
aioxrpy

Maciej Janiszewski

Apr 21, 2020

CONTENTS

1	Introduction	1
1.1	Features	1
1.2	Getting Started	1
1.2.1	Submitting your first transaction	1
1.2.2	Example code	3
2	API	5
2.1	Addresses	5
2.2	Decimals	5
2.3	Definitions	5
2.4	Exceptions	11
2.5	Hash	12
2.6	Keys	12
2.7	RPC	13
2.8	Serializer	13
3	Changelog	17
3.1	1.0.0 (08.04.2020)	17
4	Indices and tables	19
	Python Module Index	21
	Index	23

INTRODUCTION

Ripple blockchain library for Python.

1.1 Features

1. Async JSON-RPC client.
2. Signing and verifying transactions using private and public keys.
3. Support for signing transactions with multiple keys.
4. Serializer and deserializer for Ripple objects.

1.2 Getting Started

This guide step-by-step explains how to use aioxrpy library to submit your first transaction. Complete example is available at the end of this chapter. Before we begin, please make sure that you have a rippled node running in [stand-alone mode](#) with RPC port exposed and that `aioxrpy` package is installed.

On macOS:

```
$ docker run -d --name ripple-regtest -p 5005:5005 ulamlabs/ripple-regtest
$ pip install aioxrpy
```

1.2.1 Submitting your first transaction

When running Ripple in stand-alone mode, a new genesis ledger is created. A hardcoded genesis address holds all 100 billion XRP. Let's start by initializing a `RippleKey` object using master seed for that address:

```
master = RippleKey(private_key='snoPBrXtMeMyMHUVTgbuqAfg1SUTb')
```

For a Ripple account to be active, it needs to be funded with minimum reserve amount. If you're submitting your transaction against a regtest node, the minimum amount is 200 XRP. You won't be able to spend it as it must remain on your account for it to remain active.

Let's generate the keys for our new account:

```
destination = RippleKey()
```

You can initialize a `RippleKey` instance with either a public, private key or none. With just a public key, you can't sign a transaction, but you can verify a transaction signature. Private key accepts either a master seed to derive the key from or a private key itself.

Ripple transaction is essentially an action executed on the blockchain. A type of action is determined by `TransactionType` field. This can be either a payment, change to the account (ex. changing the keys, nickname and parameters). If a transaction creates a new object on the ledger, reserve amount will be increased. On genesis ledger, that amount is 50 XRP per object.

Transaction object needs to contain at least these fields:

- *Fee* - transaction fee, in drops,
- *Account* - account you're sending the funds from,
- *Sequence* - determines an order in which transactions should be submitted,
- *SigningPubKey* - specifies public key transaction was signed with. In case of multi-signed transaction, it's still required but it should be left empty.
- *TransactionType* - type of transaction.

While *Flags* field is optional, it's recommended to pass `tfFullyCanonicalSig` value. This protects the transaction from a malicious actor being able to modify the signature. The issue is more thoroughly explained [here](#).

Depending on a transaction type, additional fields might be required. Let's focus on a payment in this example.

- *Amount* - amount of XRP (in drops) or issued currency sent in this transaction,
- *Destination* - address which will receive these funds.

Depending on whether the transaction is signed by a single or multiple keys you need to also pass one of these fields:

- *TxnSignature* - signature for transaction in case of single signature,
- *Signers* - contains a list of signer objects, sorted by account ID (account name in binary format).

RPC class contains helper methods (`sign_and_submit`, `multisign_and_submit`) that will sign and submit it to the node for you. These will also set *Sequence* field for you.

```
rpc = RippleJsonRpc('http://localhost:5005')
reserve = await rpc.get_reserve()
fee = await rpc.fee()

tx = {
    'Account': master.to_account(),
    'Flags': RippleTransactionFlags.FullyCanonicalSig,
    'TransactionType': RippleTransactionType.Payment,
    'Amount': decimals.xrp_to_drops(reserve.base),
    'Destination': destination.to_account(),
    'Fee': fee.minimum
}

# post TX blob to rippled JSON-RPC
result = await rpc.sign_and_submit(tx, master)
```

`result` should contain the response from RPC node if transaction was successfully submitted. Otherwise, the last line will throw an exception.

1.2.2 Example code

Complete example code:

```
import asyncio

from aioxrpy import decimals
from aioxrpy.definitions import RippleTransactionType, RippleTransactionFlags
from aioxrpy.keys import RippleKey
from aioxrpy.rpc import RippleJsonRpc

async def example():
    rpc = RippleJsonRpc('http://localhost:5005')
    reserve = await rpc.get_reserve()
    fee = await rpc.fee()

    master = RippleKey(private_key='snoPBrXtMeMyMHUVTgbuqAfg1SUTb')
    destination = RippleKey()

    tx = {
        'Account': master.to_account(),
        'Flags': RippleTransactionFlags.FullyCanonicalSig,
        'TransactionType': RippleTransactionType.Payment,
        'Amount': decimals.xrp_to_drops(reserve.base),
        'Destination': destination.to_account(),
        'Fee': fee.minimum
    }

    # post TX blob to rippled JSON-RPC
    result = await rpc.sign_and_submit(tx, master)
    print(result)

asyncio.get_event_loop().run_until_complete(example())
```


2.1 Addresses

`aioxrpy.address.decode_address` (*address*)
Decodes base58-encoded Ripple account ID

Return type `bytes`

`aioxrpy.address.encode_address` (*value*)
Encodes Ripple account ID using base58

Return type `str`

2.2 Decimals

`aioxrpy.decimals.drops_to_xrp` (*amount*)

`aioxrpy.decimals.xrp_to_drops` (*amount*)

2.3 Definitions

Ripple type and field definitions

```
class aioxrpy.definitions.RippleField(name, is_serialized, is_signing_field, is_vl_encoded,  
                                     nth, type_)
```

```
    property field_id
```

```
    classmethod from_definition(name, definition)
```

```
    is_serialized: bool = None
```

```
    is_signing_field: bool = None
```

```
    is_vl_encoded: bool = None
```

```
    name: str = None
```

```
    nth: int = None
```

```
    type_: RippleType = None
```

```
class aioxrpy.definitions.RippleLedgerEntryType  
    An enumeration.
```

```
AccountRoot = 97
Amendments = 102
Any = -3
Check = 67
Child = -2
Contract = 99
DepositPreauth = 112
DirectoryNode = 100
Escrow = 117
FeeSettings = 115
GeneratorMap = 103
Invalid = -1
LedgerHashes = 104
Nickname = 110
Offer = 111
PayChannel = 120
RippleState = 114
SignerList = 83
Ticket = 84
```

```
class aioxrpy.definitions.RippleTransactionFlags
    An enumeration.
```

```
    FullyCanonicalSig = 2147483648
```

```
class aioxrpy.definitions.RippleTransactionHashPrefix
    An enumeration.
```

```
    HASH_TX_ID = b'TXN\x00'
```

```
    HASH_TX_SIGN = b'STX\x00'
```

```
    HASH_TX_SIGN_MULTI = b'SMT\x00'
```

```
class aioxrpy.definitions.RippleTransactionResult
    An enumeration.
```

```
    tecCLAIM = 100
```

```
    tecCRYPTOCONDITION_ERROR = 146
```

```
    tecDIR_FULL = 121
```

```
    tecDST_TAG_NEEDED = 143
```

```
    tecDUPLICATE = 149
```

```
    tecEXPIRED = 148
```

```
    tecFAILED_PROCESSING = 105
```

```
    tecFROZEN = 137
```

tecHAS_OBLIGATIONS = 151
tecINSUFFICIENT_RESERVE = 141
tecINSUFF_FEE = 136
tecINSUF_RESERVE_LINE = 122
tecINSUF_RESERVE_OFFER = 123
tecINTERNAL = 144
tecINVARIANT_FAILED = 147
tecKILLED = 150
tecNEED_MASTER_KEY = 142
tecNO_ALTERNATIVE_KEY = 130
tecNO_AUTH = 134
tecNO_DST = 124
tecNO_DST_INSUF_XRP = 125
tecNO_ENTRY = 140
tecNO_ISSUER = 133
tecNO_LINE = 135
tecNO_LINE_INSUF_RESERVE = 126
tecNO_LINE_REDUNDANT = 127
tecNO_PERMISSION = 139
tecNO_REGULAR_KEY = 131
tecNO_TARGET = 138
tecOVERSIZE = 145
tecOWNERS = 132
tecPATH_DRY = 128
tecPATH_PARTIAL = 101
tecTOO_SOON = 152
tecUNFUNDED = 129
tecUNFUNDED_ADD = 102
tecUNFUNDED_OFFER = 103
tecUNFUNDED_PAYMENT = 104
tefALREADY = -198
tefBAD_ADD_AUTH = -197
tefBAD_AUTH = -196
tefBAD_AUTH_MASTER = -183
tefBAD_LEDGER = -195
tefBAD_QUORUM = -185

tefBAD_SIGNATURE = -186
tefCREATED = -194
tefEXCEPTION = -193
tefFAILURE = -199
tefINTERNAL = -192
tefINVARIANT_FAILED = -182
tefMASTER_DISABLED = -188
tefMAX_LEDGER = -187
tefNOT_MULTI_SIGNING = -184
tefNO_AUTH_REQUIRED = -191
tefPAST_SEQ = -190
tefTOO_BIG = -181
tefWRONG_PRIOR = -189
telBAD_DOMAIN = -398
telBAD_PATH_COUNT = -397
telBAD_PUBLIC_KEY = -396
telCAN_NOT_QUEUE = -392
telCAN_NOT_QUEUE_BALANCE = -391
telCAN_NOT_QUEUE_BLOCKED = -389
telCAN_NOT_QUEUE_BLOCKS = -390
telCAN_NOT_QUEUE_FEE = -388
telCAN_NOT_QUEUE_FULL = -387
telFAILED_PROCESSING = -395
telINSUF_FEE_P = -394
telLOCAL_ERROR = -399
telNO_DST_PARTIAL = -393
temBAD_AMOUNT = -298
temBAD_CURRENCY = -297
temBAD_EXPIRATION = -296
temBAD_FEE = -295
temBAD_ISSUER = -294
temBAD_LIMIT = -293
temBAD_OFFER = -292
temBAD_PATH = -291
temBAD_PATH_LOOP = -290
temBAD_QUORUM = -271

```
temBAD_REGKEY = -289
temBAD_SEND_XRP_LIMIT = -288
temBAD_SEND_XRP_MAX = -287
temBAD_SEND_XRP_NO_DIRECT = -286
temBAD_SEND_XRP_PARTIAL = -285
temBAD_SEND_XRP_PATHS = -284
temBAD_SEQUENCE = -283
temBAD_SIGNATURE = -282
temBAD_SIGNER = -272
temBAD_SRC_ACCOUNT = -281
temBAD_TICK_SIZE = -269
temBAD_TRANSFER_RATE = -280
temBAD_WEIGHT = -270
temCANNOT_PREAUTH_SELF = -267
temDISABLED = -273
temDST_IS_SRC = -279
temDST_NEEDED = -278
temINVALID = -277
temINVALID_ACCOUNT_ID = -268
temINVALID_FLAG = -276
temMALFORMED = -299
temREDUNDANT = -275
temRIPPLE_EMPTY = -274
temUNCERTAIN = -266
temUNKNOWN = -265
terFUNDS_SPENT = -98
terINSUF_FEE_B = -97
terLAST = -91
terNO_ACCOUNT = -96
terNO_AUTH = -95
terNO_LINE = -94
terNO_RIPPLE = -90
terOWNERS = -93
terPRE_SEQ = -92
terQUEUED = -89
terRETRY = -99
```

```
tesSUCCESS = 0
```

```
class aioxrpy.definitions.RippleTransactionResultCategory
```

```
Enum containing Ripple transaction categories. https://xrpl.org/tec-codes.html
```

The original abbreviations for transaction result categories (tec, tel codes) are not expanded anywhere so I had to get creative with the names.

```
CostlyFailure = 'tec'
```

```
Failure = 'tef'
```

```
LocalFailure = 'tel'
```

```
MalformedFailure = 'tem'
```

```
RetriableFailure = 'ter'
```

```
Success = 'tes'
```

```
class aioxrpy.definitions.RippleTransactionType
```

```
An enumeration.
```

```
AccountDelete = 21
```

```
AccountSet = 3
```

```
CheckCancel = 18
```

```
CheckCash = 17
```

```
CheckCreate = 16
```

```
Contract = 9
```

```
DepositPreauth = 19
```

```
EnableAmendment = 100
```

```
EscrowCancel = 4
```

```
EscrowCreate = 1
```

```
EscrowFinish = 2
```

```
Invalid = -1
```

```
NickNameSet = 6
```

```
OfferCancel = 8
```

```
OfferCreate = 7
```

```
Payment = 0
```

```
PaymentChannelClaim = 15
```

```
PaymentChannelCreate = 13
```

```
PaymentChannelFund = 14
```

```
SetFee = 101
```

```
SetRegularKey = 5
```

```
SignerListSet = 12
```

```
TicketCancel = 11
```

```
TicketCreate = 10
```

```

    TrustSet = 20
class aioxrpy.definitions.RippleType
    An enumeration.
    AccountID = 8
    Amount = 6
    Blob = 7
    Done = -1
    Hash128 = 4
    Hash160 = 17
    Hash256 = 5
    LedgerEntry = 10002
    NotPresent = 0
    PathSet = 18
    STArray = 15
    STObject = 14
    Transaction = 10001
    UInt16 = 1
    UInt32 = 2
    UInt64 = 3
    UInt8 = 16
    Unknown = -2
    Validation = 10003
    Vector256 = 19

```

2.4 Exceptions

```

exception aioxrpy.exceptions.AccountNotFoundException (payload={})
exception aioxrpy.exceptions.InvalidTransactionException (payload={})
exception aioxrpy.exceptions.RippleBaseException (error, payload={})
exception aioxrpy.exceptions.RippleSerializerUnsupportedTypeException (payload={})
exception aioxrpy.exceptions.RippleTransactionCostlyFailureException (error,
                                                                    pay-
                                                                    load={})
exception aioxrpy.exceptions.RippleTransactionException (error, category, pay-
                                                                    load={})
exception aioxrpy.exceptions.RippleTransactionFailureException (error, pay-
                                                                    load={})

```

```
exception aioxrpy.exceptions.RippleTransactionLocalFailureException (error,  
                                                                    pay-  
                                                                    load={})  
exception aioxrpy.exceptions.RippleTransactionMalformedException (error, pay-  
                                                                    load={})  
exception aioxrpy.exceptions.RippleTransactionRetriableException (error, pay-  
                                                                    load={})  
exception aioxrpy.exceptions.UnknownRippleException (payload={})  
exception aioxrpy.exceptions.ValidatedLedgerUnavailableException (payload={})
```

2.5 Hash

```
aioxrpy.hash.first_half_of_sha512 (*data)  
    Returns first 32 bytes of SHA512 hash
```

Return type bytes

```
aioxrpy.hash.hash_transaction (prefix, tx, suffix)  
    Serializes transaction object and returns first half of SHA512 hash
```

Return type bytes

2.6 Keys

```
class aioxrpy.keys.RippleKey (*, private_key=None, public_key=None)  
    RippleKey instance
```

Parameters

- **private_key** (Union[str, bytes, None]) – private key or master seed,
- **public_key** (Optional[bytes]) – public key

If no arguments are passed, new key will be generated.

```
sign (data, sigencode=<function sigencode_der>, **kwargs)  
    Signs the provided data and returns a canonical signature
```

Return type str

```
sign_tx (tx, *, multi_sign=False, **kwargs)
```

Return type str

```
to_account ()
```

Returns base58-encoded RIPEMD-160 hash of SHA256 hash of public key, which is used as an account name on Ripple ledger.

For example: rHb9CJAWyB4rj91VRWn96DkukG4bwdtyTh

Return type str

```
to_public ()
```

Returns public key encoded in compressed format.

Return type bytes

```
verify (data, signature, *, sigdecode=<function sigdecode_der>, **kwargs)
```


Return type bool

verify_tx (*tx*, *signature*, *, *multi_sign=False*, ***kwargs*)

Return type bool

`aioxrpy.keys.make_canonical` (*r*, *s*, *order*)

Makes ecdsa signature canonical

`aioxrpy.keys.signing_key_from_seed` (*encoded_seed*)

Derives SigningKey from master seed.

Reference: https://ripple.com/wiki/Account_Family#Root_Key_.28GenerateRootDeterministicKey.29

Return type SigningKey

2.7 RPC

2.8 Serializer

class `aioxrpy.serializer.AccountIDSerializer`

Serializer for AccountID type

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, str]

serialize (*value*)

Returns byte-encoded value

Return type bytes

class `aioxrpy.serializer.AmountSerializer`

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

scale_to_xrp_amount (*value*)

serialize (*value*)

Returns byte-encoded value

class `aioxrpy.serializer.ArraySerializer`

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

serialize (*value*)

Returns byte-encoded value

class `aioxrpy.serializer.BaseSerializer`

abstract deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, Any]

abstract serialize (*value*)

Returns byte-encoded value

Return type bytes

class aioxrpy.serializer.**BasicTypeSerializer** (*fmt=""*)

Serializes basic types such as integers and floats using `struct` module

Params *fmt* format string, please refer to documentation for `struct` module

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, Any]

serialize (*value*)

Returns byte-encoded value

Return type bytes

class aioxrpy.serializer.**BlobSerializer**

Serializer for blob format

Reference: <https://xrpl.org/serialization.html#length-prefixing>

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, bytes]

serialize (*value*)

Returns byte-encoded value

Return type bytes

class aioxrpy.serializer.**CurrencySerializer**

Currency code serializer

[12 reserved bytes][3-character currency code][5 reserved bytes]

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, str]

serialize (*value*)

Returns byte-encoded value

Return type bytes

class aioxrpy.serializer.**HashSerializer** (*length*)

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

serialize (*value*)

Returns byte-encoded value

class aioxrpy.serializer.**ObjectSerializer**

To serialize an object to Ripple format, we need to follow these steps:

1. Convert each field data to binary format
2. Sort fields in “canonical order”
3. Prefix each field with a field ID.

4. Concatenate fields (with prefixes) in their sorted order

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

Return type Tuple[int, Dict]

serialize (*value*)

Returns byte-encoded value

Return type bytes

class aioxrpy.serializer.**PathSetSerializer**

deserialize (*value*)

Returns a tuple containing length of original data and deserialized value

serialize (*value*)

Returns byte-encoded value

aioxrpy.serializer.**decode** (*key*, *binary*)

aioxrpy.serializer.**deserialize** (*binary*)

Deserializes object from binary format. Shorthand for `ObjectSerializer().deserialize(binary)`

Return type Dict

aioxrpy.serializer.**encode** (*key*, *value*)

aioxrpy.serializer.**lookup_field** (*binary*)

aioxrpy.serializer.**serialize** (*obj*)

Serializes object to binary format. Shorthand for `ObjectSerializer().serialize(obj)`

Return type bytes

CHANGELOG

3.1 1.0.0 (08.04.2020)

- Initial release

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

- `aiorpy.address`, 5
- `aiorpy.decimals`, 5
- `aiorpy.definitions`, 5
- `aiorpy.exceptions`, 11
- `aiorpy.hash`, 12
- `aiorpy.keys`, 12
- `aiorpy.serializer`, 13

A

AccountDelete (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 AccountID (*aioxrpy.definitions.RippleType attribute*), 11
 AccountIDSerializer (*class in aioxrpy.serializer*), 13
 AccountNotFoundException, 11
 AccountRoot (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 5
 AccountSet (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 aioxrpy.address (*module*), 5
 aioxrpy.decimals (*module*), 5
 aioxrpy.definitions (*module*), 5
 aioxrpy.exceptions (*module*), 11
 aioxrpy.hash (*module*), 12
 aioxrpy.keys (*module*), 12
 aioxrpy.serializer (*module*), 13
 Amendments (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 Amount (*aioxrpy.definitions.RippleType attribute*), 11
 AmountSerializer (*class in aioxrpy.serializer*), 13
 Any (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 ArraySerializer (*class in aioxrpy.serializer*), 13

B

BaseSerializer (*class in aioxrpy.serializer*), 13
 BasicTypeSerializer (*class in aioxrpy.serializer*), 14
 Blob (*aioxrpy.definitions.RippleType attribute*), 11
 BlobSerializer (*class in aioxrpy.serializer*), 14

C

Check (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 CheckCancel (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 CheckCash (*aioxrpy.definitions.RippleTransactionType attribute*), 10

CheckCreate (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 Child (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 Contract (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 Contract (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 CostlyFailure (*aioxrpy.definitions.RippleTransactionResultCategory attribute*), 10
 CurrencySerializer (*class in aioxrpy.serializer*), 14

D

decode () (*in module aioxrpy.serializer*), 15
 decode_address () (*in module aioxrpy.address*), 5
 DepositPreauth (*aioxrpy.definitions.RippleLedgerEntryType attribute*), 6
 DepositPreauth (*aioxrpy.definitions.RippleTransactionType attribute*), 10
 deserialize () (*aioxrpy.serializer.AccountIDSerializer method*), 13
 deserialize () (*aioxrpy.serializer.AmountSerializer method*), 13
 deserialize () (*aioxrpy.serializer.ArraySerializer method*), 13
 deserialize () (*aioxrpy.serializer.BaseSerializer method*), 13
 deserialize () (*aioxrpy.serializer.BasicTypeSerializer method*), 14
 deserialize () (*aioxrpy.serializer.BlobSerializer method*), 14
 deserialize () (*aioxrpy.serializer.CurrencySerializer method*), 14
 deserialize () (*aioxrpy.serializer.HashSerializer method*), 14
 deserialize () (*aioxrpy.serializer.ObjectSerializer method*), 15
 deserialize () (*aioxrpy.serializer.PathSetSerializer method*), 15
 deserialize () (*in module aioxrpy.serializer*), 15
 DirectoryNode (*aioxrpy.definitions.RippleLedgerEntryType*

attribute), 6
 Done (*aioxrpy.definitions.RippleType* attribute), 11
 drops_to_xrp() (in module *aioxrpy.decimals*), 5

E

EnableAmendment (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 encode() (in module *aioxrpy.serializer*), 15
 encode_address() (in module *aioxrpy.address*), 5
 Escrow (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 EscrowCancel (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 EscrowCreate (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 EscrowFinish (*aioxrpy.definitions.RippleTransactionType* attribute), 10

F

Failure (*aioxrpy.definitions.RippleTransactionResultCategory* attribute), 10
 FeeSettings (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 field_id() (*aioxrpy.definitions.RippleField* property), 5
 first_half_of_sha512() (in module *aioxrpy.hash*), 12
 from_definition() (*aioxrpy.definitions.RippleField* class method), 5
 FullyCanonicalSig (*aioxrpy.definitions.RippleTransactionFlags* attribute), 6

G

GeneratorMap (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6

H

Hash128 (*aioxrpy.definitions.RippleType* attribute), 11
 Hash160 (*aioxrpy.definitions.RippleType* attribute), 11
 Hash256 (*aioxrpy.definitions.RippleType* attribute), 11
 hash_transaction() (in module *aioxrpy.hash*), 12
 HASH_TX_ID (*aioxrpy.definitions.RippleTransactionHashPrefix* attribute), 6
 HASH_TX_SIGN (*aioxrpy.definitions.RippleTransactionHashPrefix* attribute), 6
 HASH_TX_SIGN_MULTII (*aioxrpy.definitions.RippleTransactionHashPrefix* attribute), 6
 HashSerializer (class in *aioxrpy.serializer*), 14

I

Invalid (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 Invalid (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 InvalidTransactionException, 11
 is_serialized (*aioxrpy.definitions.RippleField* attribute), 5
 is_signing_field (*aioxrpy.definitions.RippleField* attribute), 5
 is_vl_encoded (*aioxrpy.definitions.RippleField* attribute), 5

L

LedgerEntry (*aioxrpy.definitions.RippleType* attribute), 11
 LedgerHashes (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 LocalFailure (*aioxrpy.definitions.RippleTransactionResultCategory* attribute), 10
 lookup_field() (in module *aioxrpy.serializer*), 15

M

make_canonical() (in module *aioxrpy.keys*), 13
 MalformedFailure (*aioxrpy.definitions.RippleTransactionResultCategory* attribute), 10

N

name (*aioxrpy.definitions.RippleField* attribute), 5
 Nickname (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 NickNameSet (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 NotPresent (*aioxrpy.definitions.RippleType* attribute), 11
 nth (*aioxrpy.definitions.RippleField* attribute), 5

O

ObjectSerializer (class in *aioxrpy.serializer*), 14
 Offer (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 OfferCancel (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 OfferCreate (*aioxrpy.definitions.RippleTransactionType* attribute), 10

P

PathSet (*aioxrpy.definitions.RippleType* attribute), 11
 PathSetSerializer (class in *aioxrpy.serializer*), 15
 PayChannel (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 Payment (*aioxrpy.definitions.RippleTransactionType* attribute), 10

PaymentChannelClaim
 (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

PaymentChannelCreate
 (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

PaymentChannelFund
 (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

R

RetriableFailure (*aioxrpy.definitions.RippleTransactionResultCategory*
 attribute), 10

RippleBaseException, 11

RippleField (*class in aioxrpy.definitions*), 5

RippleKey (*class in aioxrpy.keys*), 12

RippleLedgerEntryType (*class in*
 aioxrpy.definitions), 5

RippleSerializerUnsupportedTypeException, 11

RippleState (*aioxrpy.definitions.RippleLedgerEntryType*
 attribute), 6

RippleTransactionCostlyFailureException,
 11

RippleTransactionException, 11

RippleTransactionFailureException, 11

RippleTransactionFlags (*class in*
 aioxrpy.definitions), 6

RippleTransactionHashPrefix (*class in*
 aioxrpy.definitions), 6

RippleTransactionLocalFailureException,
 11

RippleTransactionMalformedException, 12

RippleTransactionResult (*class in*
 aioxrpy.definitions), 6

RippleTransactionResultCategory (*class in*
 aioxrpy.definitions), 10

RippleTransactionRetriableException, 12

RippleTransactionType (*class in*
 aioxrpy.definitions), 10

RippleType (*class in aioxrpy.definitions*), 11

serialize() (*aioxrpy.serializer.AmountSerializer*
 method), 13

serialize() (*aioxrpy.serializer.AmountSerializer*
 method), 13

serialize() (*aioxrpy.serializer.ArraySerializer*
 method), 13

serialize() (*aioxrpy.serializer.BaseSerializer*
 method), 13

serialize() (*aioxrpy.serializer.BasicTypeSerializer*
 method), 14

serialize() (*aioxrpy.serializer.BlobSerializer*
 method), 14

serialize() (*aioxrpy.serializer.CurrencySerializer*
 method), 14

serialize() (*aioxrpy.serializer.HashSerializer*
 method), 14

serialize() (*aioxrpy.serializer.ObjectSerializer*
 method), 15

serialize() (*aioxrpy.serializer.PathSetSerializer*
 method), 15

serialize() (*in module aioxrpy.serializer*), 15

SetFee (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

SetRegularKey (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

sign() (*aioxrpy.keys.RippleKey* *method*), 12

sign_tx() (*aioxrpy.keys.RippleKey* *method*), 12

SignerList (*aioxrpy.definitions.RippleLedgerEntryType*
 attribute), 6

SignerListSet (*aioxrpy.definitions.RippleTransactionType*
 attribute), 10

signing_key_from_seed() (*in module*
 aioxrpy.keys), 13

STArray (*aioxrpy.definitions.RippleType* *attribute*), 11

STObject (*aioxrpy.definitions.RippleType* *attribute*), 11

Success (*aioxrpy.definitions.RippleTransactionResultCategory*
 attribute), 10

T

tecCLAIM (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecCRYPTOCONDITION_ERROR
 (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecDIR_FULL (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecDST_TAG_NEEDED
 (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecDUPLICATE (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecEXPIRED (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecFAILED_PROCESSING
 (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecFROZEN (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

techAS_OBLIGATIONS
 (*aioxrpy.definitions.RippleTransactionResult*
 attribute), 6

tecINSUF_RESERVE_LINE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecINSUF_RESERVE_OFFER (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecINSUFF_FEE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecINSUFFICIENT_RESERVE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecINTERNAL (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecINVARIANT_FAILED (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecKILLED (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNEED_MASTER_KEY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_ALTERNATIVE_KEY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_AUTH (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_DST (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_DST_INSUF_XRP (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_ENTRY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_ISSUER (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_LINE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_LINE_INSUF_RESERVE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_LINE_REDUNDANT (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_PERMISSION (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_REGULAR_KEY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecNO_TARGET (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecOVERSIZE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecOWNERS (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecPATH_DRY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecPATH_PARTIAL (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecTOO_SOON (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecUNFUNDED (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecUNFUNDED_ADD (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecUNFUNDED_OFFER (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tecUNFUNDED_PAYMENT (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefALREADY (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_ADD_AUTH (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_AUTH (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_AUTH_MASTER (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_LEDGER (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_QUORUM (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefBAD_SIGNATURE (*aioxrpy.definitions.RippleTransactionResult attribute*), 7
 tefCREATED (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefEXCEPTION (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefFAILURE (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefINTERNAL (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefINVARIANT_FAILED (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefMASTER_DISABLED (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefMAX_LEDGER (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefNO_AUTH_REQUIRED (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefNOT_MULTI_SIGNING (*aioxrpy.definitions.RippleTransactionResult attribute*), 8
 tefPAST_SEQ (*aioxrpy.definitions.RippleTransactionResult attribute*), 8

attribute), 8
 tefTOO_BIG (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 tefWRONG_PRIOR (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telBAD_DOMAIN (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telBAD_PATH_COUNT
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telBAD_PUBLIC_KEY
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE_BALANCE
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE_BLOCKED
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE_BLOCKS
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE_FEE
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telCAN_NOT_QUEUE_FULL
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telFAILED_PROCESSING
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telINSUF_FEE_P (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telLOCAL_ERROR (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 telNO_DST_PARTIAL
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_AMOUNT (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_CURRENCY (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_EXPIRATION
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_FEE (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_ISSUER (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_LIMIT (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_OFFER (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_PATH (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_PATH_LOOP (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_QUORUM (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_REGKEY (*aioxrpy.definitions.RippleTransactionResult*
attribute), 8
 temBAD_SEND_XRP_LIMIT
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SEND_XRP_MAX
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SEND_XRP_NO_DIRECT
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SEND_XRP_PARTIAL
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SEND_XRP_PATHS
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SEQUENCE (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SIGNATURE (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SIGNER (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_SRC_ACCOUNT
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_TICK_SIZE (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_TRANSFER_RATE
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temBAD_WEIGHT (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temCANNOT_PREAUTH_SELF
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temDISABLED (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temDST_IS_SRC (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temDST_NEEDED (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temINVALID (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9
 temINVALID_ACCOUNT_ID
 (*aioxrpy.definitions.RippleTransactionResult*
attribute), 9

temINVALID_FLAG (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 temMALFORMED (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 temREDUNDANT (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 temRIPPLE_EMPTY (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 temUNCERTAIN (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 temUNKNOWN (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terFUNDS_SPENT (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terINSUF_FEE_B (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terLAST (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terNO_ACCOUNT (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terNO_AUTH (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terNO_LINE (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terNO_RIPPLE (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terOWNERS (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terPRE_SEQ (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terQUEUED (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 terRETRY (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 tesSUCCESS (*aioxrpy.definitions.RippleTransactionResult* attribute), 9
 Ticket (*aioxrpy.definitions.RippleLedgerEntryType* attribute), 6
 TicketCancel (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 TicketCreate (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 to_account () (*aioxrpy.keys.RippleKey* method), 12
 to_public () (*aioxrpy.keys.RippleKey* method), 12
 Transaction (*aioxrpy.definitions.RippleType* attribute), 11
 TrustSet (*aioxrpy.definitions.RippleTransactionType* attribute), 10
 type_ (*aioxrpy.definitions.RippleField* attribute), 5

U

UInt16 (*aioxrpy.definitions.RippleType* attribute), 11
 UInt32 (*aioxrpy.definitions.RippleType* attribute), 11
 UInt64 (*aioxrpy.definitions.RippleType* attribute), 11