← Back home

LUNO

Search...

Authentication

Conventions

Currency

Libraries

Rate Limiting

Security

Accounts >

Beneficiaries >

Market >

Orders >

Luno API (1.0.0)

Download OpenAPI specification: Download

The Luno API provides developers with a wealth of financial information provided through the Luno platform. Through this secure system developers can:

- Create accounts for trading in cryptocurrencies
- Access current and historic cryptocurrency market data
- Submit trade orders and view order status
- Buy and sell Bitcoin and Ethereum
- Send and receive Bitcoin and Ethereum
- Generate Bitcoin and Ethereum wallet addresses

The Luno API brings the world of Bitcoin and Ethereum to your doorstep.

Authentication

Some API calls require your application to authenticate itself. This is done using an API key associated with your account. You can create an API key by visiting the API Keys section on the settings page.

An API key consists of a `key id` and a `key secret`. For example, `cnz2yjswbv3jd` (key id) and `0hydMZDb9HRR3Qq-iqALwZtXLkbLR4fWxtDZvkB9h4I` (key secret).

API requests are authenticated using HTTP basic authentication with the key id as the username and the key secret as the password. A missing, incorrect or revoked key causes error 401 to be returned.

Each API key is granted a set of permissions when it is created. The key can only be used to call the permitted API functions.

Permissions

The following is a list of the possible permissions.

- `Perm_R_Balance = 1` (View balance)
- `Perm_R_Transactions = 2` (View transactions)
- `Perm_W_Send = 4` (Send to any address)
- `Perm_R_Addresses = 8` (View addresses)
- `Perm_W_Addresses = 16` (Create addresses)
- `Perm_R_Orders = 32` (View orders)
- `Perm_W_Orders = 64` (Create orders)
- `Perm_R_Withdrawals = 128` (View withdrawals)
- `Perm_W_Withdrawals = 256` (Create withdrawals)
- `Perm_W_ClientDebit = 8192` (Debit accounts)
- `Perm_W_ClientCredit = 16384` (Credit accounts)
- `Perm_R_Beneficiaries = 32768` (View beneficiaries)
- `Perm_W_Beneficiaries = 65536` (Create and delete beneficiaries)

A set of permissions is represented as the bitwise OR of each permission in the set. For example the set of permissions required to view balances and orders is `Perm_R_Balance | Perm_R_Orders = 33`.

When API keys are created, users can select permission sets that automatically include various permissions for their key. These are listed below.

Permission Set	Included Permissions
Read-only access	View Balance View transactions Send to any address View orders View withdrawals View Beneficiaries
Trading access	View Balance View transactions Send to any address View orders View withdrawals View Beneficiaries Create Orders

Conventions

Timestamps are always represented as an integer number of milliseconds since the `UTC Epoch` (a Unix timestamp).

Prices and volumes are always represented as a decimal strings e.g. "123.3432". Strings are used rather than floats to preserve the precision.

Parameters for POST calls are sent as URL-encoded forms (`application/x-www-form-urlencoded`).

Currency

The following currencies are supported through the Luno market platform. For complete details, please see Fees & features:

- **XBT**: Bitcoin
- **BCH**: Bitcoin Cash
- **ETH**: Ethereum
- **EUR**: Euro
- **GBP**: Pounds
- **LTC**: Litecoin
- **MYR**: Malaysian Ringgit
- **NGN**: Nigerian Naira
- **UGX**: Ugandan Shilling
- **ZAR**: South African Rand
- **XRP**: XRP
- **ZMW**: Zambian Kwacha

The following are examples of currency pairs that are supported through the Luno market platform. For complete details, please see Fees & Features:

- **XBTEUR**
- **XBTZAR**
- **XBTUGX**
- **XBTZMW**
- **ETHXBT**
- **BCHXBT**

The following methods are available for Funds Withdrawal based on the type of currency or currency pair being withdrawn.

Currency:

- **BTC**: Bitcoin
- **BCH**: Bitcoin Cash
- **ETH**: Ethereum
- **LTC**: Litecoin
- **XRP**: XRP

Withdrawal methods:

- **ZAR_EFT**: EFT
- **NAD_EFT**: EFT
- **KES_EFT**: EFT
- **KES_MPESA**: M-Pesa
- **MYR_IBG**: Interbank GIRO / IBFT
- **IDR_LLG**: Bank transfer, Lalu Lintas Giro
- **NGN_EFT**: Bank transfer
- **ZMW_EFT**: Bank transfer
- **SGD_GIRO**: GIRO / FAST
- **SGD_WIRE**: International Wire
- **EUR_SEPA**: SEPA transfer
- **GBP**: Bank transfer
- **UGX_EFT**: Bank transfer

Libraries

The Go library is the recommended way to access the API.

The following libraries were implemented by third parties or are no longer under active development and are listed here for convenience. No support is provided by Luno and they may be out of date. A thorough review of the code is recommended before including them in any project.

- Android
- Haskell
- Java
- Node.js
- PHP
- Python
- Ruby

Rate Limiting

Calls to the Market Data APIs are rate limited to 1 call per second per IP address, with a burst of 5 calls. All other API calls are rate limited to 5 calls per second per customer, with a burst of 25 calls. API call rate limits allow bursts of up to 5 consecutive calls. Exceeding the limit causes HTTP error `Code 429` to be returned.

Security

Always use HTTPS when calling the API. Non-TLS HTTP requests cause error 403 to be returned. Using non-TLS requests can leak your authentication credentials.

Verify that your client validates the server's SSL certificate. Many libraries (e.g. `urllib2` in Python2) do not validate server certificates by default. Failing to verify the server certificate makes your application vulnerable to man-in-the-middle attack.

Accounts

All transactions on the Luno platform operate on *Accounts*. Each Account is denominated in a single currency and contains an ordered list of entries that track its running balance.

Each Account has a separate balance and available balance. The available balance may be lower than the balance if some funds have been reserved (e.g. for an open limit order). Account entries affect the balance and available balance independently.

Account entries are numbered sequentially. It is guaranteed that entries are never reordered or deleted. It is also guaranteed that the core attributes of the entry (the running balances and index) are never modified. Therefore, an Account acts as an append-only log of transactions.

Create account

This request creates an Account for the specified currency. Please note that the balances for the Account will be displayed based on the `asset` value, which is the currency the Account is based on.

Permissions required: `Perm_W_Addresses`

QUERY PARAMETERS

`currency` required	string Example: `currency=XBT` The currency code for the Account you want to create. Please see the Currency section for a detailed list of currencies supported by the Luno platform. Users must be verified to trade currency in order to be able to create an Account. For more information on the verification process, please see How do I verify my identity?. Users have a limit of 4 accounts per currency.
`name` required	string Example: `name=Trading ACC` The label to use for this account

Responses

> 200 OK

POST `/api/1/accounts`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "currency": "string",
    "id": "string",
    "name": "string",
  - "pending": [
      + { … }
    ],
  - "transactions": [
      + { … }
    ]
}
```

Update Account Name

Update the name of an account with a given ID.

Permissions required: `Perm_W_Addresses`

PATH PARAMETERS

`id` required	integer <int64> Example: `12345` Account ID - the unique identifier for the specific Account.

QUERY PARAMETERS

`name` required	string Example: `name=Trading ACC` The label to use for this account

Responses

> 200 OK

PUT `/api/1/accounts/{id}/name`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "success": true
}
```

List pending transactions

Return a list of all transactions that have not completed for the Account.

Pending transactions are not numbered, and may be reordered, deleted or updated at any time.

Permissions required: `Perm_R_Transactions`

PATH PARAMETERS

`id` required	integer <int64> Example: `12345` Account ID

Responses

> 200 OK

GET `/api/1/accounts/{id}/pending`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
   "currency": "string",
   "id": "string",
   "name": "string",
 - "pending": [
     + { … }
   ],
 - "transactions": [
     + { … }
   ]
}
```

List transactions

Return a list of transaction entries from an account.

Transaction entry rows are numbered sequentially starting from 1, where 1 is the oldest entry. The range of rows to return are specified with the `min_row` (inclusive) and `max_row` (exclusive) parameters. At most 1000 rows can be requested per call.

If `min_row` or `max_row` is non-positive, the range wraps around the most recent row. For example, to fetch the 100 most recent rows, use `min_row=-100` and `max_row=0`.

Permissions required: `Perm_R_Transactions`

PATH PARAMETERS

`id` required	integer <int64> Example: `12345` Account ID - the unique identifier for the specific Account.

QUERY PARAMETERS

`min_row` required	integer <int64> Example: `min_row=1` Minimum of the row range to return (inclusive)
`max_row` required	integer <int64> Example: `max_row=1000` Maximum of the row range to return (exclusive)

Responses

> 200 OK

GET `/api/1/accounts/{id}/transactions`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
   "currency": "string",
   "id": "string",
   "name": "string",
 - "pending": [
     + { … }
   ],
 - "transactions": [
     + { … }
   ]
}
```

List balances

The list of all Accounts and their respective balances for the requesting user.

Permissions required: `Perm_R_Balance`

QUERY PARAMETERS

`assets`	Array of strings Example: `assets=XBT,ETH` Only return balances for wallets with these currencies (if not provided, all balances will be returned)

Responses

> 200 OK

GET `/api/1/balance`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
 - "balance": [
     + { … }
   ]
}
```

Beneficiaries

Users are able to access information about their beneficiaries - banks or other financial institutions that are able to receive assets.

List beneficiaries

Returns a list of bank beneficiaries.

Permissions required: `Perm_R_Beneficiaries`

Responses

> 200 OK

GET `/api/1/beneficiaries`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "beneficiaries": [
      + { … }
    ]
}
```

Market

Market data API calls can be accessed by anyone without authentication. The data returned may be cached for up to 1 second. The Streaming API (see below) can be used if lower latency market data is needed.

Get full order book

This request returns a list of all *bids* and *asks* for the currency pair specified in the Order Book.

Ask orders are sorted by price ascending.

Bid orders are sorted by price descending.

Multiple orders at the same price are not aggregated.

Warning: This may return a large amount of data. Users are recommended to use the top 100 bids and asks or the Streaming API.

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` Currency pair of the Orders to retrieve

Responses

> 200 OK

GET `/api/1/orderbook`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "asks": [
      + { … }
    ],
  - "bids": [
      + { … }
    ],
    "timestamp": 0
}
```

Get the top 100 bids and asks

Returns a list of the top 100 *bids* and *asks* for the currency pair specified in the Order Book.

Ask Orders are sorted by price ascending.

Bid Orders are sorted by price descending.

Orders of the same price are aggregated.

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` Currency pair of the Orders to retrieve

Responses

GET `/api/1/orderbook_top`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "asks": [
      + { … }
    ],
  - "bids": [
      + { … }
    ],
    "timestamp": 0
}
```

> 200 OK

Get ticker for currency pair

Returns the latest ticker indicators for the specified currency pair.

Please see the Currency list for the complete list of supported currency pairs.

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` Currency pair

Responses

> 200 OK

GET `/api/1/ticker`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "ask": "string",
    "bid": "string",
    "last_trade": "string",
    "pair": "string",
    "rolling_24_hour_volume": "string",
    "status": "ACTIVE",
    "timestamp": "string"
}
```

List tickers for all currency pairs

Returns the latest ticker indicators from all active Luno exchanges.

Please see the Currency list for the complete list of supported currency pairs.

Responses

> 200 OK

GET `/api/1/tickers`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "tickers": [
      + { … }
    ]
}
```

List trades

Returns a list of the most recent Trades for the specified currency pair in the last 24 hours. At most 100 results are returned per call.

Please see the Currency list for the complete list of supported currency pairs.

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` Currency pair
`since`	string <timestamp> Example: `since=1520438111000` Fetch trades executed after this time, specified as a Unix timestamp in milliseconds.

Responses

> 200 OK

GET `/api/1/trades`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "trades": [
      + { … }
    ]
}
```

Orders

Trading on the market is done by submitting Orders. After a new Order has been created, it is submitted for processing by the order matching engine. The Order then either matches against an existing order in the order book and is filled or it rests in the order book until it is stopped.

Click here to read more about how order matching works..

Get fee information

Returns the fees and 30 day trading volume (as of midnight) for a given currency pair. For complete details, please see Fees & Features.

Permissions required: `Perm_R_Orders`

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` Get fee information about this pair.

Responses

> 200 OK

`GET` `/api/1/fee_info`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "maker_fee": "string",
    "taker_fee": "string",
    "thirty_day_volume": "string"
}
```

List orders

Returns a list of the most recently placed Orders. Users can specify an optional `state=PENDING` parameter to restrict the results to only open Orders. Users can also specify the market by using the optional currency pair parameter. The list is truncated after 100 items.

Permissions required: `Perm_R_Orders`

QUERY PARAMETERS

`state`	string Enum: `"PENDING"` `"COMPLETE"` Example: `state=PENDING` Filter to only orders of this state
`pair`	string <pair> Example: `pair=XBTZAR` Filter to only orders of this currency pair
`created_before`	integer <int64> Example: `created_before=1530865703508` Filter to orders created before this timestamp (Unix milliseconds)
`limit`	integer <int64> Example: `limit=100` Limit to this many orders

Responses

> 200 OK

`GET` `/api/1/listorders`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "orders": [
       + { … }
    ]
}
```

List trades

Returns a list of the recent Trades for a given currency pair for this user, sorted by oldest first. If `before` is specified, then Trades are returned sorted by most-recent first.

`type` in the response indicates the type of Order that was placed to participate in the trade. Possible types: `BID`, `ASK`.

If `is_buy` in the response is true, then the Order which completed the trade (market taker) was a Bid Order.

Results of this query may lag behind the latest data.

Permissions required: `Perm_R_Orders`

QUERY PARAMETERS

`GET` `/api/1/listtrades`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  - "trades": [
       + { … }
    ]
}
```

`pair` required	string <pair> Example: `pair=XBTZAR` Filter to trades of this currency pair.
`since`	string <timestamp> Example: `since=1470810728478` Filter to trades on or after this timestamp.
`before`	string <timestamp> Example: `before=1470810728478` Filter to trades before this timestamp.
`after_seq`	integer <int64> Example: `after_seq=10` Filter to trades from (including) this sequence number. Default behaviour is not to include this filter.
`before_seq`	integer <int64> Example: `before_seq=1` Filter to trades before (excluding) this sequence number. Default behaviour is not to include this filter.
`sort_desc`	boolean Example: `sort_desc=true` If set to true, sorts trades in descending order, otherwise ascending order will be assumed.
`limit`	integer <int64> `[ 1 .. 100 ]` Example: `limit=100` Limit to this number of trades (default 100).

Responses

> 200 OK

Post market order

POST `/api/1/marketorder`

Create a new Market Order.

A Market Order executes immediately, and either buys as much of the asset that can be bought for a set amount of fiat currency, or sells a set amount of the asset for as much as possible.

Warning! Orders cannot be reversed once they have executed. Please ensure your program has been thoroughly tested before submitting Orders.

If no `base_account_id` or `counter_account_id` are specified, the default base currency or counter currency account will be used. Users can find their account IDs by calling the Balances request.

Permissions required: `Perm_W_Orders`

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` The currency pair to trade.
`type` required	string Enum: `"BUY"` `"SELL"` Example: `type=BUY` `BUY` to buy an asset `SELL` to sell an asset
`counter_volume`	string <amount> Example: `counter_volume=100.50` For a `BUY` order: amount of the counter currency to use (e.g. how much EUR to use to buy BTC in the BTC/EUR market)
`base_volume`	string <amount> Example: `base_volume=1.423` For a `SELL` order: amount of the base currency to use (e.g. how much BTC to sell for EUR in the BTC/EUR market)
`base_account_id`	integer <int64> Example: `base_account_id=12345` The base currency account to use in the trade.
`counter_account_id`	integer <int64> Example: `counter_account_id=12345` The counter currency account to use in the trade.

Responses

> 200 OK

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    “order_id”: “string”
}
```

Get order

Get an Order's details by its ID.

Permissions required: `Perm_R_Orders`

PATH PARAMETERS

`id` required	string Example: `12345` The order ID.

Responses

> 200 OK

GET `/api/1/orders/{id}`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "base": "string",
    "completed_timestamp": "string",
    "counter": "string",
    "creation_timestamp": "string",
    "expiration_timestamp": "string",
    "fee_base": "string",
    "fee_counter": "string",
    "limit_price": "string",
    "limit_volume": "string",
    "order_id": "string",
    "pair": "string",
    "state": "PENDING",
    "type": "BID"
}
```

Post limit order

Create a new Trade Order.

Warning! Orders cannot be reversed once they have executed. Please ensure your program has been thoroughly tested before submitting Orders.

If no `base_account_id` or `counter_account_id` are specified, your default base currency or counter currency account will be used. You can find your Account IDs by calling the Balances API.

Permissions required: `Perm_W_Orders`

QUERY PARAMETERS

`pair` required	string <pair> Example: `pair=XBTZAR` The currency pair to trade.
`type` required	string Enum: `"BID"` `"ASK"` Example: `type=BID` `BID` for a bid (buy) limit order `ASK` for an ask (sell) limit order
`post_only`	boolean Post-only Orders will be cancelled if they would otherwise have traded immediately. For example, if there's a bid at ZAR 100,000 and you place a post-only ask at ZAR 100,000, your order will be cancelled instead of trading. If the best bid is ZAR 100,000 and you place a post-only ask at ZAR 101,000, your order won't trade but will go into the order book.
`volume` required	string <amount> Example: `volume=1.423` Amount of cryptocurrency to buy or sell as a decimal string in units of the currency.
`price` required	string <amount> Example: `price=1200` Limit price as a decimal string in units of ZAR/BTC.
`stop_price`	string <amount> Example: `stop_price=1150` Trigger trade price to activate this order as a decimal string. If this is set then this is treated as a Stop Limit Order and `stop_direction` is expected to be set too.
`stop_direction`	string Enum: `"BELOW"` `"ABOVE"` `"RELATIVE_LAST_TRADE"` Example: `stop_direction=ABOVE` Side of the trigger price to activate the order. This should be set if `stop_price` is also set. `RELATIVE_LAST_TRADE` will automatically infer the direction based on the last trade price and the stop price. If last trade price is less than stop price then stop direction is ABOVE otherwise is BELOW.
`base_account_id`	integer <int64>

POST `/api/1/postorder`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "order_id": "string"
}
```

	Example: `base_account_id=12345` The base currency Account to use in the trade.
`counter_account_id`	integer <int64> Example: `counter_account_id=12345` The counter currency Account to use in the trade.

Responses

> **200** OK

Stop order

Request to stop an Order.

Note!: Once an Order has been completed, it can not be reversed. The return value from this request will indicate if the Stop request was successful or not.

Permissions required: `Perm_W_Orders`

QUERY PARAMETERS

`order_id` `required`	string Example: `order_id=BXMC2CJ7HNB88U4` The Order identifier as a string.

Responses

> **200** OK

POST `/api/1/stoporder`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "success": true
}
```

List orders V2 (BETA)

Returns a list of the most recently placed orders. This endpoint will list up to 100 open orders by default.
This endpoint is in BETA, behaviour and specification may change without any previous notice.

Permissions required: `Perm_R_Orders`

QUERY PARAMETERS

`pair`	string <pair> Example: `pair=XBTZAR` Filter to only orders of this currency pair.
`closed`	boolean Default: `false` Example: `closed=true` If true, will return closed orders instead of open orders.
`created_before`	integer <int64> Example: `created_before=1530865703508` Filter to orders created before this timestamp (Unix milliseconds)
`limit`	integer <int64> `[ 1 .. 1000 ]` Default: `100` Example: `limit=986` Limit to this many orders

Responses

> **200** OK

> **default** Error

GET `/api/exchange/2/listorders`

Response samples

200 default

Content type
application/json

Copy Expand all Collapse all

```
{
  - "orders": [
      + { … }
    ]
}
```

Get order V2 (BETA)

Get the details for an order.
This endpoint is in BETA, behaviour and specification may change without any previous notice.

Permissions required: `Perm_R_Orders`

GET `/api/exchange/2/orders/{id}`

Response samples

200 default

Content type

QUERY PARAMETERS

`id` required	string Order reference

Responses

200 OK

default Error

application/json

Copy Expand all Collapse all

```
{
    "base": "string",
    "completed_timestamp": "string",
    "counter": "string",
    "creation_timestamp": "string",
    "expiration_timestamp": "string",
    "fee_base": "string",
    "fee_counter": "string",
    "limit_price": "string",
    "limit_volume": "string",
    "order_id": "string",
    "pair": "string",
    "side": "BUY",
    "status": "AWAITING",
    "stop_direction": "ABOVE",
    "stop_price": "string",
    "type": "LIMIT"
}
```

Quotes

Quotes allow users to lock in an exchange rate for a short time with the option of either accepting or rejecting the quote.

Quotes are useful for various customer-facing applications where price fluctuations may be confusing.

The API functions as follows:

- Create a quote for the transaction to perform.
- To accept the quote before it expires, exercise the quote.
- To reject the quote before it expires, discard the quote.
- Users can retrieve the status of a quote at any time.

Create quote

Creates a new quote to buy or sell a particular amount of a base currency for a counter currency.

Users can specify either the exact amount to pay or the exact amount to receive.

For example, to buy exactly 0.1 Bitcoin using ZAR, you would create a quote to BUY 0.1 XBTZAR. The returned quote includes the appropriate ZAR amount. To buy Bitcoin using exactly ZAR 100, create a quote to SELL 100 ZARXBT. The returned quote specifies the Bitcoin as the counter amount returned.

An error is returned if the Account is not verified for the currency pair, or if the Account would have insufficient balance to ever exercise the quote.

Permissions required: `Perm_W_Orders`

QUERY PARAMETERS

`type` required	string Example: `type=BUY` `BUY` or `SELL`.
`base_amount` required	string <amount> Example: `base_amount=1.5` Amount to buy or sell in the pair base currency.
`pair` required	string <pair> Example: `pair=XBTZAR` Currency pair to trade. The pair can also be flipped if you want to buy or sell the counter currency (e.g. ZARXBT).
`base_account_id`	integer <int64> Example: `base_account_id=12345` Optional Account for the pair's base currency.
`counter_account_id`	integer <int64> Example: `counter_account_id=12345` Optional Account for the pair's counter currency.

Responses

200 OK

POST `/api/1/quotes`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "base_amount": "string",
    "counter_amount": "string",
    "created_at": "string",
    "discarded": true,
    "exercised": true,
    "expires_at": "string",
    "id": "string",
    "pair": "string",
    "type": "string"
}
```

Get quote

Get the latest status of a quote by its id.

Permissions required: `Perm_R_Orders`

PATH PARAMETERS

`id` `required`	integer <int64> Example: `12345` ID of the quote to retrieve.

Responses

> 200 OK

GET `/api/1/quotes/{id}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "base_amount": "string",
    "counter_amount": "string",
    "created_at": "string",
    "discarded": true,
    "exercised": true,
    "expires_at": "string",
    "id": "string",
    "pair": "string",
    "type": "string"
}
```

Exercise quote

Exercise a quote to perform the Trade. If there is sufficient balance available in the Account, it will be debited and the counter amount credited.

An error is returned if the quote has expired or if the Account has insufficient available balance.

Permissions required: `Perm_W_Orders`

PATH PARAMETERS

`id` `required`	integer <int64> Example: `12345` ID of the quote to exercise.

Responses

> 200 OK

PUT `/api/1/quotes/{id}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "base_amount": "string",
    "counter_amount": "string",
    "created_at": "string",
    "discarded": true,
    "exercised": true,
    "expires_at": "string",
    "id": "string",
    "pair": "string",
    "type": "string"
}
```

Discard quote

Discard a Quote. Once a Quote has been discarded, it cannot be exercised even if it has not expired.

Permissions required: `Perm_W_Orders`

PATH PARAMETERS

`id` `required`	integer <int64> Example: `12345` ID of the quote to discard.

Responses

> 200 OK

DELETE `/api/1/quotes/{id}`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "base_amount": "string",
    "counter_amount": "string",
    "created_at": "string",
    "discarded": true,
    "exercised": true,
    "expires_at": "string",
    "id": "string",
    "pair": "string",
    "type": "string"
}
```

Receive

Receive addresses are used by cryptocurrencies to send assets to a specific "wallet" or user's account. They are a unique address within the blockchain, so assets sent to that address will only be associated with one wallet.

Users may have multiple receive addresses depending on the number of Accounts they have and what currency is associated with that Account.

Get receive address

Returns the default receive address associated with your account and the amount received via the address. Users can specify an optional address parameter to return information for a non-default receive address.

In the response, `total_received` is the total confirmed amount received excluding unconfirmed transactions. `total_unconfirmed` is the total sum of unconfirmed receive transactions.

Permissions required: `Perm_R_Addresses`

QUERY PARAMETERS

`asset` required	string Example: `asset=XBT` Currency code of the asset.
`address`	string Example: `address=1AbbJJzevwFFVBKvZRtQHHFgrJyYTKaMw2` Specific cryptocurrency address to retrieve. If not provided, the default address will be used.

Responses

> 200 OK

GET `/api/1/funding_address`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "account_id": "string",
    "address": "string",
  - "address_meta": [
      + { … }
    ],
    "asset": "string",
    "assigned_at": "string",
    "name": "string",
    "qr_code_uri": "string",
    "receive_fee": "string",
    "total_received": "string",
    "total_unconfirmed": "string"
}
```

Create receive address

Allocates a new receive address to your account. There is a rate limit of 1 address per hour, but bursts of up to 10 addresses are allowed. Only 1 Ethereum receive address can be created.

Permissions required: `Perm_W_Addresses`

QUERY PARAMETERS

`asset` required	string Example: `asset=XBT` Currency code of the asset.
`name`	string Example: `name=My BTC wallet` An optional name for the new Receive Address

Responses

> 200 OK

POST `/api/1/funding_address`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
    "account_id": "string",
    "address": "string",
  - "address_meta": [
      + { … }
    ],
    "asset": "string",
    "assigned_at": "string",
    "name": "string",
    "qr_code_uri": "string",
    "receive_fee": "string",
    "total_received": "string",
    "total_unconfirmed": "string"
}
```

Send

Users are able to send assets from their accounts to the receive address for a cryptocurrency of the same type as their account. For example, a Bitcoin account can send assets to a Bitcoin receive address, etc.

Assets can be sent either to a cryptocurrency receive address, or to the email address of a Luno user. If an email address is specified and the receiver does not have a Luno account, they are sent an invitation to create a Luno account and receive their funds.

Warning! Cryptocurrency transactions are irreversible. Please ensure your program has been thoroughly tested before using this call.

Send

Send assets from an Account. Please note that the asset type sent must match the receive address of the same cryptocurrency of the same type - Bitcoin to Bitcoin, Ethereum to Ethereum, etc.

Sends can be to a cryptocurrency receive address, or the email address of another Luno platform user.

POST `/api/1/send`

Response samples

200

Content type

Permissions required: `Perm_W_Send`

QUERY PARAMETERS

`amount` `required`	string <amount> Example: `amount=1.5` Amount to send as a decimal string.
`currency` `required`	string Example: `currency=XBT` Currency to send.
`address` `required`	string Example: `address=1AbbJJzevwFFVBKvZRtQHHFgrJyYTKaMw2` Destination address or email address. **Note**: • Ethereum addresses must be checksummed. • Ethereum sends to email addresses are not supported.
`description`	string Example: `description=My description` User description for the transaction to record on the account statement.
`message`	string Example: `message=My message` Message to send to the recipient. This is only relevant when sending to an email address.
`external_id`	string Example: `external_id=123e4567-e89b-12d3-a456-426655440000` Optional unique ID to associate with this withdrawal. Useful to prevent duplicate sends in case of failure. This supports all alphanumeric characters, as well as "-" and "_".
`has_destination_tag`	boolean Example: `has_destination_tag=true` Optional boolean flag indicating that a XRP destination tag is provided (even if zero).
`destination_tag`	integer <int64> Example: `destination_tag=12345` Optional XRP destination tag. Note that HasDestinationTag must be true if this value is provided.

Responses

> 200 OK

application/json

Copy Expand all Collapse all

```
{
    "success": true,
    "withdrawal_id": "string"
}
```

Withdrawals

Users are able to withdraw assets from their Accounts to their beneficiaries - banks or other financial institutions that are able to receive the asset type for the Account.

List withdrawal requests

Returns a list of withdrawal requests.

Permissions required: `Perm_R_Withdrawals`

Responses

> 200 OK

GET `/api/1/withdrawals`

Response samples

200

Content type

application/json

Copy Expand all Collapse all

```
{
  - "withdrawals": [
      + { … }
    ]
}
```

Request a withdrawal

Creates a new withdrawal request to the specified beneficiary.

POST `/api/1/withdrawals`

Response samples

Permissions required: `Perm_W_Withdrawals`

QUERY PARAMETERS

`type` required	string Example: `type=ZAR_EFT` Withdrawal type.
`amount` required	string <amount> Example: `amount=10000.00` Amount to withdraw. The currency withdrawn depends on the type setting.
`beneficiary_id`	integer <int64> Example: `beneficiary_id=12345` The beneficiary ID of the bank account the withdrawal will be paid out to. This parameter is required if the user has set up multiple beneficiaries. The beneficiary ID can be found by selecting on the beneficiary name on the user's Beneficiaries page.
`reference`	string For internal use.
`external_id`	string Example: `external_id=123e4567-e89b-12d3-a456-426655440000` Optional unique ID to associate with this withdrawal. Useful to prevent duplicate sends. This field supports all alphanumeric characters including "-" and "_".

Responses

> 200 OK

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "amount": "string",
    "created_at": "string",
    "currency": "string",
    "external_id": "string",
    "fee": "string",
    "id": "string",
    "status": "string",
    "type": "string"
}
```

Get withdrawal request

Returns the status of a particular withdrawal request.

Permissions required: `Perm_R_Withdrawals`

PATH PARAMETERS

`id` required	integer <int64> Example: `12345` Withdrawal ID to retrieve.

Responses

> 200 OK

GET `/api/1/withdrawals/{id}`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "amount": "string",
    "created_at": "string",
    "currency": "string",
    "external_id": "string",
    "fee": "string",
    "id": "string",
    "status": "string",
    "type": "string"
}
```

Cancel withdrawal request

Cancel a withdrawal request. This can only be done if the request is still in state `PENDING`.

Permissions required: `Perm_W_Withdrawals`

PATH PARAMETERS

`id` required	integer <int64> Example: `12345` ID of the withdrawal to cancel.

Responses

> 200 OK

DELETE `/api/1/withdrawals/{id}`

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
    "amount": "string",
    "created_at": "string",
    "currency": "string",
    "external_id": "string",
    "fee": "string",
    "id": "string",
    "status": "string",
    "type": "string"
}
```

Streaming API (beta)

Beta warning: This API is still in beta phase. This means that it may still undergo incompatible changes.

The websocket API provides streaming access to market data. It is more efficient and provides lower latency information than repeatedly polling the orderbook and recent trades, but is more complicated to implement.

The streaming protocol works by requiring the client to keep an in-memory record of the order book. Update messages are then sent from the server and the client uses these to update its copy of the order book. When applied correctly, the client's view of the order book will be identical to the server's view.

Protocol

The client state consists of the following data:

- sequence number
- set of bid orders (id, price, volume)
- set of ask orders (id, price, volume)
- list of trades
- market status

Each update message transmitted from the server has a unique increasing sequence number. The message with sequence number n can be applied to state sequence n-1 to produce state sequence n.

A message may contain multiple updates which must be applied atomically and in order.

If an update is received out-of-sequence (for example update sequence n+2 or n-1 received after update sequence n), the client cannot continue and must reinitialize the state.

There are four types of update:

- Create
- Delete
- Trade
- Status

Create

Add a bid or ask Order to the Order Book with a given id, price and volume. For example:

```
{
  "order_id": "12345678",
  "type": "BID",
  "price": "1234.00",
  "volume": "1.23"
}
```

Delete

Remove the order from the order book with a given id. For example:

```
{
  "order_id": "12345678"
}
```

Trade

Reduce the outstanding volume of an Order in the Order Book (`maker_order_id`) and append a Trade to the Trades List. For example:

```
{
  "base": "0.1",
  "counter": "5232.00",
  "maker_order_id": "12345678",
  "taker_order_id": "87654321"
}
```

Status

Set the status of the market to the given value. For example:

```
{
  "status": "POSTONLY",
}
```

Examples

A new order is placed below market

In this case, an update message is sent with a single create update.

A market order is placed that is immediately filled

In this case, an update message is sent containing multiple trade updates. There will be no create update since the new order never enters the order book.

An order is placed that is partially filled

In this case, the update message contains multiple trade updates and a single create update. The volume in the create update is the remaining volume for the order.

An order is stopped

In this case, the update message contains a single delete update.

The market switches to post-only and trading is suspended

In this case, the update message contains a single status update.

Websockets

The streaming updates protocol described above can be accessed using websockets. The server sends the current order book state, and then sends update messages as quickly as possible. Both the client and server must send regular keep alive messages to avoid disconnection during periods of low update message activity.

Connect to the websocket server at: `wss://ws.luno.com/api/1/stream/:pair`

Beta warning: This API is still in beta phase. This means that it may still undergo incompatible changes.

The client must start by sending API key credentials:

```
{
  "api_key_id": "abcdef",
  "api_key_secret": "api_key_secret_goes_here"
}
```

The server will then send the current order book in the following format:

```
{
  "sequence": "24352",
  "asks": [
    {
      "id": "23298343",
      "price": "1234.00",
      "volume": "0.93"
    }
  ],
  "bids": [
    {
      "id": "3498282",
      "price": "1201.00",
      "volume": "1.22"
    }
  ],
  "status": "ACTIVE",
  "timestamp": 1528884331021
}
```

The server then sends messages like the following:

```
{
  "sequence": "24353",
  "trade_updates": null, // array of 0 or more trade updates
  "create_update": null, // null or 1 create update
  "delete_update": null, // null or 1 delete update
  "status_update": null, // null or 1 status update
  "timestamp": 1469031991
}
```

An empty message is a *keep alive message*.

If there is any error while processing an update (e.g. an out-of-order update) or there is a network error or timeout (e.g. keep alive message not received in time), the client should close the connection and reconnect in order to reinitialize its state. It is critical that clients implement exponential backoff for all reconnections to avoid overloading the server in case of errors.

Changelog

- **2020-08-20:** Removed Lightning API section.
- **2020-06-22:** Updated domain to api.luno.com
- **2020-02-24:** Added optional `destination_tag` and `has_destination_tag` parameter to `POST /api/1/send` to support XRP sends. Please note, not specifying `has_destination_tag` parameters to `POST /api/1/send` to support XRP sends with destination tags. Please see https://xrpl.org/source-and-destination-tags.html to learn more about XRP destination tags.
- **2020-02-12:** Added `PUT /api/1/accounts/:id/name` to allow updating of an account name.
- **2019-11-21:** Added `GET /api/1/beneficiaries` to allow listing of bank beneficiaries.
- **2018-07-16:** Added aggregated order book API. Rate limits for market data have been increased to 1 per second. Market data may be cached for up to 1 second.
- **2018-06-29:** Add `post_only` parameter to `POST /api/1/postorder`.
- **2018-06-15:** Update PHP and Python SDK URLs.
- **2018-06-13:** Added `timestamp` to the orderbook streamer response.
- **2018-06-08:** Update Go SDK URL.
- **2017-12-23:** Added `maker_order_id` and `taker_order_id` to streaming trade updates. Deprecated `order_id`.
- **2017-10-31:** OAuth2 is no longer available for new applications.
- **2017-10-16:** Updated `/api/1/trades` to only return `BID` or `ASK` types and it may now lag behind latest data.
- **2017-07-02:** Updated websocket server to wss://ws.luno.com.
- **2017-03-02:** Added `/api/1/fee_info` which returns your fees and 30 day trading volume.
- **2016-11-21:** The `/api/1/trades` now returns at most 100 results per call.
- **2016-11-01:** Removed 50 receive address create limit on `POST /api/1/funding_address` to allow unlimited receive addresses per account. Address creation is rate limited to 1 per hour, allowing for bursts of up to 10 consecutive calls.
- **2016-08-10:** Added `GET /api/1/listtrades` to allow listing of recent trades. Please note that trades will soon be removed from the response of `GET /api/1/listorders` `GET /api/1/orders/:id` .

- **2016-08-05:** Added beta Streaming API section
- **2016-07-25:** Added optional `beneficiary_id` parameter to `POST /api/1/withdrawals`.
- **2016-05-29:** Error code 429 may be returned when exceeding rate limits. This will become the default as of 2016-07-01.
- **2016-04-04:** Added `completed_timestamp` field to `GET /api/1/listorders` and `GET /api/1/orders/:id` responses.
- **2016-02-05:** Added optional `since` parameter to `GET /api/1/trades` and added `is_buy` field to the response.
- **2015-09-14:** Added `POST /api/1/marketorder` to allow placing of market orders.
- **2015-07-29:** Added `Perm_R_Beneficiaries` and `Perm_W_Beneficiaries` permissions. You will have to generate a new API key if you require these permissions.
- **2015-06-08:** Renamed `GET /api/1/withdrawals/` to `GET /api/1/withdrawals` and `POST /api/1/withdrawals/` to `POST /api/1/withdrawals` to be more consistent with other endpoints. The old URLs are now deprecated.
- **2015-05-28:** Added `POST accounts` for creating additional accounts in specified currencies.
- **2015-05-07:** Added the "Name" field to the "Balance" response
- **2015-04-25:**
  - Added the "Accounts" section.
  - Added the account transactions and pending transactions calls.
  - Added the "Permissions" section.
  - Documented which permissions are required for each call.
  - Updated description of the "send" call. A pin is no longer required.
  - Added "name" parameter to `POST /api/1/funding_address.`
- **2015-03-27:** Return a list of trades for an order on `GET orders/:id` if the order has any trades.
- **2015-01-30:** Clarified the interpretation of `base`, `counter`, `base_fee` and `counter_fee` in the `list_orders` response in the case where `counter_fee` is nonzero for buy orders and where `base_fee` is nonzero for sell orders.
- **2014-12-17:** The amount parameter for withdrawal requests now excludes the withdrawal fee.
- **2014-12-12:**
  - Added the new Quotes API.
  - The `transactions` beta call has been deprecated.
- **2014-12-04:** The `balance` method can now be called with no arguments to return all account balances.
- **2014-08-26:**
  - Added Send API call.
  - Added OAuth2 API.
- **2014-06-10:** Orders placed through the API are no longer subject to different limits than those placed through the website.
- **2014-06-02:**
  - You can now create multiple API keys with different permissions (e.g. read-only, read/write).
  - Added calls to list, create, get and cancel withdrawal requests.
  - Added link to Android client library.
- **2014-05-29:**
  - The preferred host name for API calls has changed to `api.mybitx.com`.
  - Added experimental call to retrieve transactions list.
  - Added call to allocate new receive addresses.
  - Receive address call now returns the amount received by that address.
  - You can now request `listorders` to return only the list of open orders.
- **2014-04-15:** Previously orders created through the API would expire after 24 hours. Now, orders created through the API do not expire. The behaviour is now the same as for orders placed through the website.
- **2014-01-25:**
  - A new `funding_address` call has been added to get the bitcoin address you need to fund your trade account balance.
- **2014-01-21:**
  - The API has been extended to support multiple asset pairs.
  - A new `balance` call has been added to query the trading account balance.
  - All URLs have been renamed from `/api/1/BTCZAR/x` to `/api/1/x?pair=XBTZAR`. The old URLs are now deprecated.
  - getlimits: This call has been deprecated. Please use the new `balance` call instead.
  - ticker: The `currency` field is now deprecated.
  - orderbook: The `currency` field is now deprecated.
  - trades: The `currency` field is now deprecated.
  - listorders: The `btc`, `zar`, `fee_btc` and `fee_zar` fields are now deprecated. Please use `base`, `counter`, `fee_base`, `fee_counter` fields instead.
  - The embedded market indicator has been removed since nobody is using it.
  - All deprecated features will continue to work for two months.
- **2014-01-06:**
  - listorders: Added fee_btc and fee_zar fields.
  - listorders: Removed SETTLEMENT state (it's no longer relevant).
  - ticker: Removed mtgox_price (use Mt Gox's API directly instead).