

---

**aioxrpy**

**Maciej Janiszewski**

**Apr 22, 2020**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Features . . . . .	1
1.2	Getting Started . . . . .	1
1.2.1	Submitting your first transaction . . . . .	1
1.2.2	Example code . . . . .	3
<b>2</b>	<b>API</b>	<b>5</b>
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
<b>3</b>	<b>Changelog</b>	<b>17</b>
3.1	1.0.0 (08.04.2020) . . . . .	17
<b>4</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



## 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](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.AccountIDSerializer*  
     *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