -6, // can't receive more of what it's buying
    MANAGE_SELL_OFFER_UNDERFUNDED = -7, // doesn't hold what it's trying to sell
    MANAGE_SELL_OFFER_CROSS_SELF =
        -8, // would cross an offer from the same user
    MANAGE_SELL_OFFER_SELL_NO_ISSUER = -9, // no issuer for what we're selling
    MANAGE_SELL_OFFER_BUY_NO_ISSUER = -10, // no issuer for what we're buying

    // update errors
    MANAGE_SELL_OFFER_NOT_FOUND =
        -11, // offerID does not match an existing offer

    MANAGE_SELL_OFFER_LOW_RESERVE =
```

(continues on next page)

(continued from previous page)

```
        -12 // not enough funds to create a new Offer
};
```

Memo

class stellar_sdk.xdr.memo.**Memo**(*type*, *text=None*, *id=None*, *hash=None*, *ret_hash=None*)

XDR Source Code:

```
union Memo switch (MemoType type)
{
case MEMO_NONE:
    void;
case MEMO_TEXT:
    string text<28>;
case MEMO_ID:
    uint64 id;
case MEMO_HASH:
    Hash hash; // the hash of what to pull from the content server
case MEMO_RETURN:
    Hash retHash; // the hash of the tx you are rejecting
};
```

MemoType

class stellar_sdk.xdr.memo_type.**MemoType**(**values*)

XDR Source Code:

```
enum MemoType
{
    MEMO_NONE = 0,
    MEMO_TEXT = 1,
    MEMO_ID = 2,
    MEMO_HASH = 3,
    MEMO_RETURN = 4
};
```

MessageType

class stellar_sdk.xdr.message_type.**MessageType**(**values*)

XDR Source Code:

```
enum MessageType
{
    ERROR_MSG = 0,
    AUTH = 2,
    DONT_HAVE = 3,
    // GET_PEERS (4) is deprecated

    PEERS = 5,

    GET_TX_SET = 6, // gets a particular txset by hash
```

(continues on next page)

(continued from previous page)

```
    TX_SET = 7,
    GENERALIZED_TX_SET = 17,

    TRANSACTION = 8, // pass on a tx you have heard about

    // SCP
    GET_SCP_QUORUMSET = 9,
    SCP_QUORUMSET = 10,
    SCP_MESSAGE = 11,
    GET_SCP_STATE = 12,

    // new messages
    HELLO = 13,

    // SURVEY_REQUEST (14) removed and replaced by TIME_SLICED_SURVEY_REQUEST
    // SURVEY_RESPONSE (15) removed and replaced by TIME_SLICED_SURVEY_RESPONSE

    SEND_MORE = 16,
    SEND_MORE_EXTENDED = 20,

    FLOOD_ADVERT = 18,
    FLOOD_DEMAND = 19,

    TIME_SLICED_SURVEY_REQUEST = 21,
    TIME_SLICED_SURVEY_RESPONSE = 22,
    TIME_SLICED_SURVEY_START_COLLECTING = 23,
    TIME_SLICED_SURVEY_STOP_COLLECTING = 24
};
```

MuxedAccount

class stellar_sdk.xdr.muxed_account.**MuxedAccount**(*type*, *ed25519=None*, *med25519=None*)

XDR Source Code:

```
union MuxedAccount switch (CryptoKeyType type)
{
case KEY_TYPE_ED25519:
    uint256 ed25519;
case KEY_TYPE_MUXED_ED25519:
    struct
    {
        uint64 id;
        uint256 ed25519;
    } med25519;
};
```

MuxedAccountMed25519

class stellar_sdk.xdr.muxed_account_med25519.**MuxedAccountMed25519**(*id*, *ed25519*)

XDR Source Code:

```
struct
    {
        uint64 id;
        uint256 ed25519;
    }
```

MuxedEd25519Account

class stellar_sdk.xdr.muxed_ed25519_account.**MuxedEd25519Account**(*id*, *ed25519*)

XDR Source Code:

```
struct MuxedEd25519Account
{
    uint64 id;
    uint256 ed25519;
};
```

NodeID

class stellar_sdk.xdr.node_id.**NodeID**(*node_id*)

XDR Source Code:

```
typedef PublicKey NodeID;
```

OfferEntry

class stellar_sdk.xdr.offer_entry.**OfferEntry**(*seller_id*, *offer_id*, *selling*, *buying*, *amount*, *price*, *flags*, *ext*)

XDR Source Code:

```
struct OfferEntry
{
    AccountID sellerID;
    int64 offerID;
    Asset selling; // A
    Asset buying;  // B
    int64 amount;  // amount of A

    /* price for this offer:
        price of A in terms of B
        price=AmountB/AmountA=priceNumerator/priceDenominator
        price is after fees
    */
    Price price;
    uint32 flags; // see OfferEntryFlags

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    }
```

(continues on next page)

(continued from previous page)

```
    ext;
};
```

OfferEntryExt

class stellar_sdk.xdr.offer_entry_ext.**OfferEntryExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

OfferEntryFlags

class stellar_sdk.xdr.offer_entry_flags.**OfferEntryFlags**(**values*)

XDR Source Code:

```
enum OfferEntryFlags
{
    // an offer with this flag will not act on and take a reverse offer of equal
    // price
    PASSIVE_FLAG = 1
};
```

Opaque

class stellar_sdk.xdr.base.**Opaque**(*value*, *size*, *fixed*)

Operation

class stellar_sdk.xdr.operation.**Operation**(*source_account*, *body*)

XDR Source Code:

```
struct Operation
{
    // sourceAccount is the account used to run the operation
    // if not set, the runtime defaults to "sourceAccount" specified at
    // the transaction level
    MuxedAccount* sourceAccount;

    union switch (OperationType type)
    {
    case CREATE_ACCOUNT:
        CreateAccountOp createAccountOp;
    case PAYMENT:
        PaymentOp paymentOp;
    case PATH_PAYMENT_STRICT_RECEIVE:
        PathPaymentStrictReceiveOp pathPaymentStrictReceiveOp;
    case MANAGE_SELL_OFFER:
```

(continues on next page)

(continued from previous page)

```
        ManageSellOfferOp manageSellOfferOp;
    case CREATE_PASSIVE_SELL_OFFER:
        CreatePassiveSellOfferOp createPassiveSellOfferOp;
    case SET_OPTIONS:
        SetOptionsOp setOptionsOp;
    case CHANGE_TRUST:
        ChangeTrustOp changeTrustOp;
    case ALLOW_TRUST:
        AllowTrustOp allowTrustOp;
    case ACCOUNT_MERGE:
        MuxedAccount destination;
    case INFLATION:
        void;
    case MANAGE_DATA:
        ManageDataOp manageDataOp;
    case BUMP_SEQUENCE:
        BumpSequenceOp bumpSequenceOp;
    case MANAGE_BUY_OFFER:
        ManageBuyOfferOp manageBuyOfferOp;
    case PATH_PAYMENT_STRICT_SEND:
        PathPaymentStrictSendOp pathPaymentStrictSendOp;
    case CREATE_CLAIMABLE_BALANCE:
        CreateClaimableBalanceOp createClaimableBalanceOp;
    case CLAIM_CLAIMABLE_BALANCE:
        ClaimClaimableBalanceOp claimClaimableBalanceOp;
    case BEGIN_SPONSORING_FUTURE_RESERVES:
        BeginSponsoringFutureReservesOp beginSponsoringFutureReservesOp;
    case END_SPONSORING_FUTURE_RESERVES:
        void;
    case REVOKE_SPONSORSHIP:
        RevokeSponsorshipOp revokeSponsorshipOp;
    case CLAWBACK:
        ClawbackOp clawbackOp;
    case CLAWBACK_CLAIMABLE_BALANCE:
        ClawbackClaimableBalanceOp clawbackClaimableBalanceOp;
    case SET_TRUST_LINE_FLAGS:
        SetTrustLineFlagsOp setTrustLineFlagsOp;
    case LIQUIDITY_POOL_DEPOSIT:
        LiquidityPoolDepositOp liquidityPoolDepositOp;
    case LIQUIDITY_POOL_WITHDRAW:
        LiquidityPoolWithdrawOp liquidityPoolWithdrawOp;
    case INVOKE_HOST_FUNCTION:
        InvokeHostFunctionOp invokeHostFunctionOp;
    case EXTEND_FOOTPRINT_TTL:
        ExtendFootprintTTLOp extendFootprintTTLOp;
    case RESTORE_FOOTPRINT:
        RestoreFootprintOp restoreFootprintOp;
    }
    body;
};
```

OperationBody

class stellar_sdk.xdr.operation_body.**OperationBody**(*type*, *create_account_op=None*, *payment_op=None*, *path_payment_strict_receive_op=None*, *manage_sell_offer_op=None*, *create_passive_sell_offer_op=None*, *set_options_op=None*, *change_trust_op=None*, *allow_trust_op=None*, *destination=None*, *manage_data_op=None*, *bump_sequence_op=None*, *manage_buy_offer_op=None*, *path_payment_strict_send_op=None*, *create_claimable_balance_op=None*, *claim_claimable_balance_op=None*, *begin_sponsoring_future_reserves_op=None*, *revoke_sponsorship_op=None*, *clawback_op=None*, *clawback_claimable_balance_op=None*, *set_trust_line_flags_op=None*, *liquidity_pool_deposit_op=None*, *liquidity_pool_withdraw_op=None*, *invoke_host_function_op=None*, *extend_footprint_ttl_op=None*, *restore_footprint_op=None*)

XDR Source Code:

```
union switch (OperationType type)
    {
    case CREATE_ACCOUNT:
        CreateAccountOp createAccountOp;
    case PAYMENT:
        PaymentOp paymentOp;
    case PATH_PAYMENT_STRICT_RECEIVE:
        PathPaymentStrictReceiveOp pathPaymentStrictReceiveOp;
    case MANAGE_SELL_OFFER:
        ManageSellOfferOp manageSellOfferOp;
    case CREATE_PASSIVE_SELL_OFFER:
        CreatePassiveSellOfferOp createPassiveSellOfferOp;
    case SET_OPTIONS:
        SetOptionsOp setOptionsOp;
    case CHANGE_TRUST:
        ChangeTrustOp changeTrustOp;
    case ALLOW_TRUST:
        AllowTrustOp allowTrustOp;
    case ACCOUNT_MERGE:
        MuxedAccount destination;
    case INFLATION:
        void;
    case MANAGE_DATA:
        ManageDataOp manageDataOp;
    case BUMP_SEQUENCE:
        BumpSequenceOp bumpSequenceOp;
    case MANAGE_BUY_OFFER:
```

(continues on next page)

(continued from previous page)

```
        ManageBuyOfferOp manageBuyOfferOp;
    case PATH_PAYMENT_STRICT_SEND:
        PathPaymentStrictSendOp pathPaymentStrictSendOp;
    case CREATE_CLAIMABLE_BALANCE:
        CreateClaimableBalanceOp createClaimableBalanceOp;
    case CLAIM_CLAIMABLE_BALANCE:
        ClaimClaimableBalanceOp claimClaimableBalanceOp;
    case BEGIN_SPONSORING_FUTURE_RESERVES:
        BeginSponsoringFutureReservesOp beginSponsoringFutureReservesOp;
    case END_SPONSORING_FUTURE_RESERVES:
        void;
    case REVOKE_SPONSORSHIP:
        RevokeSponsorshipOp revokeSponsorshipOp;
    case CLAWBACK:
        ClawbackOp clawbackOp;
    case CLAWBACK_CLAIMABLE_BALANCE:
        ClawbackClaimableBalanceOp clawbackClaimableBalanceOp;
    case SET_TRUST_LINE_FLAGS:
        SetTrustLineFlagsOp setTrustLineFlagsOp;
    case LIQUIDITY_POOL_DEPOSIT:
        LiquidityPoolDepositOp liquidityPoolDepositOp;
    case LIQUIDITY_POOL_WITHDRAW:
        LiquidityPoolWithdrawOp liquidityPoolWithdrawOp;
    case INVOKE_HOST_FUNCTION:
        InvokeHostFunctionOp invokeHostFunctionOp;
    case EXTEND_FOOTPRINT_TTL:
        ExtendFootprintTTLOp extendFootprintTTLOp;
    case RESTORE_FOOTPRINT:
        RestoreFootprintOp restoreFootprintOp;
    }
```

OperationMeta

class stellar_sdk.xdr.operation_meta.**OperationMeta**(*changes*)

XDR Source Code:

```
struct OperationMeta
{
    LedgerEntryChanges changes;
};
```

OperationMetaV2

class stellar_sdk.xdr.operation_meta_v2.**OperationMetaV2**(*ext*, *changes*, *events*)

XDR Source Code:

```
struct OperationMetaV2
{
    ExtensionPoint ext;

    LedgerEntryChanges changes;

```

(continues on next page)

(continued from previous page)

```
    ContractEvent events<>;
};
```

OperationResult

class stellar_sdk.xdr.operation_result.**OperationResult**(*code*, *tr=None*)

XDR Source Code:

```
union OperationResult switch (OperationResultCode code)
{
case opINNER:
    union switch (OperationType type)
    {
    case CREATE_ACCOUNT:
        CreateAccountResult createAccountResult;
    case PAYMENT:
        PaymentResult paymentResult;
    case PATH_PAYMENT_STRICT_RECEIVE:
        PathPaymentStrictReceiveResult pathPaymentStrictReceiveResult;
    case MANAGE_SELL_OFFER:
        ManageSellOfferResult manageSellOfferResult;
    case CREATE_PASSIVE_SELL_OFFER:
        ManageSellOfferResult createPassiveSellOfferResult;
    case SET_OPTIONS:
        SetOptionsResult setOptionsResult;
    case CHANGE_TRUST:
        ChangeTrustResult changeTrustResult;
    case ALLOW_TRUST:
        AllowTrustResult allowTrustResult;
    case ACCOUNT_MERGE:
        AccountMergeResult accountMergeResult;
    case INFLATION:
        InflationResult inflationResult;
    case MANAGE_DATA:
        ManageDataResult manageDataResult;
    case BUMP_SEQUENCE:
        BumpSequenceResult bumpSeqResult;
    case MANAGE_BUY_OFFER:
        ManageBuyOfferResult manageBuyOfferResult;
    case PATH_PAYMENT_STRICT_SEND:
        PathPaymentStrictSendResult pathPaymentStrictSendResult;
    case CREATE_CLAIMABLE_BALANCE:
        CreateClaimableBalanceResult createClaimableBalanceResult;
    case CLAIM_CLAIMABLE_BALANCE:
        ClaimClaimableBalanceResult claimClaimableBalanceResult;
    case BEGIN_SPONSORING_FUTURE_RESERVES:
        BeginSponsoringFutureReservesResult beginSponsoringFutureReservesResult;
    case END_SPONSORING_FUTURE_RESERVES:
        EndSponsoringFutureReservesResult endSponsoringFutureReservesResult;
    case REVOKE_SPONSORSHIP:
        RevokeSponsorshipResult revokeSponsorshipResult;
    case CLAWBACK:
```

(continues on next page)

(continued from previous page)

```
        ClawbackResult clawbackResult;
    case CLAWBACK_CLAIMABLE_BALANCE:
        ClawbackClaimableBalanceResult clawbackClaimableBalanceResult;
    case SET_TRUST_LINE_FLAGS:
        SetTrustLineFlagsResult setTrustLineFlagsResult;
    case LIQUIDITY_POOL_DEPOSIT:
        LiquidityPoolDepositResult liquidityPoolDepositResult;
    case LIQUIDITY_POOL_WITHDRAW:
        LiquidityPoolWithdrawResult liquidityPoolWithdrawResult;
    case INVOKE_HOST_FUNCTION:
        InvokeHostFunctionResult invokeHostFunctionResult;
    case EXTEND_FOOTPRINT_TTL:
        ExtendFootprintTTLResult extendFootprintTTLResult;
    case RESTORE_FOOTPRINT:
        RestoreFootprintResult restoreFootprintResult;
    }
    tr;
case opBAD_AUTH:
case opNO_ACCOUNT:
case opNOT_SUPPORTED:
case opTOO_MANY_SUBENTRIES:
case opEXCEEDED_WORK_LIMIT:
case opTOO_MANY_SPONSORING:
    void;
};
```

OperationResultCode

class stellar_sdk.xdr.operation_result_code.**OperationResultCode**(**values*)

XDR Source Code:

```
enum OperationResultCode
{
    opINNER = 0, // inner object result is valid

    opBAD_AUTH = -1,            // too few valid signatures / wrong network
    opNO_ACCOUNT = -2,          // source account was not found
    opNOT_SUPPORTED = -3,       // operation not supported at this time
    opTOO_MANY_SUBENTRIES = -4, // max number of subentries already reached
    opEXCEEDED_WORK_LIMIT = -5, // operation did too much work
    opTOO_MANY_SPONSORING = -6  // account is sponsoring too many entries
};
```

OperationResultTr

class stellar_sdk.xdr.operation_result_tr.**OperationResultTr**(*type*, *create_account_result=None*, *payment_result=None*, *path_payment_strict_receive_result=None*, *manage_sell_offer_result=None*, *create_passive_sell_offer_result=None*, *set_options_result=None*, *change_trust_result=None*, *allow_trust_result=None*, *account_merge_result=None*, *inflation_result=None*, *manage_data_result=None*, *bump_seq_result=None*, *manage_buy_offer_result=None*, *path_payment_strict_send_result=None*, *create_claimable_balance_result=None*, *claim_claimable_balance_result=None*, *begin_sponsoring_future_reserves_result=None*, *end_sponsoring_future_reserves_result=None*, *revoke_sponsorship_result=None*, *clawback_result=None*, *clawback_claimable_balance_result=None*, *set_trust_line_flags_result=None*, *liquidity_pool_deposit_result=None*, *liquidity_pool_withdraw_result=None*, *invoke_host_function_result=None*, *extend_footprint_ttl_result=None*, *restore_footprint_result=None*)

XDR Source Code:

```
union switch (OperationType type)
    {
    case CREATE_ACCOUNT:
        CreateAccountResult createAccountResult;
    case PAYMENT:
        PaymentResult paymentResult;
    case PATH_PAYMENT_STRICT_RECEIVE:
        PathPaymentStrictReceiveResult pathPaymentStrictReceiveResult;
    case MANAGE_SELL_OFFER:
        ManageSellOfferResult manageSellOfferResult;
    case CREATE_PASSIVE_SELL_OFFER:
        ManageSellOfferResult createPassiveSellOfferResult;
    case SET_OPTIONS:
        SetOptionsResult setOptionsResult;
    case CHANGE_TRUST:
        ChangeTrustResult changeTrustResult;
    case ALLOW_TRUST:
        AllowTrustResult allowTrustResult;
    case ACCOUNT_MERGE:
        AccountMergeResult accountMergeResult;
    case INFLATION:
```

(continues on next page)

(continued from previous page)

```
        InflationResult inflationResult;
    case MANAGE_DATA:
        ManageDataResult manageDataResult;
    case BUMP_SEQUENCE:
        BumpSequenceResult bumpSeqResult;
    case MANAGE_BUY_OFFER:
        ManageBuyOfferResult manageBuyOfferResult;
    case PATH_PAYMENT_STRICT_SEND:
        PathPaymentStrictSendResult pathPaymentStrictSendResult;
    case CREATE_CLAIMABLE_BALANCE:
        CreateClaimableBalanceResult createClaimableBalanceResult;
    case CLAIM_CLAIMABLE_BALANCE:
        ClaimClaimableBalanceResult claimClaimableBalanceResult;
    case BEGIN_SPONSORING_FUTURE_RESERVES:
        BeginSponsoringFutureReservesResult beginSponsoringFutureReservesResult;
    case END_SPONSORING_FUTURE_RESERVES:
        EndSponsoringFutureReservesResult endSponsoringFutureReservesResult;
    case REVOKE_SPONSORSHIP:
        RevokeSponsorshipResult revokeSponsorshipResult;
    case CLAWBACK:
        ClawbackResult clawbackResult;
    case CLAWBACK_CLAIMABLE_BALANCE:
        ClawbackClaimableBalanceResult clawbackClaimableBalanceResult;
    case SET_TRUST_LINE_FLAGS:
        SetTrustLineFlagsResult setTrustLineFlagsResult;
    case LIQUIDITY_POOL_DEPOSIT:
        LiquidityPoolDepositResult liquidityPoolDepositResult;
    case LIQUIDITY_POOL_WITHDRAW:
        LiquidityPoolWithdrawResult liquidityPoolWithdrawResult;
    case INVOKE_HOST_FUNCTION:
        InvokeHostFunctionResult invokeHostFunctionResult;
    case EXTEND_FOOTPRINT_TTL:
        ExtendFootprintTTLResult extendFootprintTTLResult;
    case RESTORE_FOOTPRINT:
        RestoreFootprintResult restoreFootprintResult;
    }
```

OperationType

class stellar_sdk.xdr.operation_type.**OperationType**(**values*)

XDR Source Code:

```
enum OperationType
{
    CREATE_ACCOUNT = 0,
    PAYMENT = 1,
    PATH_PAYMENT_STRICT_RECEIVE = 2,
    MANAGE_SELL_OFFER = 3,
    CREATE_PASSIVE_SELL_OFFER = 4,
    SET_OPTIONS = 5,
    CHANGE_TRUST = 6,
    ALLOW_TRUST = 7,
```

(continues on next page)

(continued from previous page)

```
    ACCOUNT_MERGE = 8,
    INFLATION = 9,
    MANAGE_DATA = 10,
    BUMP_SEQUENCE = 11,
    MANAGE_BUY_OFFER = 12,
    PATH_PAYMENT_STRICT_SEND = 13,
    CREATE_CLAIMABLE_BALANCE = 14,
    CLAIM_CLAIMABLE_BALANCE = 15,
    BEGIN_SPONSORING_FUTURE_RESERVES = 16,
    END_SPONSORING_FUTURE_RESERVES = 17,
    REVOKE_SPONSORSHIP = 18,
    CLAWBACK = 19,
    CLAWBACK_CLAIMABLE_BALANCE = 20,
    SET_TRUST_LINE_FLAGS = 21,
    LIQUIDITY_POOL_DEPOSIT = 22,
    LIQUIDITY_POOL_WITHDRAW = 23,
    INVOKE_HOST_FUNCTION = 24,
    EXTEND_FOOTPRINT_TTL = 25,
    RESTORE_FOOTPRINT = 26
};
```

ParallelTxExecutionStage

class stellar_sdk.xdr.parallel_tx_execution_stage.**ParallelTxExecutionStage**(*parallel_tx_execution_stage*)

XDR Source Code:

```
typedef DependentTxCluster ParallelTxExecutionStage<>;
```

ParallelTxsComponent

class stellar_sdk.xdr.parallel_txs_component.**ParallelTxsComponent**(*base_fee*, *execution_stages*)

XDR Source Code:

```
struct ParallelTxsComponent
{
  int64* baseFee;
  // A sequence of stages that *may* have arbitrary data dependencies between
  // each other, i.e. in a general case the stage execution order may not be
  // arbitrarily shuffled without affecting the end result.
  ParallelTxExecutionStage executionStages<>;
};
```

PathPaymentStrictReceiveOp

class stellar_sdk.xdr.path_payment_strict_receive_op.**PathPaymentStrictReceiveOp**(*send_asset*, *send_max*, *destination*, *dest_asset*, *dest_amount*, *path*)

XDR Source Code:

```
struct PathPaymentStrictReceiveOp
{
    Asset sendAsset; // asset we pay with
    int64 sendMax;   // the maximum amount of sendAsset to
                     // send (excluding fees).
                     // The operation will fail if can't be met

    MuxedAccount destination; // recipient of the payment
    Asset destAsset;          // what they end up with
    int64 destAmount;         // amount they end up with

    Asset path<5>; // additional hops it must go through to get there
};
```

PathPaymentStrictReceiveResult

class stellar_sdk.xdr.path_payment_strict_receive_result.**PathPaymentStrictReceiveResult**(*code*, *success=None*, *no_issuer=None*)

XDR Source Code:

```
union PathPaymentStrictReceiveResult switch (
    PathPaymentStrictReceiveResultCode code)
{
case PATH_PAYMENT_STRICT_RECEIVE_SUCCESS:
    struct
    {
        ClaimAtom offers<>;
        SimplePaymentResult last;
    } success;
case PATH_PAYMENT_STRICT_RECEIVE_MALFORMED:
case PATH_PAYMENT_STRICT_RECEIVE_UNDERFUNDED:
case PATH_PAYMENT_STRICT_RECEIVE_SRC_NO_TRUST:
case PATH_PAYMENT_STRICT_RECEIVE_SRC_NOT_AUTHORIZED:
case PATH_PAYMENT_STRICT_RECEIVE_NO_DESTINATION:
case PATH_PAYMENT_STRICT_RECEIVE_NO_TRUST:
case PATH_PAYMENT_STRICT_RECEIVE_NOT_AUTHORIZED:
case PATH_PAYMENT_STRICT_RECEIVE_LINE_FULL:
    void;
case PATH_PAYMENT_STRICT_RECEIVE_NO_ISSUER:
    Asset noIssuer; // the asset that caused the error
case PATH_PAYMENT_STRICT_RECEIVE_TOO_FEW_OFFERS:
case PATH_PAYMENT_STRICT_RECEIVE_OFFER_CROSS_SELF:
case PATH_PAYMENT_STRICT_RECEIVE_OVER_SENDMAX:
    void;
};
```

PathPaymentStrictReceiveResultCode

class stellar_sdk.xdr.path_payment_strict_receive_result_code.**PathPaymentStrictReceiveResultCode**(**values*)

XDR Source Code:

```
enum PathPaymentStrictReceiveResultCode
{
    // codes considered as "success" for the operation
    PATH_PAYMENT_STRICT_RECEIVE_SUCCESS = 0, // success

    // codes considered as "failure" for the operation
    PATH_PAYMENT_STRICT_RECEIVE_MALFORMED = -1, // bad input
    PATH_PAYMENT_STRICT_RECEIVE_UNDERFUNDED =
        -2, // not enough funds in source account
    PATH_PAYMENT_STRICT_RECEIVE_SRC_NO_TRUST =
        -3, // no trust line on source account
    PATH_PAYMENT_STRICT_RECEIVE_SRC_NOT_AUTHORIZED =
        -4, // source not authorized to transfer
    PATH_PAYMENT_STRICT_RECEIVE_NO_DESTINATION =
        -5, // destination account does not exist
    PATH_PAYMENT_STRICT_RECEIVE_NO_TRUST =
        -6, // dest missing a trust line for asset
    PATH_PAYMENT_STRICT_RECEIVE_NOT_AUTHORIZED =
        -7, // dest not authorized to hold asset
    PATH_PAYMENT_STRICT_RECEIVE_LINE_FULL =
        -8, // dest would go above their limit
    PATH_PAYMENT_STRICT_RECEIVE_NO_ISSUER = -9, // missing issuer on one asset
    PATH_PAYMENT_STRICT_RECEIVE_TOO_FEW_OFFERS =
        -10, // not enough offers to satisfy path
    PATH_PAYMENT_STRICT_RECEIVE_OFFER_CROSS_SELF =
        -11, // would cross one of its own offers
    PATH_PAYMENT_STRICT_RECEIVE_OVER_SENDMAX = -12 // could not satisfy sendmax
};
```

PathPaymentStrictReceiveResultSuccess

class stellar_sdk.xdr.path_payment_strict_receive_result_success.**PathPaymentStrictReceiveResultSuccess**(*

XDR Source Code:

```
struct
    {
        ClaimAtom offers<>;
        SimplePaymentResult last;
    }
```

PathPaymentStrictSendOp

class stellar_sdk.xdr.path_payment_strict_send_op.**PathPaymentStrictSendOp**(*send_asset*, *send_amount*, *destination*, *dest_asset*, *dest_min*, *path*)

XDR Source Code:

```
struct PathPaymentStrictSendOp
{
    Asset sendAsset;  // asset we pay with
    int64 sendAmount; // amount of sendAsset to send (excluding fees)

    MuxedAccount destination; // recipient of the payment
    Asset destAsset;          // what they end up with
    int64 destMin;            // the minimum amount of dest asset to
                              // be received
                              // The operation will fail if it can't be met

    Asset path<5>; // additional hops it must go through to get there
};
```

PathPaymentStrictSendResult

class stellar_sdk.xdr.path_payment_strict_send_result.**PathPaymentStrictSendResult**(*code*, *success=None*, *no_issuer=None*)

XDR Source Code:

```
union PathPaymentStrictSendResult switch (PathPaymentStrictSendResultCode code)
{
case PATH_PAYMENT_STRICT_SEND_SUCCESS:
    struct
    {
        ClaimAtom offers<>;
        SimplePaymentResult last;
    } success;
case PATH_PAYMENT_STRICT_SEND_MALFORMED:
case PATH_PAYMENT_STRICT_SEND_UNDERFUNDED:
case PATH_PAYMENT_STRICT_SEND_SRC_NO_TRUST:
case PATH_PAYMENT_STRICT_SEND_SRC_NOT_AUTHORIZED:
case PATH_PAYMENT_STRICT_SEND_NO_DESTINATION:
case PATH_PAYMENT_STRICT_SEND_NO_TRUST:
case PATH_PAYMENT_STRICT_SEND_NOT_AUTHORIZED:
case PATH_PAYMENT_STRICT_SEND_LINE_FULL:
    void;
case PATH_PAYMENT_STRICT_SEND_NO_ISSUER:
    Asset noIssuer; // the asset that caused the error
case PATH_PAYMENT_STRICT_SEND_TOO_FEW_OFFERS:
case PATH_PAYMENT_STRICT_SEND_OFFER_CROSS_SELF:
case PATH_PAYMENT_STRICT_SEND_UNDER_DESTMIN:
    void;
};
```

PathPaymentStrictSendResultCode

class stellar_sdk.xdr.path_payment_strict_send_result_code.**PathPaymentStrictSendResultCode**(**values*)

XDR Source Code:

```
enum PathPaymentStrictSendResultCode
{
    // codes considered as "success" for the operation
    PATH_PAYMENT_STRICT_SEND_SUCCESS = 0, // success

    // codes considered as "failure" for the operation
    PATH_PAYMENT_STRICT_SEND_MALFORMED = -1, // bad input
    PATH_PAYMENT_STRICT_SEND_UNDERFUNDED =
        -2, // not enough funds in source account
    PATH_PAYMENT_STRICT_SEND_SRC_NO_TRUST =
        -3, // no trust line on source account
    PATH_PAYMENT_STRICT_SEND_SRC_NOT_AUTHORIZED =
        -4, // source not authorized to transfer
    PATH_PAYMENT_STRICT_SEND_NO_DESTINATION =
        -5, // destination account does not exist
    PATH_PAYMENT_STRICT_SEND_NO_TRUST =
        -6, // dest missing a trust line for asset
    PATH_PAYMENT_STRICT_SEND_NOT_AUTHORIZED =
        -7, // dest not authorized to hold asset
    PATH_PAYMENT_STRICT_SEND_LINE_FULL = -8, // dest would go above their limit
    PATH_PAYMENT_STRICT_SEND_NO_ISSUER = -9, // missing issuer on one asset
    PATH_PAYMENT_STRICT_SEND_TOO_FEW_OFFERS =
        -10, // not enough offers to satisfy path
    PATH_PAYMENT_STRICT_SEND_OFFER_CROSS_SELF =
        -11, // would cross one of its own offers
    PATH_PAYMENT_STRICT_SEND_UNDER_DESTMIN = -12 // could not satisfy destMin
};
```

PathPaymentStrictSendResultSuccess

class stellar_sdk.xdr.path_payment_strict_send_result_success.**PathPaymentStrictSendResultSuccess**(*offers*, *last*)

XDR Source Code:

```
struct
    {
        ClaimAtom offers<>;
        SimplePaymentResult last;
    }
```

PaymentOp

class stellar_sdk.xdr.payment_op.**PaymentOp**(*destination*, *asset*, *amount*)

XDR Source Code:

```
struct PaymentOp
{
    MuxedAccount destination; // recipient of the payment
    Asset asset;              // what they end up with
    int64 amount;             // amount they end up with
};
```

PaymentResult

class stellar_sdk.xdr.payment_result.**PaymentResult**(*code*)

XDR Source Code:

```
union PaymentResult switch (PaymentResultCode code)
{
case PAYMENT_SUCCESS:
    void;
case PAYMENT_MALFORMED:
case PAYMENT_UNDERFUNDED:
case PAYMENT_SRC_NO_TRUST:
case PAYMENT_SRC_NOT_AUTHORIZED:
case PAYMENT_NO_DESTINATION:
case PAYMENT_NO_TRUST:
case PAYMENT_NOT_AUTHORIZED:
case PAYMENT_LINE_FULL:
case PAYMENT_NO_ISSUER:
    void;
};
```

PaymentResultCode

class stellar_sdk.xdr.payment_result_code.**PaymentResultCode**(**values*)

XDR Source Code:

```
enum PaymentResultCode
{
    // codes considered as "success" for the operation
    PAYMENT_SUCCESS = 0, // payment successfully completed

    // codes considered as "failure" for the operation
    PAYMENT_MALFORMED = -1,          // bad input
    PAYMENT_UNDERFUNDED = -2,        // not enough funds in source account
    PAYMENT_SRC_NO_TRUST = -3,       // no trust line on source account
    PAYMENT_SRC_NOT_AUTHORIZED = -4, // source not authorized to transfer
    PAYMENT_NO_DESTINATION = -5,     // destination account does not exist
    PAYMENT_NO_TRUST = -6,       // destination missing a trust line for asset
    PAYMENT_NOT_AUTHORIZED = -7, // destination not authorized to hold asset
    PAYMENT_LINE_FULL = -8,      // destination would go above their limit
    PAYMENT_NO_ISSUER = -9       // missing issuer on asset
};
```

PeerAddress

class stellar_sdk.xdr.peer_address.**PeerAddress**(*ip*, *port*, *num_failures*)

XDR Source Code:

```
struct PeerAddress
{
    union switch (IPAddrType type)
    {
    case IPv4:
```

(continues on next page)

(continued from previous page)

```
        opaque ipv4[4];
    case IPv6:
        opaque ipv6[16];
    }
    ip;
    uint32 port;
    uint32 numFailures;
};
```

PeerAddressIp

class stellar_sdk.xdr.peer_address_ip.**PeerAddressIp**(*type*, *ipv4=None*, *ipv6=None*)

XDR Source Code:

```
union switch (IPAddrType type)
    {
    case IPv4:
        opaque ipv4[4];
    case IPv6:
        opaque ipv6[16];
    }
```

PeerStats

class stellar_sdk.xdr.peer_stats.**PeerStats**(*id*, *version_str*, *messages_read*, *messages_written*, *bytes_read*, *bytes_written*, *seconds_connected*, *unique_flood_bytes_recv*, *duplicate_flood_bytes_recv*, *unique_fetch_bytes_recv*, *duplicate_fetch_bytes_recv*, *unique_flood_message_recv*, *duplicate_flood_message_recv*, *unique_fetch_message_recv*, *duplicate_fetch_message_recv*)

XDR Source Code:

```
struct PeerStats
{
    NodeID id;
    string versionStr<100>;
    uint64 messagesRead;
    uint64 messagesWritten;
    uint64 bytesRead;
    uint64 bytesWritten;
    uint64 secondsConnected;

    uint64 uniqueFloodBytesRecv;
    uint64 duplicateFloodBytesRecv;
    uint64 uniqueFetchBytesRecv;
    uint64 duplicateFetchBytesRecv;

    uint64 uniqueFloodMessageRecv;
    uint64 duplicateFloodMessageRecv;
    uint64 uniqueFetchMessageRecv;
    uint64 duplicateFetchMessageRecv;
};
```

PersistedSCPState

class stellar_sdk.xdr.persisted_scp_state.**PersistedSCPState**(*v*, *v0=None*, *v1=None*)

XDR Source Code:

```
union PersistedSCPState switch (int v)
{
case 0:
        PersistedSCPStateV0 v0;
case 1:
        PersistedSCPStateV1 v1;
};
```

PersistedSCPStateV0

class stellar_sdk.xdr.persisted_scp_state_v0.**PersistedSCPStateV0**(*scp_envelopes*, *quorum_sets*, *tx_sets*)

XDR Source Code:

```
struct PersistedSCPStateV0
{
        SCPEnvelope scpEnvelopes<>;
        SCPQuorumSet quorumSets<>;
        StoredTransactionSet txSets<>;
};
```

PersistedSCPStateV1

class stellar_sdk.xdr.persisted_scp_state_v1.**PersistedSCPStateV1**(*scp_envelopes*, *quorum_sets*)

XDR Source Code:

```
struct PersistedSCPStateV1
{
        // Tx sets are saved separately
        SCPEnvelope scpEnvelopes<>;
        SCPQuorumSet quorumSets<>;
};
```

PoolID

class stellar_sdk.xdr.pool_id.**PoolID**(*pool_id*)

XDR Source Code:

```
typedef Hash PoolID;
```

PreconditionType

class stellar_sdk.xdr.precondition_type.**PreconditionType**(**values*)

XDR Source Code:

```
enum PreconditionType
{
    PRECOND_NONE = 0,
```

(continues on next page)

(continued from previous page)

```
    PRECOND_TIME = 1,
    PRECOND_V2 = 2
};
```

Preconditions

class stellar_sdk.xdr.preconditions.**Preconditions**(*type*, *time_bounds=None*, *v2=None*)

XDR Source Code:

```
union Preconditions switch (PreconditionType type)
{
case PRECOND_NONE:
    void;
case PRECOND_TIME:
    TimeBounds timeBounds;
case PRECOND_V2:
    PreconditionsV2 v2;
};
```

PreconditionsV2

class stellar_sdk.xdr.preconditions_v2.**PreconditionsV2**(*time_bounds*, *ledger_bounds*, *min_seq_num*, *min_seq_age*, *min_seq_ledger_gap*, *extra_signers*)

XDR Source Code:

```
struct PreconditionsV2
{
    TimeBounds* timeBounds;

    // Transaction only valid for ledger numbers n such that
    // minLedger <= n < maxLedger (if maxLedger == 0, then
    // only minLedger is checked)
    LedgerBounds* ledgerBounds;

    // If NULL, only valid when sourceAccount's sequence number
    // is seqNum - 1.  Otherwise, valid when sourceAccount's
    // sequence number n satisfies minSeqNum <= n < tx.seqNum.
    // Note that after execution the account's sequence number
    // is always raised to tx.seqNum, and a transaction is not
    // valid if tx.seqNum is too high to ensure replay protection.
    SequenceNumber* minSeqNum;

    // For the transaction to be valid, the current ledger time must
    // be at least minSeqAge greater than sourceAccount's seqTime.
    Duration minSeqAge;

    // For the transaction to be valid, the current ledger number
    // must be at least minSeqLedgerGap greater than sourceAccount's
    // seqLedger.
    uint32 minSeqLedgerGap;

```

(continues on next page)

(continued from previous page)

```
    // For the transaction to be valid, there must be a signature
    // corresponding to every Signer in this array, even if the
    // signature is not otherwise required by the sourceAccount or
    // operations.
    SignerKey extraSigners<2>;
};
```

Price

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

XDR Source Code:

```
struct Price
{
    int32 n; // numerator
    int32 d; // denominator
};
```

PublicKey

class stellar_sdk.xdr.public_key.**PublicKey**(*type*, *ed25519=None*)

XDR Source Code:

```
union PublicKey switch (PublicKeyType type)
{
case PUBLIC_KEY_TYPE_ED25519:
    uint256 ed25519;
};
```

PublicKeyType

class stellar_sdk.xdr.public_key_type.**PublicKeyType**(**values*)

XDR Source Code:

```
enum PublicKeyType
{
    PUBLIC_KEY_TYPE_ED25519 = KEY_TYPE_ED25519
};
```

RestoreFootprintOp

class stellar_sdk.xdr.restore_footprint_op.**RestoreFootprintOp**(*ext*)

XDR Source Code:

```
struct RestoreFootprintOp
{
    ExtensionPoint ext;
};
```

RestoreFootprintResult

class stellar_sdk.xdr.restore_footprint_result.**RestoreFootprintResult**(*code*)

XDR Source Code:

```
union RestoreFootprintResult switch (RestoreFootprintResultCode code)
{
case RESTORE_FOOTPRINT_SUCCESS:
    void;
case RESTORE_FOOTPRINT_MALFORMED:
case RESTORE_FOOTPRINT_RESOURCE_LIMIT_EXCEEDED:
case RESTORE_FOOTPRINT_INSUFFICIENT_REFUNDABLE_FEE:
    void;
};
```

RestoreFootprintResultCode

class stellar_sdk.xdr.restore_footprint_result_code.**RestoreFootprintResultCode**(**values*)

XDR Source Code:

```
enum RestoreFootprintResultCode
{
    // codes considered as "success" for the operation
    RESTORE_FOOTPRINT_SUCCESS = 0,

    // codes considered as "failure" for the operation
    RESTORE_FOOTPRINT_MALFORMED = -1,
    RESTORE_FOOTPRINT_RESOURCE_LIMIT_EXCEEDED = -2,
    RESTORE_FOOTPRINT_INSUFFICIENT_REFUNDABLE_FEE = -3
};
```

RevokeSponsorshipOp

class stellar_sdk.xdr.revoke_sponsorship_op.**RevokeSponsorshipOp**(*type*, *ledger_key=None*, *signer=None*)

XDR Source Code:

```
union RevokeSponsorshipOp switch (RevokeSponsorshipType type)
{
case REVOKE_SPONSORSHIP_LEDGER_ENTRY:
    LedgerKey ledgerKey;
case REVOKE_SPONSORSHIP_SIGNER:
    struct
    {
        AccountID accountID;
        SignerKey signerKey;
    } signer;
};
```

RevokeSponsorshipOpSigner

class stellar_sdk.xdr.revoke_sponsorship_op_signer.**RevokeSponsorshipOpSigner**(*account_id*, *signer_key*)

XDR Source Code:

```
struct
    {
        AccountID accountID;
        SignerKey signerKey;
    }
```

RevokeSponsorshipResult

class stellar_sdk.xdr.revoke_sponsorship_result.**RevokeSponsorshipResult**(*code*)

XDR Source Code:

```
union RevokeSponsorshipResult switch (RevokeSponsorshipResultCode code)
{
case REVOKE_SPONSORSHIP_SUCCESS:
    void;
case REVOKE_SPONSORSHIP_DOES_NOT_EXIST:
case REVOKE_SPONSORSHIP_NOT_SPONSOR:
case REVOKE_SPONSORSHIP_LOW_RESERVE:
case REVOKE_SPONSORSHIP_ONLY_TRANSFERABLE:
case REVOKE_SPONSORSHIP_MALFORMED:
    void;
};
```

RevokeSponsorshipResultCode

class stellar_sdk.xdr.revoke_sponsorship_result_code.**RevokeSponsorshipResultCode**(**values*)

XDR Source Code:

```
enum RevokeSponsorshipResultCode
{
    // codes considered as "success" for the operation
    REVOKE_SPONSORSHIP_SUCCESS = 0,

    // codes considered as "failure" for the operation
    REVOKE_SPONSORSHIP_DOES_NOT_EXIST = -1,
    REVOKE_SPONSORSHIP_NOT_SPONSOR = -2,
    REVOKE_SPONSORSHIP_LOW_RESERVE = -3,
    REVOKE_SPONSORSHIP_ONLY_TRANSFERABLE = -4,
    REVOKE_SPONSORSHIP_MALFORMED = -5
};
```

RevokeSponsorshipType

class stellar_sdk.xdr.revoke_sponsorship_type.**RevokeSponsorshipType**(**values*)

XDR Source Code:

```
enum RevokeSponsorshipType
{
    REVOKE_SPONSORSHIP_LEDGER_ENTRY = 0,
    REVOKE_SPONSORSHIP_SIGNER = 1
};
```

SCAddress

class stellar_sdk.xdr.sc_address.**SCAddress**(*type*, *account_id=None*, *contract_id=None*, *muxed_account=None*, *claimable_balance_id=None*, *liquidity_pool_id=None*)

XDR Source Code:

```
union SCAddress switch (SCAddressType type)
{
case SC_ADDRESS_TYPE_ACCOUNT:
    AccountID accountId;
case SC_ADDRESS_TYPE_CONTRACT:
    ContractID contractId;
case SC_ADDRESS_TYPE_MUXED_ACCOUNT:
    MuxedEd25519Account muxedAccount;
case SC_ADDRESS_TYPE_CLAIMABLE_BALANCE:
    ClaimableBalanceID claimableBalanceId;
case SC_ADDRESS_TYPE_LIQUIDITY_POOL:
    PoolID liquidityPoolId;
};
```

SCAddressType

class stellar_sdk.xdr.sc_address_type.**SCAddressType**(**values*)

XDR Source Code:

```
enum SCAddressType
{
    SC_ADDRESS_TYPE_ACCOUNT = 0,
    SC_ADDRESS_TYPE_CONTRACT = 1,
    SC_ADDRESS_TYPE_MUXED_ACCOUNT = 2,
    SC_ADDRESS_TYPE_CLAIMABLE_BALANCE = 3,
    SC_ADDRESS_TYPE_LIQUIDITY_POOL = 4
};
```

SCBytes

class stellar_sdk.xdr.sc_bytes.**SCBytes**(*sc_bytes*)

XDR Source Code:

```
typedef opaque SCBytes<>;
```

SCContractInstance

class stellar_sdk.xdr.sc_contract_instance.**SCContractInstance**(*executable*, *storage*)

XDR Source Code:

```
struct SCContractInstance {
    ContractExecutable executable;
    SCMap* storage;
};
```

SCEnvMetaEntry

class stellar_sdk.xdr.sc_env_meta_entry.**SCEnvMetaEntry**(*kind*, *interface_version=None*)

XDR Source Code:

```
union SCEnvMetaEntry switch (SCEnvMetaKind kind)
{
case SC_ENV_META_KIND_INTERFACE_VERSION:
    struct {
        uint32 protocol;
        uint32 preRelease;
    } interfaceVersion;
};
```

SCEnvMetaEntryInterfaceVersion

class stellar_sdk.xdr.sc_env_meta_entry_interface_version.**SCEnvMetaEntryInterfaceVersion**(*protocol*, *pre_release*)

XDR Source Code:

```
struct {
        uint32 protocol;
        uint32 preRelease;
    }
```

SCEnvMetaKind

class stellar_sdk.xdr.sc_env_meta_kind.**SCEnvMetaKind**(**values*)

XDR Source Code:

```
enum SCEnvMetaKind
{
    SC_ENV_META_KIND_INTERFACE_VERSION = 0
};
```

SCError

class stellar_sdk.xdr.sc_error.**SCError**(*type*, *contract_code=None*, *code=None*)

XDR Source Code:

```
union SCError switch (SCErrorType type)
{
case SCE_CONTRACT:
```

(continues on next page)

(continued from previous page)

```
    uint32 contractCode;
case SCE_WASM_VM:
case SCE_CONTEXT:
case SCE_STORAGE:
case SCE_OBJECT:
case SCE_CRYPTO:
case SCE_EVENTS:
case SCE_BUDGET:
case SCE_VALUE:
case SCE_AUTH:
    SCErrorCode code;
};
```

SCErrorCode

class stellar_sdk.xdr.sc_error_code.**SCErrorCode**(**values*)

XDR Source Code:

```
enum SCErrorCode
{
    SCEC_ARITH_DOMAIN = 0,      // Some arithmetic was undefined (overflow, divide-
↪by-zero).
    SCEC_INDEX_BOUNDS = 1,      // Something was indexed beyond its bounds.
    SCEC_INVALID_INPUT = 2,     // User provided some otherwise-bad data.
    SCEC_MISSING_VALUE = 3,     // Some value was required but not provided.
    SCEC_EXISTING_VALUE = 4,    // Some value was provided where not allowed.
    SCEC_EXCEEDED_LIMIT = 5,    // Some arbitrary limit -- gas or otherwise -- was␣
↪hit.
    SCEC_INVALID_ACTION = 6,    // Data was valid but action requested was not.
    SCEC_INTERNAL_ERROR = 7,    // The host detected an error in its own logic.
    SCEC_UNEXPECTED_TYPE = 8,   // Some type wasn't as expected.
    SCEC_UNEXPECTED_SIZE = 9    // Something's size wasn't as expected.
};
```

SCErrorType

class stellar_sdk.xdr.sc_error_type.**SCErrorType**(**values*)

XDR Source Code:

```
enum SCErrorType
{
    SCE_CONTRACT = 0,          // Contract-specific, user-defined codes.
    SCE_WASM_VM = 1,           // Errors while interpreting WASM bytecode.
    SCE_CONTEXT = 2,           // Errors in the contract's host context.
    SCE_STORAGE = 3,           // Errors accessing host storage.
    SCE_OBJECT = 4,            // Errors working with host objects.
    SCE_CRYPTO = 5,            // Errors in cryptographic operations.
    SCE_EVENTS = 6,            // Errors while emitting events.
    SCE_BUDGET = 7,            // Errors relating to budget limits.
    SCE_VALUE = 8,             // Errors working with host values or SCVals.
    SCE_AUTH = 9               // Errors from the authentication subsystem.
};
```

SCMap

class stellar_sdk.xdr.sc_map.**SCMap**(*sc_map*)

XDR Source Code:

```
typedef SCMapEntry SCMap<>;
```

SCMapEntry

class stellar_sdk.xdr.sc_map_entry.**SCMapEntry**(*key*, *val*)

XDR Source Code:

```
struct SCMapEntry
{
    SCVal key;
    SCVal val;
};
```

SCMetaEntry

class stellar_sdk.xdr.sc_meta_entry.**SCMetaEntry**(*kind*, *v0=None*)

XDR Source Code:

```
union SCMetaEntry switch (SCMetaKind kind)
{
case SC_META_V0:
    SCMetaV0 v0;
};
```

SCMetaKind

class stellar_sdk.xdr.sc_meta_kind.**SCMetaKind**(**values*)

XDR Source Code:

```
enum SCMetaKind
{
    SC_META_V0 = 0
};
```

SCMetaV0

class stellar_sdk.xdr.sc_meta_v0.**SCMetaV0**(*key*, *val*)

XDR Source Code:

```
struct SCMetaV0
{
    string key<>;
    string val<>;
};
```

SCNonceKey

class stellar_sdk.xdr.sc_nonce_key.**SCNonceKey**(*nonce*)

XDR Source Code:

```
struct SCNonceKey {
    int64 nonce;
};
```

SCPBallot

class stellar_sdk.xdr.scp_ballot.**SCPBallot**(*counter*, *value*)

XDR Source Code:

```
struct SCPBallot
{
    uint32 counter; // n
    Value value;    // x
};
```

SCPEnvelope

class stellar_sdk.xdr.scp_envelope.**SCPEnvelope**(*statement*, *signature*)

XDR Source Code:

```
struct SCPEnvelope
{
    SCPStatement statement;
    Signature signature;
};
```

SCPHistoryEntry

class stellar_sdk.xdr.scp_history_entry.**SCPHistoryEntry**(*v*, *v0=None*)

XDR Source Code:

```
union SCPHistoryEntry switch (int v)
{
case 0:
    SCPHistoryEntryV0 v0;
};
```

SCPHistoryEntryV0

class stellar_sdk.xdr.scp_history_entry_v0.**SCPHistoryEntryV0**(*quorum_sets*, *ledger_messages*)

XDR Source Code:

```
struct SCPHistoryEntryV0
{
    SCPQuorumSet quorumSets<>; // additional quorum sets used by ledgerMessages
    LedgerSCPMessages ledgerMessages;
};
```

SCPNomination

class stellar_sdk.xdr.scp_nomination.**SCPNomination**(*quorum_set_hash*, *votes*, *accepted*)

XDR Source Code:

```
struct SCPNomination
{
    Hash quorumSetHash; // D
    Value votes<>;      // X
    Value accepted<>;   // Y
};
```

SCPQuorumSet

class stellar_sdk.xdr.scp_quorum_set.**SCPQuorumSet**(*threshold*, *validators*, *inner_sets*)

XDR Source Code:

```
struct SCPQuorumSet
{
    uint32 threshold;
    NodeID validators<>;
    SCPQuorumSet innerSets<>;
};
```

SCPStatement

class stellar_sdk.xdr.scp_statement.**SCPStatement**(*node_id*, *slot_index*, *pledges*)

XDR Source Code:

```
struct SCPStatement
{
    NodeID nodeID;    // v
    uint64 slotIndex; // i

    union switch (SCPStatementType type)
    {
    case SCP_ST_PREPARE:
        struct
        {
            Hash quorumSetHash;       // D
            SCPBallot ballot;         // b
            SCPBallot* prepared;      // p
            SCPBallot* preparedPrime; // p'
            uint32 nC;                // c.n
            uint32 nH;                // h.n
        } prepare;
    case SCP_ST_CONFIRM:
        struct
        {
            SCPBallot ballot;   // b
            uint32 nPrepared;   // p.n
            uint32 nCommit;     // c.n
            uint32 nH;          // h.n
```

(continues on next page)

(continued from previous page)

```
            Hash quorumSetHash; // D
        } confirm;
    case SCP_ST_EXTERNALIZE:
        struct
        {
            SCPBallot commit;         // c
            uint32 nH;                // h.n
            Hash commitQuorumSetHash; // D used before EXTERNALIZE
        } externalize;
    case SCP_ST_NOMINATE:
        SCPNomination nominate;
    }
    pledges;
};
```

SCPStatementConfirm

class stellar_sdk.xdr.scp_statement_confirm.**SCPStatementConfirm**(*ballot*, *n_prepared*, *n_commit*, *n_h*, *quorum_set_hash*)

XDR Source Code:

```
struct
        {
            SCPBallot ballot;   // b
            uint32 nPrepared;   // p.n
            uint32 nCommit;     // c.n
            uint32 nH;          // h.n
            Hash quorumSetHash; // D
        }
```

SCPStatementExternalize

class stellar_sdk.xdr.scp_statement_externalize.**SCPStatementExternalize**(*commit*, *n_h*, *commit_quorum_set_hash*)

XDR Source Code:

```
struct
        {
            SCPBallot commit;         // c
            uint32 nH;                // h.n
            Hash commitQuorumSetHash; // D used before EXTERNALIZE
        }
```

SCPStatementPledges

class stellar_sdk.xdr.scp_statement_pledges.**SCPStatementPledges**(*type*, *prepare=None*, *confirm=None*, *externalize=None*, *nominate=None*)

XDR Source Code:

```
union switch (SCPStatementType type)
    {
    case SCP_ST_PREPARE:
        struct
        {
            Hash quorumSetHash;       // D
            SCPBallot ballot;         // b
            SCPBallot* prepared;      // p
            SCPBallot* preparedPrime; // p'
            uint32 nC;                // c.n
            uint32 nH;                // h.n
        } prepare;
    case SCP_ST_CONFIRM:
        struct
        {
            SCPBallot ballot;   // b
            uint32 nPrepared;   // p.n
            uint32 nCommit;     // c.n
            uint32 nH;          // h.n
            Hash quorumSetHash; // D
        } confirm;
    case SCP_ST_EXTERNALIZE:
        struct
        {
            SCPBallot commit;         // c
            uint32 nH;                // h.n
            Hash commitQuorumSetHash; // D used before EXTERNALIZE
        } externalize;
    case SCP_ST_NOMINATE:
        SCPNomination nominate;
    }
```

SCPStatementPrepare

class stellar_sdk.xdr.scp_statement_prepare.**SCPStatementPrepare**(*quorum_set_hash*, *ballot*, *prepared*, *prepared_prime*, *n_c*, *n_h*)

XDR Source Code:

```
struct
        {
            Hash quorumSetHash;       // D
            SCPBallot ballot;         // b
            SCPBallot* prepared;      // p
            SCPBallot* preparedPrime; // p'
            uint32 nC;                // c.n
            uint32 nH;                // h.n
        }
```

SCPStatementType

class stellar_sdk.xdr.scp_statement_type.**SCPStatementType**(**values*)

XDR Source Code:

```
enum SCPStatementType
{
    SCP_ST_PREPARE = 0,
    SCP_ST_CONFIRM = 1,
    SCP_ST_EXTERNALIZE = 2,
    SCP_ST_NOMINATE = 3
};
```

SCSpecEntry

class stellar_sdk.xdr.sc_spec_entry.**SCSpecEntry**(*kind*, *function_v0=None*, *udt_struct_v0=None*, *udt_union_v0=None*, *udt_enum_v0=None*, *udt_error_enum_v0=None*, *event_v0=None*)

XDR Source Code:

```
union SCSpecEntry switch (SCSpecEntryKind kind)
{
case SC_SPEC_ENTRY_FUNCTION_V0:
    SCSpecFunctionV0 functionV0;
case SC_SPEC_ENTRY_UDT_STRUCT_V0:
    SCSpecUDTStructV0 udtStructV0;
case SC_SPEC_ENTRY_UDT_UNION_V0:
    SCSpecUDTUnionV0 udtUnionV0;
case SC_SPEC_ENTRY_UDT_ENUM_V0:
    SCSpecUDTEnumV0 udtEnumV0;
case SC_SPEC_ENTRY_UDT_ERROR_ENUM_V0:
    SCSpecUDTErrorEnumV0 udtErrorEnumV0;
case SC_SPEC_ENTRY_EVENT_V0:
    SCSpecEventV0 eventV0;
};
```

SCSpecEntryKind

class stellar_sdk.xdr.sc_spec_entry_kind.**SCSpecEntryKind**(**values*)

XDR Source Code:

```
enum SCSpecEntryKind
{
    SC_SPEC_ENTRY_FUNCTION_V0 = 0,
    SC_SPEC_ENTRY_UDT_STRUCT_V0 = 1,
    SC_SPEC_ENTRY_UDT_UNION_V0 = 2,
    SC_SPEC_ENTRY_UDT_ENUM_V0 = 3,
    SC_SPEC_ENTRY_UDT_ERROR_ENUM_V0 = 4,
    SC_SPEC_ENTRY_EVENT_V0 = 5
};
```

SCSpecEventDataFormat

class stellar_sdk.xdr.sc_spec_event_data_format.**SCSpecEventDataFormat**(**values*)

XDR Source Code:

```
enum SCSpecEventDataFormat
{
    SC_SPEC_EVENT_DATA_FORMAT_SINGLE_VALUE = 0,
    SC_SPEC_EVENT_DATA_FORMAT_VEC = 1,
    SC_SPEC_EVENT_DATA_FORMAT_MAP = 2
};
```

SCSpecEventParamLocationV0

class stellar_sdk.xdr.sc_spec_event_param_location_v0.**SCSpecEventParamLocationV0**(**values*)

XDR Source Code:

```
enum SCSpecEventParamLocationV0
{
    SC_SPEC_EVENT_PARAM_LOCATION_DATA = 0,
    SC_SPEC_EVENT_PARAM_LOCATION_TOPIC_LIST = 1
};
```

SCSpecEventParamV0

class stellar_sdk.xdr.sc_spec_event_param_v0.**SCSpecEventParamV0**(*doc*, *name*, *type*, *location*)

XDR Source Code:

```
struct SCSpecEventParamV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<30>;
    SCSpecTypeDef type;
    SCSpecEventParamLocationV0 location;
};
```

SCSpecEventV0

class stellar_sdk.xdr.sc_spec_event_v0.**SCSpecEventV0**(*doc*, *lib*, *name*, *prefix_topics*, *params*, *data_format*)

XDR Source Code:

```
struct SCSpecEventV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string lib<80>;
    SCSymbol name;
    SCSymbol prefixTopics<2>;
    SCSpecEventParamV0 params<>;
    SCSpecEventDataFormat dataFormat;
};
```

SCSpecFunctionInputV0

class stellar_sdk.xdr.sc_spec_function_input_v0.**SCSpecFunctionInputV0**(*doc*, *name*, *type*)

XDR Source Code:

```
struct SCSpecFunctionInputV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<30>;
    SCSpecTypeDef type;
};
```

SCSpecFunctionV0

class stellar_sdk.xdr.sc_spec_function_v0.**SCSpecFunctionV0**(*doc*, *name*, *inputs*, *outputs*)

XDR Source Code:

```
struct SCSpecFunctionV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    SCSymbol name;
    SCSpecFunctionInputV0 inputs<>;
    SCSpecTypeDef outputs<1>;
};
```

SCSpecType

class stellar_sdk.xdr.sc_spec_type.**SCSpecType**(**values*)

XDR Source Code:

```
enum SCSpecType
{
    SC_SPEC_TYPE_VAL = 0,

    // Types with no parameters.
    SC_SPEC_TYPE_BOOL = 1,
    SC_SPEC_TYPE_VOID = 2,
    SC_SPEC_TYPE_ERROR = 3,
    SC_SPEC_TYPE_U32 = 4,
    SC_SPEC_TYPE_I32 = 5,
    SC_SPEC_TYPE_U64 = 6,
    SC_SPEC_TYPE_I64 = 7,
    SC_SPEC_TYPE_TIMEPOINT = 8,
    SC_SPEC_TYPE_DURATION = 9,
    SC_SPEC_TYPE_U128 = 10,
    SC_SPEC_TYPE_I128 = 11,
    SC_SPEC_TYPE_U256 = 12,
    SC_SPEC_TYPE_I256 = 13,
    SC_SPEC_TYPE_BYTES = 14,
    SC_SPEC_TYPE_STRING = 16,
    SC_SPEC_TYPE_SYMBOL = 17,
    SC_SPEC_TYPE_ADDRESS = 19,
    SC_SPEC_TYPE_MUXED_ADDRESS = 20,
```

(continues on next page)

(continued from previous page)

```
    // Types with parameters.
    SC_SPEC_TYPE_OPTION = 1000,
    SC_SPEC_TYPE_RESULT = 1001,
    SC_SPEC_TYPE_VEC = 1002,
    SC_SPEC_TYPE_MAP = 1004,
    SC_SPEC_TYPE_TUPLE = 1005,
    SC_SPEC_TYPE_BYTES_N = 1006,

    // User defined types.
    SC_SPEC_TYPE_UDT = 2000
};
```

SCSpecTypeBytesN

class stellar_sdk.xdr.sc_spec_type_bytes_n.**SCSpecTypeBytesN**(*n*)

XDR Source Code:

```
struct SCSpecTypeBytesN
{
    uint32 n;
};
```

SCSpecTypeDef

class stellar_sdk.xdr.sc_spec_type_def.**SCSpecTypeDef**(*type*, *option=None*, *result=None*, *vec=None*, *map=None*, *tuple=None*, *bytes_n=None*, *udt=None*)

XDR Source Code:

```
union SCSpecTypeDef switch (SCSpecType type)
{
case SC_SPEC_TYPE_VAL:
case SC_SPEC_TYPE_BOOL:
case SC_SPEC_TYPE_VOID:
case SC_SPEC_TYPE_ERROR:
case SC_SPEC_TYPE_U32:
case SC_SPEC_TYPE_I32:
case SC_SPEC_TYPE_U64:
case SC_SPEC_TYPE_I64:
case SC_SPEC_TYPE_TIMEPOINT:
case SC_SPEC_TYPE_DURATION:
case SC_SPEC_TYPE_U128:
case SC_SPEC_TYPE_I128:
case SC_SPEC_TYPE_U256:
case SC_SPEC_TYPE_I256:
case SC_SPEC_TYPE_BYTES:
case SC_SPEC_TYPE_STRING:
case SC_SPEC_TYPE_SYMBOL:
case SC_SPEC_TYPE_ADDRESS:
case SC_SPEC_TYPE_MUXED_ADDRESS:
    void;
```

(continues on next page)

(continued from previous page)

```
case SC_SPEC_TYPE_OPTION:
    SCSpecTypeOption option;
case SC_SPEC_TYPE_RESULT:
    SCSpecTypeResult result;
case SC_SPEC_TYPE_VEC:
    SCSpecTypeVec vec;
case SC_SPEC_TYPE_MAP:
    SCSpecTypeMap map;
case SC_SPEC_TYPE_TUPLE:
    SCSpecTypeTuple tuple;
case SC_SPEC_TYPE_BYTES_N:
    SCSpecTypeBytesN bytesN;
case SC_SPEC_TYPE_UDT:
    SCSpecTypeUDT udt;
};
```

SCSpecTypeMap

class stellar_sdk.xdr.sc_spec_type_map.**SCSpecTypeMap**(*key_type*, *value_type*)

XDR Source Code:

```
struct SCSpecTypeMap
{
    SCSpecTypeDef keyType;
    SCSpecTypeDef valueType;
};
```

SCSpecTypeOption

class stellar_sdk.xdr.sc_spec_type_option.**SCSpecTypeOption**(*value_type*)

XDR Source Code:

```
struct SCSpecTypeOption
{
    SCSpecTypeDef valueType;
};
```

SCSpecTypeResult

class stellar_sdk.xdr.sc_spec_type_result.**SCSpecTypeResult**(*ok_type*, *error_type*)

XDR Source Code:

```
struct SCSpecTypeResult
{
    SCSpecTypeDef okType;
    SCSpecTypeDef errorType;
};
```

SCSpecTypeTuple

class stellar_sdk.xdr.sc_spec_type_tuple.**SCSpecTypeTuple**(*value_types*)

XDR Source Code:

```
struct SCSpecTypeTuple
{
    SCSpecTypeDef valueTypes<12>;
};
```

SCSpecTypeUDT

class stellar_sdk.xdr.sc_spec_type_udt.**SCSpecTypeUDT**(*name*)

XDR Source Code:

```
struct SCSpecTypeUDT
{
    string name<60>;
};
```

SCSpecTypeVec

class stellar_sdk.xdr.sc_spec_type_vec.**SCSpecTypeVec**(*element_type*)

XDR Source Code:

```
struct SCSpecTypeVec
{
    SCSpecTypeDef elementType;
};
```

SCSpecUDTEnumCaseV0

class stellar_sdk.xdr.sc_spec_udt_enum_case_v0.**SCSpecUDTEnumCaseV0**(*doc*, *name*, *value*)

XDR Source Code:

```
struct SCSpecUDTEnumCaseV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<60>;
    uint32 value;
};
```

SCSpecUDTEnumV0

class stellar_sdk.xdr.sc_spec_udt_enum_v0.**SCSpecUDTEnumV0**(*doc*, *lib*, *name*, *cases*)

XDR Source Code:

```
struct SCSpecUDTEnumV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string lib<80>;
    string name<60>;
    SCSpecUDTEnumCaseV0 cases<>;
};
```

SCSpecUDTErrorEnumCaseV0

class stellar_sdk.xdr.sc_spec_udt_error_enum_case_v0.**SCSpecUDTErrorEnumCaseV0**(*doc*, *name*, *value*)

XDR Source Code:

```
struct SCSpecUDTErrorEnumCaseV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<60>;
    uint32 value;
};
```

SCSpecUDTErrorEnumV0

class stellar_sdk.xdr.sc_spec_udt_error_enum_v0.**SCSpecUDTErrorEnumV0**(*doc*, *lib*, *name*, *cases*)

XDR Source Code:

```
struct SCSpecUDTErrorEnumV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string lib<80>;
    string name<60>;
    SCSpecUDTErrorEnumCaseV0 cases<>;
};
```

SCSpecUDTStructFieldV0

class stellar_sdk.xdr.sc_spec_udt_struct_field_v0.**SCSpecUDTStructFieldV0**(*doc*, *name*, *type*)

XDR Source Code:

```
struct SCSpecUDTStructFieldV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<30>;
    SCSpecTypeDef type;
};
```

SCSpecUDTStructV0

class stellar_sdk.xdr.sc_spec_udt_struct_v0.**SCSpecUDTStructV0**(*doc*, *lib*, *name*, *fields*)

XDR Source Code:

```
struct SCSpecUDTStructV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string lib<80>;
    string name<60>;
    SCSpecUDTStructFieldV0 fields<>;
};
```

SCSpecUDTUnionCaseTupleV0

class stellar_sdk.xdr.sc_spec_udt_union_case_tuple_v0.**SCSpecUDTUnionCaseTupleV0**(*doc*, *name*, *type*)

XDR Source Code:

```
struct SCSpecUDTUnionCaseTupleV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<60>;
    SCSpecTypeDef type<>;
};
```

SCSpecUDTUnionCaseV0

class stellar_sdk.xdr.sc_spec_udt_union_case_v0.**SCSpecUDTUnionCaseV0**(*kind*, *void_case=None*, *tuple_case=None*)

XDR Source Code:

```
union SCSpecUDTUnionCaseV0 switch (SCSpecUDTUnionCaseV0Kind kind)
{
case SC_SPEC_UDT_UNION_CASE_VOID_V0:
    SCSpecUDTUnionCaseVoidV0 voidCase;
case SC_SPEC_UDT_UNION_CASE_TUPLE_V0:
    SCSpecUDTUnionCaseTupleV0 tupleCase;
};
```

SCSpecUDTUnionCaseV0Kind

class stellar_sdk.xdr.sc_spec_udt_union_case_v0_kind.**SCSpecUDTUnionCaseV0Kind**(**values*)

XDR Source Code:

```
enum SCSpecUDTUnionCaseV0Kind
{
    SC_SPEC_UDT_UNION_CASE_VOID_V0 = 0,
    SC_SPEC_UDT_UNION_CASE_TUPLE_V0 = 1
};
```

SCSpecUDTUnionCaseVoidV0

class stellar_sdk.xdr.sc_spec_udt_union_case_void_v0.**SCSpecUDTUnionCaseVoidV0**(*doc*, *name*)

XDR Source Code:

```
struct SCSpecUDTUnionCaseVoidV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string name<60>;
};
```

SCSpecUDTUnionV0

class stellar_sdk.xdr.sc_spec_udt_union_v0.**SCSpecUDTUnionV0**(*doc*, *lib*, *name*, *cases*)

XDR Source Code:

```
struct SCSpecUDTUnionV0
{
    string doc<SC_SPEC_DOC_LIMIT>;
    string lib<80>;
    string name<60>;
    SCSpecUDTUnionCaseV0 cases<>;
};
```

SCString

class stellar_sdk.xdr.sc_string.**SCString**(*sc_string*)

XDR Source Code:

```
typedef string SCString<>;
```

SCSymbol

class stellar_sdk.xdr.sc_symbol.**SCSymbol**(*sc_symbol*)

XDR Source Code:

```
typedef string SCSymbol<SCSYMBOL_LIMIT>;
```

SCVal

class stellar_sdk.xdr.sc_val.**SCVal**(*type*, *b=None*, *error=None*, *u32=None*, *i32=None*, *u64=None*, *i64=None*, *timepoint=None*, *duration=None*, *u128=None*, *i128=None*, *u256=None*, *i256=None*, *bytes=None*, *str=None*, *sym=None*, *vec=None*, *map=None*, *address=None*, *instance=None*, *nonce_key=None*)

XDR Source Code:

```
union SCVal switch (SCValType type)
{

case SCV_BOOL:
    bool b;
case SCV_VOID:
    void;
case SCV_ERROR:
    SCError error;

case SCV_U32:
    uint32 u32;
case SCV_I32:
    int32 i32;

case SCV_U64:
    uint64 u64;
```

(continues on next page)

(continued from previous page)

```
case SCV_I64:
    int64 i64;
case SCV_TIMEPOINT:
    TimePoint timepoint;
case SCV_DURATION:
    Duration duration;

case SCV_U128:
    UInt128Parts u128;
case SCV_I128:
    Int128Parts i128;

case SCV_U256:
    UInt256Parts u256;
case SCV_I256:
    Int256Parts i256;

case SCV_BYTES:
    SCBytes bytes;
case SCV_STRING:
    SCString str;
case SCV_SYMBOL:
    SCSymbol sym;

// Vec and Map are recursive so need to live
// behind an option, due to xdrpp limitations.
case SCV_VEC:
    SCVec *vec;
case SCV_MAP:
    SCMap *map;

case SCV_ADDRESS:
    SCAddress address;

// Special SCVals reserved for system-constructed contract-data
// ledger keys, not generally usable elsewhere.
case SCV_CONTRACT_INSTANCE:
    SCContractInstance instance;
case SCV_LEDGER_KEY_CONTRACT_INSTANCE:
    void;
case SCV_LEDGER_KEY_NONCE:
    SCNonceKey nonce_key;
};
```

SCValType

class stellar_sdk.xdr.sc_val_type.**SCValType**(**values*)

XDR Source Code:

```
enum SCValType
{
    SCV_BOOL = 0,
```

(continues on next page)

(continued from previous page)

```
    SCV_VOID = 1,
    SCV_ERROR = 2,

    // 32 bits is the smallest type in WASM or XDR; no need for u8/u16.
    SCV_U32 = 3,
    SCV_I32 = 4,

    // 64 bits is naturally supported by both WASM and XDR also.
    SCV_U64 = 5,
    SCV_I64 = 6,

    // Time-related u64 subtypes with their own functions and formatting.
    SCV_TIMEPOINT = 7,
    SCV_DURATION = 8,

    // 128 bits is naturally supported by Rust and we use it for Soroban
    // fixed-point arithmetic prices / balances / similar "quantities". These
    // are represented in XDR as a pair of 2 u64s.
    SCV_U128 = 9,
    SCV_I128 = 10,

    // 256 bits is the size of sha256 output, ed25519 keys, and the EVM machine
    // word, so for interop use we include this even though it requires a small
    // amount of Rust guest and/or host library code.
    SCV_U256 = 11,
    SCV_I256 = 12,

    // Bytes come in 3 flavors, 2 of which have meaningfully different
    // formatting and validity-checking / domain-restriction.
    SCV_BYTES = 13,
    SCV_STRING = 14,
    SCV_SYMBOL = 15,

    // Vecs and maps are just polymorphic containers of other ScVals.
    SCV_VEC = 16,
    SCV_MAP = 17,

    // Address is the universal identifier for contracts and classic
    // accounts.
    SCV_ADDRESS = 18,

    // The following are the internal SCVal variants that are not
    // exposed to the contracts.
    SCV_CONTRACT_INSTANCE = 19,

    // SCV_LEDGER_KEY_CONTRACT_INSTANCE and SCV_LEDGER_KEY_NONCE are unique
    // symbolic SCVals used as the key for ledger entries for a contract's
    // instance and an address' nonce, respectively.
    SCV_LEDGER_KEY_CONTRACT_INSTANCE = 20,
    SCV_LEDGER_KEY_NONCE = 21
};
```

SCVec

class stellar_sdk.xdr.sc_vec.**SCVec**(*sc_vec*)

XDR Source Code:

```
typedef SCVal SCVec<>;
```

SendMore

class stellar_sdk.xdr.send_more.**SendMore**(*num_messages*)

XDR Source Code:

```
struct SendMore
{
    uint32 numMessages;
};
```

SendMoreExtended

class stellar_sdk.xdr.send_more_extended.**SendMoreExtended**(*num_messages*, *num_bytes*)

XDR Source Code:

```
struct SendMoreExtended
{
    uint32 numMessages;
    uint32 numBytes;
};
```

SequenceNumber

class stellar_sdk.xdr.sequence_number.**SequenceNumber**(*sequence_number*)

XDR Source Code:

```
typedef int64 SequenceNumber;
```

SerializedBinaryFuseFilter

class stellar_sdk.xdr.serialized_binary_fuse_filter.**SerializedBinaryFuseFilter**(*type*, *input_hash_seed*, *filter_seed*, *segment_length*, *segement_length_mask*, *segment_count*, *segment_count_length*, *fingerprint_length*, *fingerprints*)

XDR Source Code:

```
struct SerializedBinaryFuseFilter
{
    BinaryFuseFilterType type;

    // Seed used to hash input to filter
    ShortHashSeed inputHashSeed;

    // Seed used for internal filter hash operations
    ShortHashSeed filterSeed;
    uint32 segmentLength;
    uint32 segementLengthMask;
    uint32 segmentCount;
    uint32 segmentCountLength;
    uint32 fingerprintLength; // Length in terms of element count, not bytes

    // Array of uint8_t, uint16_t, or uint32_t depending on filter type
    opaque fingerprints<>;
};
```

SetOptionsOp

class stellar_sdk.xdr.set_options_op.**SetOptionsOp**(*inflation_dest*, *clear_flags*, *set_flags*, *master_weight*, *low_threshold*, *med_threshold*, *high_threshold*, *home_domain*, *signer*)

XDR Source Code:

```
struct SetOptionsOp
{
    AccountID* inflationDest; // sets the inflation destination

    uint32* clearFlags; // which flags to clear
    uint32* setFlags;   // which flags to set

    // account threshold manipulation
    uint32* masterWeight; // weight of the master account
    uint32* lowThreshold;
    uint32* medThreshold;
    uint32* highThreshold;

    string32* homeDomain; // sets the home domain

    // Add, update or remove a signer for the account
    // signer is deleted if the weight is 0
    Signer* signer;
};
```

SetOptionsResult

class stellar_sdk.xdr.set_options_result.**SetOptionsResult**(*code*)

XDR Source Code:

```
union SetOptionsResult switch (SetOptionsResultCode code)
{
case SET_OPTIONS_SUCCESS:
    void;
case SET_OPTIONS_LOW_RESERVE:
case SET_OPTIONS_TOO_MANY_SIGNERS:
case SET_OPTIONS_BAD_FLAGS:
case SET_OPTIONS_INVALID_INFLATION:
case SET_OPTIONS_CANT_CHANGE:
case SET_OPTIONS_UNKNOWN_FLAG:
case SET_OPTIONS_THRESHOLD_OUT_OF_RANGE:
case SET_OPTIONS_BAD_SIGNER:
case SET_OPTIONS_INVALID_HOME_DOMAIN:
case SET_OPTIONS_AUTH_REVOCABLE_REQUIRED:
    void;
};
```

SetOptionsResultCode

class stellar_sdk.xdr.set_options_result_code.**SetOptionsResultCode**(**values*)

XDR Source Code:

```
enum SetOptionsResultCode
{
    // codes considered as "success" for the operation
    SET_OPTIONS_SUCCESS = 0,
    // codes considered as "failure" for the operation
    SET_OPTIONS_LOW_RESERVE = -1,      // not enough funds to add a signer
    SET_OPTIONS_TOO_MANY_SIGNERS = -2, // max number of signers already reached
    SET_OPTIONS_BAD_FLAGS = -3,        // invalid combination of clear/set flags
    SET_OPTIONS_INVALID_INFLATION = -4,      // inflation account does not exist
    SET_OPTIONS_CANT_CHANGE = -5,            // can no longer change this option
    SET_OPTIONS_UNKNOWN_FLAG = -6,           // can't set an unknown flag
    SET_OPTIONS_THRESHOLD_OUT_OF_RANGE = -7, // bad value for weight/threshold
    SET_OPTIONS_BAD_SIGNER = -8,             // signer cannot be masterkey
    SET_OPTIONS_INVALID_HOME_DOMAIN = -9,    // malformed home domain
    SET_OPTIONS_AUTH_REVOCABLE_REQUIRED =
        -10 // auth revocable is required for clawback
};
```

SetTrustLineFlagsOp

class stellar_sdk.xdr.set_trust_line_flags_op.**SetTrustLineFlagsOp**(*trustor*, *asset*, *clear_flags*, *set_flags*)

XDR Source Code:

```
struct SetTrustLineFlagsOp
{
    AccountID trustor;
    Asset asset;

    uint32 clearFlags; // which flags to clear
```

(continues on next page)

(continued from previous page)

```
    uint32 setFlags;   // which flags to set
};
```

SetTrustLineFlagsResult

class stellar_sdk.xdr.set_trust_line_flags_result.**SetTrustLineFlagsResult**(*code*)

XDR Source Code:

```
union SetTrustLineFlagsResult switch (SetTrustLineFlagsResultCode code)
{
case SET_TRUST_LINE_FLAGS_SUCCESS:
    void;
case SET_TRUST_LINE_FLAGS_MALFORMED:
case SET_TRUST_LINE_FLAGS_NO_TRUST_LINE:
case SET_TRUST_LINE_FLAGS_CANT_REVOKE:
case SET_TRUST_LINE_FLAGS_INVALID_STATE:
case SET_TRUST_LINE_FLAGS_LOW_RESERVE:
    void;
};
```

SetTrustLineFlagsResultCode

class stellar_sdk.xdr.set_trust_line_flags_result_code.**SetTrustLineFlagsResultCode**(**values*)

XDR Source Code:

```
enum SetTrustLineFlagsResultCode
{
    // codes considered as "success" for the operation
    SET_TRUST_LINE_FLAGS_SUCCESS = 0,

    // codes considered as "failure" for the operation
    SET_TRUST_LINE_FLAGS_MALFORMED = -1,
    SET_TRUST_LINE_FLAGS_NO_TRUST_LINE = -2,
    SET_TRUST_LINE_FLAGS_CANT_REVOKE = -3,
    SET_TRUST_LINE_FLAGS_INVALID_STATE = -4,
    SET_TRUST_LINE_FLAGS_LOW_RESERVE = -5 // claimable balances can't be created
                                          // on revoke due to low reserves
};
```

ShortHashSeed

class stellar_sdk.xdr.short_hash_seed.**ShortHashSeed**(*seed*)

XDR Source Code:

```
struct ShortHashSeed
{
    opaque seed[16];
};
```

Signature

class stellar_sdk.xdr.signature.**Signature**(*signature*)

XDR Source Code:

```
typedef opaque Signature<64>;
```

SignatureHint

class stellar_sdk.xdr.signature_hint.**SignatureHint**(*signature_hint*)

XDR Source Code:

```
typedef opaque SignatureHint[4];
```

SignedTimeSlicedSurveyRequestMessage

class stellar_sdk.xdr.signed_time_sliced_survey_request_message.**SignedTimeSlicedSurveyRequestMessage**(*req*
re-
que

XDR Source Code:

```
struct SignedTimeSlicedSurveyRequestMessage
{
    Signature requestSignature;
    TimeSlicedSurveyRequestMessage request;
};
```

SignedTimeSlicedSurveyResponseMessage

class stellar_sdk.xdr.signed_time_sliced_survey_response_message.**SignedTimeSlicedSurveyResponseMessage**(

XDR Source Code:

```
struct SignedTimeSlicedSurveyResponseMessage
{
    Signature responseSignature;
    TimeSlicedSurveyResponseMessage response;
};
```

SignedTimeSlicedSurveyStartCollectingMessage

class stellar_sdk.xdr.signed_time_sliced_survey_start_collecting_message.**SignedTimeSlicedSurveyStartCol**

XDR Source Code:

```
struct SignedTimeSlicedSurveyStartCollectingMessage
{
    Signature signature;
    TimeSlicedSurveyStartCollectingMessage startCollecting;
};
```

SignedTimeSlicedSurveyStopCollectingMessage

class stellar_sdk.xdr.signed_time_sliced_survey_stop_collecting_message.**SignedTimeSlicedSurveyStopColle**

XDR Source Code:

```
struct SignedTimeSlicedSurveyStopCollectingMessage
{
    Signature signature;
    TimeSlicedSurveyStopCollectingMessage stopCollecting;
};
```

Signer

class stellar_sdk.xdr.signer.**Signer**(*key*, *weight*)

XDR Source Code:

```
struct Signer
{
    SignerKey key;
    uint32 weight; // really only need 1 byte
};
```

SignerKey

class stellar_sdk.xdr.signer_key.**SignerKey**(*type*, *ed25519=None*, *pre_auth_tx=None*, *hash_x=None*, *ed25519_signed_payload=None*)

XDR Source Code:

```
union SignerKey switch (SignerKeyType type)
{
case SIGNER_KEY_TYPE_ED25519:
    uint256 ed25519;
case SIGNER_KEY_TYPE_PRE_AUTH_TX:
    /* SHA-256 Hash of TransactionSignaturePayload structure */
    uint256 preAuthTx;
case SIGNER_KEY_TYPE_HASH_X:
    /* Hash of random 256 bit preimage X */
    uint256 hashX;
case SIGNER_KEY_TYPE_ED25519_SIGNED_PAYLOAD:
    struct
    {
        /* Public key that must sign the payload. */
        uint256 ed25519;
        /* Payload to be raw signed by ed25519. */
        opaque payload<64>;
    } ed25519SignedPayload;
};
```

SignerKeyEd25519SignedPayload

class stellar_sdk.xdr.signer_key_ed25519_signed_payload.**SignerKeyEd25519SignedPayload**(*ed25519*, *payload*)

XDR Source Code:

```
struct
    {
        /* Public key that must sign the payload. */
        uint256 ed25519;
        /* Payload to be raw signed by ed25519. */
        opaque payload<64>;
    }
```

SignerKeyType

class stellar_sdk.xdr.signer_key_type.**SignerKeyType**(**values*)

XDR Source Code:

```
enum SignerKeyType
{
    SIGNER_KEY_TYPE_ED25519 = KEY_TYPE_ED25519,
    SIGNER_KEY_TYPE_PRE_AUTH_TX = KEY_TYPE_PRE_AUTH_TX,
    SIGNER_KEY_TYPE_HASH_X = KEY_TYPE_HASH_X,
    SIGNER_KEY_TYPE_ED25519_SIGNED_PAYLOAD = KEY_TYPE_ED25519_SIGNED_PAYLOAD
};
```

SimplePaymentResult

class stellar_sdk.xdr.simple_payment_result.**SimplePaymentResult**(*destination*, *asset*, *amount*)

XDR Source Code:

```
struct SimplePaymentResult
{
    AccountID destination;
    Asset asset;
    int64 amount;
};
```

SorobanAddressCredentials

class stellar_sdk.xdr.soroban_address_credentials.**SorobanAddressCredentials**(*address*, *nonce*, *signature_expiration_ledger*, *signature*)

XDR Source Code:

```
struct SorobanAddressCredentials
{
    SCAddress address;
    int64 nonce;
    uint32 signatureExpirationLedger;
```

(continues on next page)

(continued from previous page)

```
    SCVal signature;
};
```

SorobanAddressCredentialsWithDelegates

class stellar_sdk.xdr.soroban_address_credentials_with_delegates.**SorobanAddressCredentialsWithDelegates**

XDR Source Code:

```
struct SorobanAddressCredentialsWithDelegates
{
    SorobanAddressCredentials addressCredentials;
    SorobanDelegateSignature delegates<>;
};
```

SorobanAuthorizationEntries

class stellar_sdk.xdr.soroban_authorization_entries.**SorobanAuthorizationEntries**(*soroban_authorization_entries*

XDR Source Code:

```
typedef SorobanAuthorizationEntry SorobanAuthorizationEntries<>;
```

SorobanAuthorizationEntry

class stellar_sdk.xdr.soroban_authorization_entry.**SorobanAuthorizationEntry**(*credentials*, *root_invocation*)

XDR Source Code:

```
struct SorobanAuthorizationEntry
{
    SorobanCredentials credentials;
    SorobanAuthorizedInvocation rootInvocation;
};
```

SorobanAuthorizedFunction

class stellar_sdk.xdr.soroban_authorized_function.**SorobanAuthorizedFunction**(*type*, *contract_fn=None*, *create_contract_host_fn=None*, *create_contract_v2_host_fn=None*)

XDR Source Code:

```
union SorobanAuthorizedFunction switch (SorobanAuthorizedFunctionType type)
{
case SOROBAN_AUTHORIZED_FUNCTION_TYPE_CONTRACT_FN:
    InvokeContractArgs contractFn;
// This variant of auth payload for creating new contract instances
```

(continues on next page)

(continued from previous page)

```
// doesn't allow specifying the constructor arguments, creating contracts
// with constructors that take arguments is only possible by authorizing
// `SOROBAN_AUTHORIZED_FUNCTION_TYPE_CREATE_CONTRACT_V2_HOST_FN`
// (protocol 22+).
case SOROBAN_AUTHORIZED_FUNCTION_TYPE_CREATE_CONTRACT_HOST_FN:
    CreateContractArgs createContractHostFn;
// This variant of auth payload for creating new contract instances
// is only accepted in and after protocol 22. It allows authorizing the
// contract constructor arguments.
case SOROBAN_AUTHORIZED_FUNCTION_TYPE_CREATE_CONTRACT_V2_HOST_FN:
    CreateContractArgsV2 createContractV2HostFn;
};
```

SorobanAuthorizedFunctionType

class stellar_sdk.xdr.soroban_authorized_function_type.**SorobanAuthorizedFunctionType**(**values*)

XDR Source Code:

```
enum SorobanAuthorizedFunctionType
{
    SOROBAN_AUTHORIZED_FUNCTION_TYPE_CONTRACT_FN = 0,
    SOROBAN_AUTHORIZED_FUNCTION_TYPE_CREATE_CONTRACT_HOST_FN = 1,
    SOROBAN_AUTHORIZED_FUNCTION_TYPE_CREATE_CONTRACT_V2_HOST_FN = 2
};
```

SorobanAuthorizedInvocation

class stellar_sdk.xdr.soroban_authorized_invocation.**SorobanAuthorizedInvocation**(*function*, *sub_invocations*)

XDR Source Code:

```
struct SorobanAuthorizedInvocation
{
    SorobanAuthorizedFunction function;
    SorobanAuthorizedInvocation subInvocations<>;
};
```

SorobanCredentials

class stellar_sdk.xdr.soroban_credentials.**SorobanCredentials**(*type*, *address=None*, *address_v2=None*, *address_with_delegates=None*)

XDR Source Code:

```
union SorobanCredentials switch (SorobanCredentialsType type)
{
case SOROBAN_CREDENTIALS_SOURCE_ACCOUNT:
    void;
case SOROBAN_CREDENTIALS_ADDRESS:
    SorobanAddressCredentials address;
case SOROBAN_CREDENTIALS_ADDRESS_V2:
```

(continues on next page)

(continued from previous page)

```
    SorobanAddressCredentials addressV2;
case SOROBAN_CREDENTIALS_ADDRESS_WITH_DELEGATES:
    SorobanAddressCredentialsWithDelegates addressWithDelegates;
};
```

SorobanCredentialsType

class stellar_sdk.xdr.soroban_credentials_type.**SorobanCredentialsType**(**values*)

XDR Source Code:

```
enum SorobanCredentialsType
{
    SOROBAN_CREDENTIALS_SOURCE_ACCOUNT = 0,
    SOROBAN_CREDENTIALS_ADDRESS = 1,
    SOROBAN_CREDENTIALS_ADDRESS_V2 = 2,
    SOROBAN_CREDENTIALS_ADDRESS_WITH_DELEGATES = 3
};
```

SorobanDelegateSignature

class stellar_sdk.xdr.soroban_delegate_signature.**SorobanDelegateSignature**(*address*, *signature*, *nested_delegates*)

XDR Source Code:

```
struct SorobanDelegateSignature
{
    SCAddress address;
    SCVal signature;
    SorobanDelegateSignature nestedDelegates<>;
};
```

SorobanResources

class stellar_sdk.xdr.soroban_resources.**SorobanResources**(*footprint*, *instructions*, *disk_read_bytes*, *write_bytes*)

XDR Source Code:

```
struct SorobanResources
{
    // The ledger footprint of the transaction.
    LedgerFootprint footprint;
    // The maximum number of instructions this transaction can use
    uint32 instructions;

    // The maximum number of bytes this transaction can read from disk backed␣
↪entries
    uint32 diskReadBytes;
    // The maximum number of bytes this transaction can write to ledger
    uint32 writeBytes;
};
```

SorobanResourcesExtV0

class stellar_sdk.xdr.soroban_resources_ext_v0.**SorobanResourcesExtV0**(*archived_soroban_entries*)

XDR Source Code:

```
struct SorobanResourcesExtV0
{
    // Vector of indices representing what Soroban
    // entries in the footprint are archived, based on the
    // order of keys provided in the readWrite footprint.
    uint32 archivedSorobanEntries<>;
};
```

SorobanTransactionData

class stellar_sdk.xdr.soroban_transaction_data.**SorobanTransactionData**(*ext*, *resources*, *resource_fee*)

XDR Source Code:

```
struct SorobanTransactionData
{
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        SorobanResourcesExtV0 resourceExt;
    } ext;
    SorobanResources resources;
    // Amount of the transaction `fee` allocated to the Soroban resource fees.
    // The fraction of `resourceFee` corresponding to `resources` specified
    // above is *not* refundable (i.e. fees for instructions, ledger I/O), as
    // well as fees for the transaction size.
    // The remaining part of the fee is refundable and the charged value is
    // based on the actual consumption of refundable resources (events, ledger
    // rent bumps).
    // The `inclusionFee` used for prioritization of the transaction is defined
    // as `tx.fee - resourceFee`.
    int64 resourceFee;
};
```

SorobanTransactionDataExt

class stellar_sdk.xdr.soroban_transaction_data_ext.**SorobanTransactionDataExt**(*v*, *resource_ext=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        SorobanResourcesExtV0 resourceExt;
    }
```

SorobanTransactionMeta

class stellar_sdk.xdr.soroban_transaction_meta.**SorobanTransactionMeta**(*ext*, *events*, *return_value*, *diagnostic_events*)

XDR Source Code:

```
struct SorobanTransactionMeta
{
    SorobanTransactionMetaExt ext;

    ContractEvent events<>;             // custom events populated by the
                                        // contracts themselves.
    SCVal returnValue;                  // return value of the host fn invocation

    // Diagnostics events that are not hashed.
    // This will contain all contract and diagnostic events. Even ones
    // that were emitted in a failed contract call.
    DiagnosticEvent diagnosticEvents<>;
};
```

SorobanTransactionMetaExt

class stellar_sdk.xdr.soroban_transaction_meta_ext.**SorobanTransactionMetaExt**(*v*, *v1=None*)

XDR Source Code:

```
union SorobanTransactionMetaExt switch (int v)
{
case 0:
    void;
case 1:
    SorobanTransactionMetaExtV1 v1;
};
```

SorobanTransactionMetaExtV1

class stellar_sdk.xdr.soroban_transaction_meta_ext_v1.**SorobanTransactionMetaExtV1**(*ext*, *total_non_refundable_resourc*
total_refundable_resource_fe
rent_fee_charged)

XDR Source Code:

```
struct SorobanTransactionMetaExtV1
{
    ExtensionPoint ext;

    // The following are the components of the overall Soroban resource fee
    // charged for the transaction.
    // The following relation holds:
    // `resourceFeeCharged = totalNonRefundableResourceFeeCharged +␣
↪totalRefundableResourceFeeCharged`
    // where `resourceFeeCharged` is the overall fee charged for the
    // transaction. Also, `resourceFeeCharged` <= `sorobanData.resourceFee`
```

(continues on next page)

(continued from previous page)

```
    // i.e.we never charge more than the declared resource fee.
    // The inclusion fee for charged the Soroban transaction can be found using
    // the following equation:
    // `result.feeCharged = resourceFeeCharged + inclusionFeeCharged`.

    // Total amount (in stroops) that has been charged for non-refundable
    // Soroban resources.
    // Non-refundable resources are charged based on the usage declared in
    // the transaction envelope (such as `instructions`, `readBytes` etc.) and
    // is charged regardless of the success of the transaction.
    int64 totalNonRefundableResourceFeeCharged;
    // Total amount (in stroops) that has been charged for refundable
    // Soroban resource fees.
    // Currently this comprises the rent fee (`rentFeeCharged`) and the
    // fee for the events and return value.
    // Refundable resources are charged based on the actual resources usage.
    // Since currently refundable resources are only used for the successful
    // transactions, this will be `0` for failed transactions.
    int64 totalRefundableResourceFeeCharged;
    // Amount (in stroops) that has been charged for rent.
    // This is a part of `totalNonRefundableResourceFeeCharged`.
    int64 rentFeeCharged;
};
```

SorobanTransactionMetaV2

class stellar_sdk.xdr.soroban_transaction_meta_v2.**SorobanTransactionMetaV2**(*ext*, *return_value*)

XDR Source Code:

```
struct SorobanTransactionMetaV2
{
    SorobanTransactionMetaExt ext;

    SCVal* returnValue;
};
```

SponsorshipDescriptor

class stellar_sdk.xdr.sponsorship_descriptor.**SponsorshipDescriptor**(*sponsorship_descriptor*)

XDR Source Code:

```
typedef AccountID* SponsorshipDescriptor;
```

StateArchivalSettings

class stellar_sdk.xdr.state_archival_settings.**StateArchivalSettings**(*max_entry_ttl*, *min_temporary_ttl*, *min_persistent_ttl*, *persistent_rent_rate_denominator*, *temp_rent_rate_denominator*, *max_entries_to_archive*, *live_soroban_state_size_window_sample_size*, *live_soroban_state_size_window_sample_peri*, *eviction_scan_size*, *starting_eviction_scan_level*)

XDR Source Code:

```
struct StateArchivalSettings {
    uint32 maxEntryTTL;
    uint32 minTemporaryTTL;
    uint32 minPersistentTTL;

    // rent_fee = wfee_rate_average / rent_rate_denominator_for_type
    int64 persistentRentRateDenominator;
    int64 tempRentRateDenominator;

    // max number of entries that emit archival meta in a single ledger
    uint32 maxEntriesToArchive;

    // Number of snapshots to use when calculating average live Soroban State size
    uint32 liveSorobanStateSizeWindowSampleSize;

    // How often to sample the live Soroban State size for the average, in ledgers
    uint32 liveSorobanStateSizeWindowSamplePeriod;

    // Maximum number of bytes that we scan for eviction per ledger
    uint32 evictionScanSize;

    // Lowest BucketList level to be scanned to evict entries
    uint32 startingEvictionScanLevel;
};
```

StellarMessage

class stellar_sdk.xdr.stellar_message.**StellarMessage**(*type*, *error=None*, *hello=None*, *auth=None*, *dont_have=None*, *peers=None*, *tx_set_hash=None*, *tx_set=None*, *generalized_tx_set=None*, *transaction=None*, *signed_time_sliced_survey_request_message=None*, *signed_time_sliced_survey_response_message=None*, *signed_time_sliced_survey_start_collecting_message=None*, *signed_time_sliced_survey_stop_collecting_message=None*, *q_set_hash=None*, *q_set=None*, *envelope=None*, *get_scp_ledger_seq=None*, *send_more_message=None*, *send_more_extended_message=None*, *flood_advert=None*, *flood_demand=None*)

XDR Source Code:

```
union StellarMessage switch (MessageType type)
{
case ERROR_MSG:
    Error error;
case HELLO:
    Hello hello;
case AUTH:
    Auth auth;
case DONT_HAVE:
    DontHave dontHave;
case PEERS:
    PeerAddress peers<100>;

case GET_TX_SET:
    uint256 txSetHash;
case TX_SET:
    TransactionSet txSet;
case GENERALIZED_TX_SET:
    GeneralizedTransactionSet generalizedTxSet;

case TRANSACTION:
    TransactionEnvelope transaction;

case TIME_SLICED_SURVEY_REQUEST:
    SignedTimeSlicedSurveyRequestMessage signedTimeSlicedSurveyRequestMessage;

case TIME_SLICED_SURVEY_RESPONSE:
    SignedTimeSlicedSurveyResponseMessage signedTimeSlicedSurveyResponseMessage;

case TIME_SLICED_SURVEY_START_COLLECTING:
    SignedTimeSlicedSurveyStartCollectingMessage
        signedTimeSlicedSurveyStartCollectingMessage;

case TIME_SLICED_SURVEY_STOP_COLLECTING:
    SignedTimeSlicedSurveyStopCollectingMessage
        signedTimeSlicedSurveyStopCollectingMessage;

// SCP
case GET_SCP_QUORUMSET:
    uint256 qSetHash;
case SCP_QUORUMSET:
    SCPQuorumSet qSet;
case SCP_MESSAGE:
    SCPEnvelope envelope;
case GET_SCP_STATE:
    uint32 getSCPLedgerSeq; // ledger seq requested ; if 0, requests the latest
case SEND_MORE:
    SendMore sendMoreMessage;
case SEND_MORE_EXTENDED:
    SendMoreExtended sendMoreExtendedMessage;
// Pull mode
```

(continues on next page)

(continued from previous page)

```
case FLOOD_ADVERT:
     FloodAdvert floodAdvert;
case FLOOD_DEMAND:
     FloodDemand floodDemand;
};
```

StellarValue

class stellar_sdk.xdr.stellar_value.**StellarValue**(*tx_set_hash*, *close_time*, *upgrades*, *ext*)

XDR Source Code:

```
struct StellarValue
{
    Hash txSetHash;      // transaction set to apply to previous ledger
    TimePoint closeTime; // network close time

    // upgrades to apply to the previous ledger (usually empty)
    // this is a vector of encoded 'LedgerUpgrade' so that nodes can drop
    // unknown steps during consensus if needed.
    // see notes below on 'LedgerUpgrade' for more detail
    // max size is dictated by number of upgrade types (+ room for future)
    UpgradeType upgrades<6>;

    // reserved for future use
    union switch (StellarValueType v)
    {
    case STELLAR_VALUE_BASIC:
        void;
    case STELLAR_VALUE_SIGNED:
        LedgerCloseValueSignature lcValueSignature;
    }
    ext;
};
```

StellarValueExt

class stellar_sdk.xdr.stellar_value_ext.**StellarValueExt**(*v*, *lc_value_signature=None*)

XDR Source Code:

```
union switch (StellarValueType v)
    {
    case STELLAR_VALUE_BASIC:
        void;
    case STELLAR_VALUE_SIGNED:
        LedgerCloseValueSignature lcValueSignature;
    }
```

StellarValueType

class stellar_sdk.xdr.stellar_value_type.**StellarValueType**(**values*)

XDR Source Code:

```
enum StellarValueType
{
    STELLAR_VALUE_BASIC = 0,
    STELLAR_VALUE_SIGNED = 1
};
```

StoredDebugTransactionSet

class stellar_sdk.xdr.stored_debug_transaction_set.**StoredDebugTransactionSet**(*tx_set*, *ledger_seq*, *scp_value*)

XDR Source Code:

```
struct StoredDebugTransactionSet
{
        StoredTransactionSet txSet;
        uint32 ledgerSeq;
        StellarValue scpValue;
};
```

StoredTransactionSet

class stellar_sdk.xdr.stored_transaction_set.**StoredTransactionSet**(*v*, *tx_set=None*, *generalized_tx_set=None*)

XDR Source Code:

```
union StoredTransactionSet switch (int v)
{
case 0:
        TransactionSet txSet;
case 1:
        GeneralizedTransactionSet generalizedTxSet;
};
```

String

class stellar_sdk.xdr.base.**String**(*value*, *size*)

String32

class stellar_sdk.xdr.string32.**String32**(*string32*)

XDR Source Code:

```
typedef string string32<32>;
```

String64

class stellar_sdk.xdr.string64.**String64**(*string64*)

XDR Source Code:

```
typedef string string64<64>;
```

SurveyMessageCommandType

class stellar_sdk.xdr.survey_message_command_type.**SurveyMessageCommandType**(**values*)

XDR Source Code:

```
enum SurveyMessageCommandType
{
    TIME_SLICED_SURVEY_TOPOLOGY = 1
};
```

SurveyMessageResponseType

class stellar_sdk.xdr.survey_message_response_type.**SurveyMessageResponseType**(**values*)

XDR Source Code:

```
enum SurveyMessageResponseType
{
    SURVEY_TOPOLOGY_RESPONSE_V2 = 2
};
```

SurveyRequestMessage

class stellar_sdk.xdr.survey_request_message.**SurveyRequestMessage**(*surveyor_peer_id*, *surveyed_peer_id*, *ledger_num*, *encryption_key*, *command_type*)

XDR Source Code:

```
struct SurveyRequestMessage
{
    NodeID surveyorPeerID;
    NodeID surveyedPeerID;
    uint32 ledgerNum;
    Curve25519Public encryptionKey;
    SurveyMessageCommandType commandType;
};
```

SurveyResponseBody

class stellar_sdk.xdr.survey_response_body.**SurveyResponseBody**(*type*, *topology_response_body_v2=None*)

XDR Source Code:

```
union SurveyResponseBody switch (SurveyMessageResponseType type)
{
case SURVEY_TOPOLOGY_RESPONSE_V2:
```

(continues on next page)

(continued from previous page)

```
    TopologyResponseBodyV2 topologyResponseBodyV2;
};
```

SurveyResponseMessage

class stellar_sdk.xdr.survey_response_message.**SurveyResponseMessage**(*surveyor_peer_id*, *surveyed_peer_id*, *ledger_num*, *command_type*, *encrypted_body*)

XDR Source Code:

```
struct SurveyResponseMessage
{
    NodeID surveyorPeerID;
    NodeID surveyedPeerID;
    uint32 ledgerNum;
    SurveyMessageCommandType commandType;
    EncryptedBody encryptedBody;
};
```

TTLEntry

class stellar_sdk.xdr.ttl_entry.**TTLEntry**(*key_hash*, *live_until_ledger_seq*)

XDR Source Code:

```
struct TTLEntry {
    // Hash of the LedgerKey that is associated with this TTLEntry
    Hash keyHash;
    uint32 liveUntilLedgerSeq;
};
```

ThresholdIndexes

class stellar_sdk.xdr.threshold_indexes.**ThresholdIndexes**(**values*)

XDR Source Code:

```
enum ThresholdIndexes
{
    THRESHOLD_MASTER_WEIGHT = 0,
    THRESHOLD_LOW = 1,
    THRESHOLD_MED = 2,
    THRESHOLD_HIGH = 3
};
```

Thresholds

class stellar_sdk.xdr.thresholds.**Thresholds**(*thresholds*)

XDR Source Code:

```
typedef opaque Thresholds[4];
```

TimeBounds

class stellar_sdk.xdr.time_bounds.**TimeBounds**(*min_time*, *max_time*)

XDR Source Code:

```
struct TimeBounds
{
    TimePoint minTime;
    TimePoint maxTime; // 0 here means no maxTime
};
```

TimePoint

class stellar_sdk.xdr.time_point.**TimePoint**(*time_point*)

XDR Source Code:

```
typedef uint64 TimePoint;
```

TimeSlicedNodeData

class stellar_sdk.xdr.time_sliced_node_data.**TimeSlicedNodeData**(*added_authenticated_peers*, *dropped_authenticated_peers*, *total_inbound_peer_count*, *total_outbound_peer_count*, *p75_scp_first_to_self_latency_ms*, *p75_scp_self_to_other_latency_ms*, *lost_sync_count*, *is_validator*, *max_inbound_peer_count*, *max_outbound_peer_count*)

XDR Source Code:

```
struct TimeSlicedNodeData
{
    uint32 addedAuthenticatedPeers;
    uint32 droppedAuthenticatedPeers;
    uint32 totalInboundPeerCount;
    uint32 totalOutboundPeerCount;

    // SCP stats
    uint32 p75SCPFirstToSelfLatencyMs;
    uint32 p75SCPSelfToOtherLatencyMs;

    // How many times the node lost sync in the time slice
    uint32 lostSyncCount;

    // Config data
    bool isValidator;
    uint32 maxInboundPeerCount;
    uint32 maxOutboundPeerCount;
};
```

TimeSlicedPeerData

class stellar_sdk.xdr.time_sliced_peer_data.**TimeSlicedPeerData**(*peer_stats*, *average_latency_ms*)

XDR Source Code:

```
struct TimeSlicedPeerData
{
    PeerStats peerStats;
    uint32 averageLatencyMs;
};
```

TimeSlicedPeerDataList

class stellar_sdk.xdr.time_sliced_peer_data_list.**TimeSlicedPeerDataList**(*time_sliced_peer_data_list*)

XDR Source Code:

```
typedef TimeSlicedPeerData TimeSlicedPeerDataList<25>;
```

TimeSlicedSurveyRequestMessage

class stellar_sdk.xdr.time_sliced_survey_request_message.**TimeSlicedSurveyRequestMessage**(*request*, *nonce*, *inbound_peers_index*, *outbound_peers_index*)

XDR Source Code:

```
struct TimeSlicedSurveyRequestMessage
{
    SurveyRequestMessage request;
    uint32 nonce;
    uint32 inboundPeersIndex;
    uint32 outboundPeersIndex;
};
```

TimeSlicedSurveyResponseMessage

class stellar_sdk.xdr.time_sliced_survey_response_message.**TimeSlicedSurveyResponseMessage**(*response*, *nonce*)

XDR Source Code:

```
struct TimeSlicedSurveyResponseMessage
{
    SurveyResponseMessage response;
    uint32 nonce;
};
```

TimeSlicedSurveyStartCollectingMessage

class stellar_sdk.xdr.time_sliced_survey_start_collecting_message.**TimeSlicedSurveyStartCollectingMessage**

XDR Source Code:

```
struct TimeSlicedSurveyStartCollectingMessage
{
    NodeID surveyorID;
    uint32 nonce;
    uint32 ledgerNum;
};
```

TimeSlicedSurveyStopCollectingMessage

class stellar_sdk.xdr.time_sliced_survey_stop_collecting_message.**TimeSlicedSurveyStopCollectingMessage**(

XDR Source Code:

```
struct TimeSlicedSurveyStopCollectingMessage
{
    NodeID surveyorID;
    uint32 nonce;
    uint32 ledgerNum;
};
```

TopologyResponseBodyV2

class stellar_sdk.xdr.topology_response_body_v2.**TopologyResponseBodyV2**(*inbound_peers*, *outbound_peers*, *node_data*)

XDR Source Code:

```
struct TopologyResponseBodyV2
{
    TimeSlicedPeerDataList inboundPeers;
    TimeSlicedPeerDataList outboundPeers;
    TimeSlicedNodeData nodeData;
};
```

Transaction

class stellar_sdk.xdr.transaction.**Transaction**(*source_account*, *fee*, *seq_num*, *cond*, *memo*, *operations*, *ext*)

XDR Source Code:

```
struct Transaction
{
    // account used to run the transaction
    MuxedAccount sourceAccount;

    // the fee the sourceAccount will pay
    uint32 fee;
```

(continues on next page)

(continued from previous page)

```
    // sequence number to consume in the account
    SequenceNumber seqNum;

    // validity conditions
    Preconditions cond;

    Memo memo;

    Operation operations<MAX_OPS_PER_TX>;

    union switch (int v)
    {
    case 0:
        void;
    case 1:
        SorobanTransactionData sorobanData;
    }
    ext;
};
```

TransactionEnvelope

class stellar_sdk.xdr.transaction_envelope.**TransactionEnvelope**(*type*, *v0=None*, *v1=None*, *fee_bump=None*)

XDR Source Code:

```
union TransactionEnvelope switch (EnvelopeType type)
{
case ENVELOPE_TYPE_TX_V0:
    TransactionV0Envelope v0;
case ENVELOPE_TYPE_TX:
    TransactionV1Envelope v1;
case ENVELOPE_TYPE_TX_FEE_BUMP:
    FeeBumpTransactionEnvelope feeBump;
};
```

TransactionEvent

class stellar_sdk.xdr.transaction_event.**TransactionEvent**(*stage*, *event*)

XDR Source Code:

```
struct TransactionEvent {
    TransactionEventStage stage;  // Stage at which an event has occurred.
    ContractEvent event;  // The contract event that has occurred.
};
```

TransactionEventStage

class stellar_sdk.xdr.transaction_event_stage.**TransactionEventStage**(**values*)

XDR Source Code:

```
enum TransactionEventStage {
    // The event has happened before any one of the transactions has its
    // operations applied.
    TRANSACTION_EVENT_STAGE_BEFORE_ALL_TXS = 0,
    // The event has happened immediately after operations of the transaction
    // have been applied.
    TRANSACTION_EVENT_STAGE_AFTER_TX = 1,
    // The event has happened after every transaction had its operations
    // applied.
    TRANSACTION_EVENT_STAGE_AFTER_ALL_TXS = 2
};
```

TransactionExt

class stellar_sdk.xdr.transaction_ext.**TransactionExt**(*v*, *soroban_data=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        SorobanTransactionData sorobanData;
    }
```

TransactionHistoryEntry

class stellar_sdk.xdr.transaction_history_entry.**TransactionHistoryEntry**(*ledger_seq*, *tx_set*, *ext*)

XDR Source Code:

```
struct TransactionHistoryEntry
{
    uint32 ledgerSeq;
    TransactionSet txSet;

    // when v != 0, txSet must be empty
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        GeneralizedTransactionSet generalizedTxSet;
    }
    ext;
};
```

TransactionHistoryEntryExt

class stellar_sdk.xdr.transaction_history_entry_ext.**TransactionHistoryEntryExt**(*v*, *generalized_tx_set=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        GeneralizedTransactionSet generalizedTxSet;
    }
```

TransactionHistoryResultEntry

class stellar_sdk.xdr.transaction_history_result_entry.**TransactionHistoryResultEntry**(*ledger_seq*, *tx_result_set*, *ext*)

XDR Source Code:

```
struct TransactionHistoryResultEntry
{
    uint32 ledgerSeq;
    TransactionResultSet txResultSet;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

TransactionHistoryResultEntryExt

class stellar_sdk.xdr.transaction_history_result_entry_ext.**TransactionHistoryResultEntryExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

TransactionMeta

class stellar_sdk.xdr.transaction_meta.**TransactionMeta**(*v*, *operations=None*, *v1=None*, *v2=None*, *v3=None*, *v4=None*)

XDR Source Code:

```
union TransactionMeta switch (int v)
{
case 0:
    OperationMeta operations<>;
case 1:
    TransactionMetaV1 v1;
```

(continues on next page)

(continued from previous page)

```
case 2:
    TransactionMetaV2 v2;
case 3:
    TransactionMetaV3 v3;
case 4:
    TransactionMetaV4 v4;
};
```

TransactionMetaV1

class stellar_sdk.xdr.transaction_meta_v1.**TransactionMetaV1**(*tx_changes*, *operations*)

XDR Source Code:

```
struct TransactionMetaV1
{
    LedgerEntryChanges txChanges; // tx level changes if any
    OperationMeta operations<>;   // meta for each operation
};
```

TransactionMetaV2

class stellar_sdk.xdr.transaction_meta_v2.**TransactionMetaV2**(*tx_changes_before*, *operations*, *tx_changes_after*)

XDR Source Code:

```
struct TransactionMetaV2
{
    LedgerEntryChanges txChangesBefore; // tx level changes before operations
                                        // are applied if any
    OperationMeta operations<>;         // meta for each operation
    LedgerEntryChanges txChangesAfter;  // tx level changes after operations are
                                        // applied if any
};
```

TransactionMetaV3

class stellar_sdk.xdr.transaction_meta_v3.**TransactionMetaV3**(*ext*, *tx_changes_before*, *operations*, *tx_changes_after*, *soroban_meta*)

XDR Source Code:

```
struct TransactionMetaV3
{
    ExtensionPoint ext;

    LedgerEntryChanges txChangesBefore;  // tx level changes before operations
                                         // are applied if any
    OperationMeta operations<>;          // meta for each operation
    LedgerEntryChanges txChangesAfter;   // tx level changes after operations are
                                         // applied if any
    SorobanTransactionMeta* sorobanMeta; // Soroban-specific meta (only for
                                         // Soroban transactions).
};
```

TransactionMetaV4

class stellar_sdk.xdr.transaction_meta_v4.**TransactionMetaV4**(*ext*, *tx_changes_before*, *operations*, *tx_changes_after*, *soroban_meta*, *events*, *diagnostic_events*)

XDR Source Code:

```
struct TransactionMetaV4
{
    ExtensionPoint ext;

    LedgerEntryChanges txChangesBefore;  // tx level changes before operations
                                         // are applied if any
    OperationMetaV2 operations<>;        // meta for each operation
    LedgerEntryChanges txChangesAfter;   // tx level changes after operations are
                                         // applied if any
    SorobanTransactionMetaV2* sorobanMeta; // Soroban-specific meta (only for
                                           // Soroban transactions).

    TransactionEvent events<>; // Used for transaction-level events (like fee␣
↪payment)
    DiagnosticEvent diagnosticEvents<>; // Used for all diagnostic information
};
```

TransactionPhase

class stellar_sdk.xdr.transaction_phase.**TransactionPhase**(*v*, *v0_components=None*, *parallel_txs_component=None*)

XDR Source Code:

```
union TransactionPhase switch (int v)
{
case 0:
    TxSetComponent v0Components<>;
case 1:
    ParallelTxsComponent parallelTxsComponent;
};
```

TransactionResult

class stellar_sdk.xdr.transaction_result.**TransactionResult**(*fee_charged*, *result*, *ext*)

XDR Source Code:

```
struct TransactionResult
{
    int64 feeCharged; // actual fee charged for the transaction

    union switch (TransactionResultCode code)
    {
    case txFEE_BUMP_INNER_SUCCESS:
    case txFEE_BUMP_INNER_FAILED:
        InnerTransactionResultPair innerResultPair;
    case txSUCCESS:
```

(continues on next page)

(continued from previous page)

```
    case txFAILED:
        OperationResult results<>;
    case txTOO_EARLY:
    case txTOO_LATE:
    case txMISSING_OPERATION:
    case txBAD_SEQ:
    case txBAD_AUTH:
    case txINSUFFICIENT_BALANCE:
    case txNO_ACCOUNT:
    case txINSUFFICIENT_FEE:
    case txBAD_AUTH_EXTRA:
    case txINTERNAL_ERROR:
    case txNOT_SUPPORTED:
    // case txFEE_BUMP_INNER_FAILED: handled above
    case txBAD_SPONSORSHIP:
    case txBAD_MIN_SEQ_AGE_OR_GAP:
    case txMALFORMED:
    case txSOROBAN_INVALID:
    case txFROZEN_KEY_ACCESSED:
        void;
    }
    result;

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

TransactionResultCode

class stellar_sdk.xdr.transaction_result_code.**TransactionResultCode**(**values*)

XDR Source Code:

```
enum TransactionResultCode
{
    txFEE_BUMP_INNER_SUCCESS = 1, // fee bump inner transaction succeeded
    txSUCCESS = 0,                // all operations succeeded

    txFAILED = -1, // one of the operations failed (none were applied)

    txTOO_EARLY = -2,         // ledger closeTime before minTime
    txTOO_LATE = -3,          // ledger closeTime after maxTime
    txMISSING_OPERATION = -4, // no operation was specified
    txBAD_SEQ = -5,           // sequence number does not match source account

    txBAD_AUTH = -6,             // too few valid signatures / wrong network
    txINSUFFICIENT_BALANCE = -7, // fee would bring account below reserve
    txNO_ACCOUNT = -8,           // source account not found
```

(continues on next page)

(continued from previous page)

```
    txINSUFFICIENT_FEE = -9,      // fee is too small
    txBAD_AUTH_EXTRA = -10,       // unused signatures attached to transaction
    txINTERNAL_ERROR = -11,       // an unknown error occurred

    txNOT_SUPPORTED = -12,          // transaction type not supported
    txFEE_BUMP_INNER_FAILED = -13,  // fee bump inner transaction failed
    txBAD_SPONSORSHIP = -14,        // sponsorship not confirmed
    txBAD_MIN_SEQ_AGE_OR_GAP = -15, // minSeqAge or minSeqLedgerGap conditions not␣
↪met
    txMALFORMED = -16,              // precondition is invalid
    txSOROBAN_INVALID = -17,        // soroban-specific preconditions were not met
    txFROZEN_KEY_ACCESSED = -18     // a 'frozen' ledger key is accessed by any␣
↪operation
};
```

TransactionResultExt

class stellar_sdk.xdr.transaction_result_ext.**TransactionResultExt**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

TransactionResultMeta

class stellar_sdk.xdr.transaction_result_meta.**TransactionResultMeta**(*result*, *fee_processing*, *tx_apply_processing*)

XDR Source Code:

```
struct TransactionResultMeta
{
    TransactionResultPair result;
    LedgerEntryChanges feeProcessing;
    TransactionMeta txApplyProcessing;
};
```

TransactionResultMetaV1

class stellar_sdk.xdr.transaction_result_meta_v1.**TransactionResultMetaV1**(*ext*, *result*, *fee_processing*, *tx_apply_processing*, *post_tx_apply_fee_processing*)

XDR Source Code:

```
struct TransactionResultMetaV1
{
    ExtensionPoint ext;
```

(continues on next page)

(continued from previous page)

```
    TransactionResultPair result;
    LedgerEntryChanges feeProcessing;
    TransactionMeta txApplyProcessing;

    LedgerEntryChanges postTxApplyFeeProcessing;
};
```

TransactionResultPair

class stellar_sdk.xdr.transaction_result_pair.**TransactionResultPair**(*transaction_hash*, *result*)

XDR Source Code:

```
struct TransactionResultPair
{
    Hash transactionHash;
    TransactionResult result; // result for the transaction
};
```

TransactionResultResult

class stellar_sdk.xdr.transaction_result_result.**TransactionResultResult**(*code*, *inner_result_pair=None*, *results=None*)

XDR Source Code:

```
union switch (TransactionResultCode code)
    {
    case txFEE_BUMP_INNER_SUCCESS:
    case txFEE_BUMP_INNER_FAILED:
        InnerTransactionResultPair innerResultPair;
    case txSUCCESS:
    case txFAILED:
        OperationResult results<>;
    case txTOO_EARLY:
    case txTOO_LATE:
    case txMISSING_OPERATION:
    case txBAD_SEQ:
    case txBAD_AUTH:
    case txINSUFFICIENT_BALANCE:
    case txNO_ACCOUNT:
    case txINSUFFICIENT_FEE:
    case txBAD_AUTH_EXTRA:
    case txINTERNAL_ERROR:
    case txNOT_SUPPORTED:
    // case txFEE_BUMP_INNER_FAILED: handled above
    case txBAD_SPONSORSHIP:
    case txBAD_MIN_SEQ_AGE_OR_GAP:
    case txMALFORMED:
    case txSOROBAN_INVALID:
    case txFROZEN_KEY_ACCESSED:
        void;
    }
```

TransactionResultSet

class stellar_sdk.xdr.transaction_result_set.**TransactionResultSet**(*results*)

XDR Source Code:

```
struct TransactionResultSet
{
    TransactionResultPair results<>;
};
```

TransactionSet

class stellar_sdk.xdr.transaction_set.**TransactionSet**(*previous_ledger_hash*, *txs*)

XDR Source Code:

```
struct TransactionSet
{
    Hash previousLedgerHash;
    TransactionEnvelope txs<>;
};
```

TransactionSetV1

class stellar_sdk.xdr.transaction_set_v1.**TransactionSetV1**(*previous_ledger_hash*, *phases*)

XDR Source Code:

```
struct TransactionSetV1
{
    Hash previousLedgerHash;
    TransactionPhase phases<>;
};
```

TransactionSignaturePayload

class stellar_sdk.xdr.transaction_signature_payload.**TransactionSignaturePayload**(*network_id*, *tagged_transaction*)

XDR Source Code:

```
struct TransactionSignaturePayload
{
    Hash networkId;
    union switch (EnvelopeType type)
    {
    // Backwards Compatibility: Use ENVELOPE_TYPE_TX to sign ENVELOPE_TYPE_TX_V0
    case ENVELOPE_TYPE_TX:
        Transaction tx;
    case ENVELOPE_TYPE_TX_FEE_BUMP:
        FeeBumpTransaction feeBump;
    }
    taggedTransaction;
};
```

TransactionSignaturePayloadTaggedTransaction

class stellar_sdk.xdr.transaction_signature_payload_tagged_transaction.**TransactionSignaturePayloadTagge**

XDR Source Code:

```
union switch (EnvelopeType type)
    {
    // Backwards Compatibility: Use ENVELOPE_TYPE_TX to sign ENVELOPE_TYPE_TX_V0
    case ENVELOPE_TYPE_TX:
        Transaction tx;
    case ENVELOPE_TYPE_TX_FEE_BUMP:
        FeeBumpTransaction feeBump;
    }
```

TransactionV0

class stellar_sdk.xdr.transaction_v0.**TransactionV0**(*source_account_ed25519*, *fee*, *seq_num*, *time_bounds*, *memo*, *operations*, *ext*)

XDR Source Code:

```
struct TransactionV0
{
    uint256 sourceAccountEd25519;
    uint32 fee;
    SequenceNumber seqNum;
    TimeBounds* timeBounds;
    Memo memo;
    Operation operations<MAX_OPS_PER_TX>;
    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

TransactionV0Envelope

class stellar_sdk.xdr.transaction_v0_envelope.**TransactionV0Envelope**(*tx*, *signatures*)

XDR Source Code:

```
struct TransactionV0Envelope
{
    TransactionV0 tx;
    /* Each decorated signature is a signature over the SHA256 hash of
     * a TransactionSignaturePayload */
    DecoratedSignature signatures<20>;
};
```

TransactionV0Ext

class stellar_sdk.xdr.transaction_v0_ext.**TransactionV0Ext**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

TransactionV1Envelope

class stellar_sdk.xdr.transaction_v1_envelope.**TransactionV1Envelope**(*tx*, *signatures*)

XDR Source Code:

```
struct TransactionV1Envelope
{
    Transaction tx;
    /* Each decorated signature is a signature over the SHA256 hash of
     * a TransactionSignaturePayload */
    DecoratedSignature signatures<20>;
};
```

TrustLineAsset

class stellar_sdk.xdr.trust_line_asset.**TrustLineAsset**(*type*, *alpha_num4=None*, *alpha_num12=None*, *liquidity_pool_id=None*)

XDR Source Code:

```
union TrustLineAsset switch (AssetType type)
{
case ASSET_TYPE_NATIVE: // Not credit
    void;

case ASSET_TYPE_CREDIT_ALPHANUM4:
    AlphaNum4 alphaNum4;

case ASSET_TYPE_CREDIT_ALPHANUM12:
    AlphaNum12 alphaNum12;

case ASSET_TYPE_POOL_SHARE:
    PoolID liquidityPoolID;

    // add other asset types here in the future
};
```

TrustLineEntry

class stellar_sdk.xdr.trust_line_entry.**TrustLineEntry**(*account_id*, *asset*, *balance*, *limit*, *flags*, *ext*)

XDR Source Code:

```
struct TrustLineEntry
{
    AccountID accountID;  // account this trustline belongs to
    TrustLineAsset asset; // type of asset (with issuer)
    int64 balance;        // how much of this asset the user has.
                          // Asset defines the unit for this;

    int64 limit;  // balance cannot be above this
    uint32 flags; // see TrustLineFlags

    // reserved for future use
    union switch (int v)
    {
    case 0:
        void;
    case 1:
        struct
        {
            Liabilities liabilities;

            union switch (int v)
            {
            case 0:
                void;
            case 2:
                TrustLineEntryExtensionV2 v2;
            }
            ext;
        } v1;
    }
    ext;
};
```

TrustLineEntryExt

class stellar_sdk.xdr.trust_line_entry_ext.**TrustLineEntryExt**(*v*, *v1=None*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    case 1:
        struct
        {
            Liabilities liabilities;

            union switch (int v)
            {
            case 0:
                void;
            case 2:
```

(continues on next page)

(continued from previous page)

```
                TrustLineEntryExtensionV2 v2;
            }
            ext;
        } v1;
    }
```

TrustLineEntryExtensionV2

class stellar_sdk.xdr.trust_line_entry_extension_v2.**TrustLineEntryExtensionV2**(*liquidity_pool_use_count*, *ext*)

XDR Source Code:

```
struct TrustLineEntryExtensionV2
{
    int32 liquidityPoolUseCount;

    union switch (int v)
    {
    case 0:
        void;
    }
    ext;
};
```

TrustLineEntryExtensionV2Ext

class stellar_sdk.xdr.trust_line_entry_extension_v2_ext.**TrustLineEntryExtensionV2Ext**(*v*)

XDR Source Code:

```
union switch (int v)
    {
    case 0:
        void;
    }
```

TrustLineEntryV1

class stellar_sdk.xdr.trust_line_entry_v1.**TrustLineEntryV1**(*liabilities*, *ext*)

XDR Source Code:

```
struct
        {
            Liabilities liabilities;

            union switch (int v)
            {
            case 0:
                void;
            case 2:
                TrustLineEntryExtensionV2 v2;
            }
```

(continues on next page)

(continued from previous page)

```
            ext;
        }
```

TrustLineEntryV1Ext

class stellar_sdk.xdr.trust_line_entry_v1_ext.**TrustLineEntryV1Ext**(*v*, *v2=None*)

XDR Source Code:

```
union switch (int v)
            {
            case 0:
                void;
            case 2:
                TrustLineEntryExtensionV2 v2;
            }
```

TrustLineFlags

class stellar_sdk.xdr.trust_line_flags.**TrustLineFlags**(**values*)

XDR Source Code:

```
enum TrustLineFlags
{
    // issuer has authorized account to perform transactions with its credit
    AUTHORIZED_FLAG = 1,
    // issuer has authorized account to maintain and reduce liabilities for its
    // credit
    AUTHORIZED_TO_MAINTAIN_LIABILITIES_FLAG = 2,
    // issuer has specified that it may clawback its credit, and that claimable
    // balances created with its credit may also be clawed back
    TRUSTLINE_CLAWBACK_ENABLED_FLAG = 4
};
```

TxAdvertVector

class stellar_sdk.xdr.tx_advert_vector.**TxAdvertVector**(*tx_advert_vector*)

XDR Source Code:

```
typedef Hash TxAdvertVector<TX_ADVERT_VECTOR_MAX_SIZE>;
```

TxDemandVector

class stellar_sdk.xdr.tx_demand_vector.**TxDemandVector**(*tx_demand_vector*)

XDR Source Code:

```
typedef Hash TxDemandVector<TX_DEMAND_VECTOR_MAX_SIZE>;
```

TxSetComponent

class stellar_sdk.xdr.tx_set_component.**TxSetComponent**(*type*, *txs_maybe_discounted_fee=None*)

XDR Source Code:

```
union TxSetComponent switch (TxSetComponentType type)
{
case TXSET_COMP_TXS_MAYBE_DISCOUNTED_FEE:
  struct
  {
    int64* baseFee;
    TransactionEnvelope txs<>;
  } txsMaybeDiscountedFee;
};
```

TxSetComponentTxsMaybeDiscountedFee

class stellar_sdk.xdr.tx_set_component_txs_maybe_discounted_fee.**TxSetComponentTxsMaybeDiscountedFee**(*base*
txs)

XDR Source Code:

```
struct
  {
    int64* baseFee;
    TransactionEnvelope txs<>;
  }
```

TxSetComponentType

class stellar_sdk.xdr.tx_set_component_type.**TxSetComponentType**(**values*)

XDR Source Code:

```
enum TxSetComponentType
{
  // txs with effective fee <= bid derived from a base fee (if any).
  // If base fee is not specified, no discount is applied.
  TXSET_COMP_TXS_MAYBE_DISCOUNTED_FEE = 0
};
```

UInt128Parts

class stellar_sdk.xdr.u_int128_parts.**UInt128Parts**(*hi*, *lo*)

XDR Source Code:

```
struct UInt128Parts {
    uint64 hi;
    uint64 lo;
};
```

UInt256Parts

class stellar_sdk.xdr.u_int256_parts.**UInt256Parts**(*hi_hi*, *hi_lo*, *lo_hi*, *lo_lo*)

XDR Source Code:

```
struct UInt256Parts {
    uint64 hi_hi;
    uint64 hi_lo;
    uint64 lo_hi;
    uint64 lo_lo;
};
```

Uint256

class stellar_sdk.xdr.uint256.**Uint256**(*uint256*)

XDR Source Code:

```
typedef opaque uint256[32];
```

Uint32

class stellar_sdk.xdr.uint32.**Uint32**(*uint32*)

XDR Source Code:

```
typedef unsigned int uint32;
```

Uint64

class stellar_sdk.xdr.uint64.**Uint64**(*uint64*)

XDR Source Code:

```
typedef unsigned hyper uint64;
```

UnsignedHyper

class stellar_sdk.xdr.base.**UnsignedHyper**(*value*)

UnsignedInteger

class stellar_sdk.xdr.base.**UnsignedInteger**(*value*)

UpgradeEntryMeta

class stellar_sdk.xdr.upgrade_entry_meta.**UpgradeEntryMeta**(*upgrade*, *changes*)

XDR Source Code:

```
struct UpgradeEntryMeta
{
    LedgerUpgrade upgrade;
    LedgerEntryChanges changes;
};
```

UpgradeType

class stellar_sdk.xdr.upgrade_type.**UpgradeType**(*upgrade_type*)

XDR Source Code:

```
typedef opaque UpgradeType<128>;
```

Value

class stellar_sdk.xdr.value.**Value**(*value*)

XDR Source Code:

```
typedef opaque Value<>;
```

Constants

stellar_sdk.xdr.constants.**AUTH_MSG_FLAG_FLOW_CONTROL_BYTES_REQUESTED: int = 200**

const AUTH_MSG_FLAG_FLOW_CONTROL_BYTES_REQUESTED = 200;

stellar_sdk.xdr.constants.**CONTRACT_COST_COUNT_LIMIT: int = 1024**

const CONTRACT_COST_COUNT_LIMIT = 1024;

stellar_sdk.xdr.constants.**LIQUIDITY_POOL_FEE_V18: int = 30**

const LIQUIDITY_POOL_FEE_V18 = 30;

stellar_sdk.xdr.constants.**MASK_ACCOUNT_FLAGS: int = 7**

const MASK_ACCOUNT_FLAGS = 0x7;

stellar_sdk.xdr.constants.**MASK_ACCOUNT_FLAGS_V17: int = 15**

const MASK_ACCOUNT_FLAGS_V17 = 0xF;

stellar_sdk.xdr.constants.**MASK_CLAIMABLE_BALANCE_FLAGS: int = 1**

const MASK_CLAIMABLE_BALANCE_FLAGS = 0x1;

stellar_sdk.xdr.constants.**MASK_LEDGER_HEADER_FLAGS: int = 7**

const MASK_LEDGER_HEADER_FLAGS = 0x7;

stellar_sdk.xdr.constants.**MASK_OFFERENTRY_FLAGS: int = 1**

const MASK_OFFERENTRY_FLAGS = 1;

stellar_sdk.xdr.constants.**MASK_TRUSTLINE_FLAGS: int = 1**

const MASK_TRUSTLINE_FLAGS = 1;

stellar_sdk.xdr.constants.**MASK_TRUSTLINE_FLAGS_V13: int = 3**

const MASK_TRUSTLINE_FLAGS_V13 = 3;

stellar_sdk.xdr.constants.**MASK_TRUSTLINE_FLAGS_V17: int = 7**

const MASK_TRUSTLINE_FLAGS_V17 = 7;

stellar_sdk.xdr.constants.**MAX_OPS_PER_TX: int = 100**

const MAX_OPS_PER_TX = 100;

stellar_sdk.xdr.constants.**MAX_SIGNERS: int = 20**

const MAX_SIGNERS = 20;

stellar_sdk.xdr.constants.**SCSYMBOL_LIMIT: int = 32**

const SCSYMBOL_LIMIT = 32;

stellar_sdk.xdr.constants.**SC_SPEC_DOC_LIMIT: int = 1024**

const SC_SPEC_DOC_LIMIT = 1024;

stellar_sdk.xdr.constants.**TX_ADVERT_VECTOR_MAX_SIZE: int = 1000**

const TX_ADVERT_VECTOR_MAX_SIZE = 1000;

stellar_sdk.xdr.constants.**TX_DEMAND_VECTOR_MAX_SIZE: int = 1000**

const TX_DEMAND_VECTOR_MAX_SIZE = 1000;

CHAPTER

THREE

STELLAR-MODEL

stellar-model allows you to parse the JSON returned by Stellar Horizon into the Python models, click here for more information.

CHAPTER

FOUR

LINKS

- Document: https://stellar-sdk.readthedocs.io
- Code: https://github.com/StellarCN/py-stellar-base
- Examples: https://github.com/StellarCN/py-stellar-base/tree/main/examples
- Issue tracker: https://github.com/StellarCN/py-stellar-base/issues
- License: Apache License 2.0
- Releases: https://pypi.org/project/stellar-sdk/

CHAPTER

FIVE

THANKS

This document is based on Stellar JavaScript SDK documentation. Thank you to all the people who have already contributed to Stellar ecosystem!

CHAPTER

SIX

GENINDEX

PYTHON MODULE INDEX

S

INDEX

A

B

C

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

X