

---

# **cbpro2 Documentation**

***Release 1.0.4***

**Daniel Paquin, yiwen song**

**May 26, 2020**



---

## Contents

---

<b>1</b>	<b>Documentation</b>	<b>3</b>
1.1	Public Client . . . . .	3
1.2	Authenticated Client . . . . .	6
1.3	Authentication . . . . .	18
1.4	Order Book . . . . .	19
1.5	Websocket Client . . . . .	19
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



cbpro2 is a python client library to interact with coinbase pro.



## 1.1 Public Client

### Contents

- *Public Client*
  - *Public Client*

### 1.1.1 Public Client

**class** `cbpro.public_client.PublicClient` (*api\_url*='https://api.pro.coinbase.com', *time-out*=30)

cbpro public client API.

All requests default to the *product\_id* specified at object creation if not otherwise specified.

**url**

API URL. Defaults to cbpro API.

**Type** Optional[str]

**get\_currencies** ()

List known currencies.

**Returns**

**List of currencies. Example::**

```
[{ "id": "BTC", "name": "Bitcoin", "min_size": "0.00000001"
}, { "id": "USD", "name": "United States Dollar", "min_size": "0.01000000"
}]
```

**Return type** list

**get\_product\_24hr\_stats** (*product\_id*)

Get 24 hr stats for the product.

**Parameters** *product\_id* (*str*) – Product

**Returns**

**24 hour stats. Volume is in base currency units.**

**Open, high, low are in quote currency units. Example::**

```
{ "open": "34.19000000", "high": "95.70000000", "low": "7.06000000", "volume":  
  "2.41000000"  
}
```

**Return type** dict

**get\_product\_historic\_rates** (*product\_id*, *start=None*, *end=None*, *granularity=None*)

Historic rates for a product.

Rates are returned in grouped buckets based on requested *granularity*. If start, end, and granularity aren't provided, the exchange will assume some (currently unknown) default values.

Historical rate data may be incomplete. No data is published for intervals where there are no ticks.

**Caution:** Historical rates should not be polled frequently. If you need real-time information, use the trade and book endpoints along with the websocket feed.

The maximum number of data points for a single request is 200 candles. If your selection of start/end time and granularity will result in more than 200 data points, your request will be rejected. If you wish to retrieve fine granularity data over a larger time range, you will need to make multiple requests with new start/end ranges.

**Parameters**

- **product\_id** (*str*) – Product
- **start** (*Optional[str]*) – Start time in ISO 8601
- **end** (*Optional[str]*) – End time in ISO 8601
- **granularity** (*Optional[int]*) – Desired time slice in seconds

**Returns**

**Historic candle data. Example:**

```
[ [ time, low, high, open, close, volume ], [ 1415398768, 0.32, 4.2, 0.35, 4.2, 12.3 ],  
  ...  
]
```

**Return type** list

**get\_product\_order\_book** (*product\_id*, *level=1*)

Get a list of open orders for a product.

The amount of detail shown can be customized with the *level* parameter: \* 1: Only the best bid and ask \* 2: Top 50 bids and asks (aggregated) \* 3: Full order book (non aggregated)

Level 1 and Level 2 are recommended for polling. For the most up-to-date data, consider using the websocket stream.



**Caution:** Level 3 is only recommended for users wishing to maintain a full real-time order book using the websocket stream. Abuse of Level 3 via polling will cause your access to be limited or blocked.

#### Parameters

- **product\_id** (*str*) – Product
- **level** (*Optional[int]*) – Order book level (1, 2, or 3). Default is 1.

#### Returns

##### Order book. Example for level 1::

```
{ "sequence": "3", "bids": [
    [ price, size, num-orders ],
  ], "asks": [
    [ price, size, num-orders ],
  ]
}
```

**Return type** dict

**get\_product\_ticker** (*product\_id*)

Snapshot about the last trade (tick), best bid/ask and 24h volume.

**Caution:** Polling is discouraged in favor of connecting via the websocket stream and listening for match messages.

**Parameters** **product\_id** (*str*) – Product

#### Returns

##### Ticker info. Example::

```
{ "trade_id": 4729088, "price": "333.99", "size": "0.193", "bid": "333.98", "ask":
  "333.99", "volume": "5957.11914015", "time": "2015-11-14T20:46:03.511254Z"
}
```

**Return type** dict

**get\_product\_trades** (*product\_id*, *before=*", *after=*", *limit=None*, *result=None*)

List the latest trades for a product.

This method returns a generator which may make multiple HTTP requests while iterating through it.

#### Parameters

- **product\_id** (*str*) – Product
- **before** (*Optional[str]*) – start time in ISO 8601
- **after** (*Optional[str]*) – end time in ISO 8601
- **limit** (*Optional[int]*) – the desired number of trades (can be more than 100, automatically paginated)
- **results** (*Optional[list]*) – list of results that is used for the pagination

#### Returns

##### Latest trades. Example::

```
[{ "time": "2014-11-07T22:19:28.578544Z", "trade_id": 74, "price":  
  "10.00000000", "size": "0.01000000", "side": "buy"  
}, { "time": "2014-11-07T01:08:43.642366Z", "trade_id": 73, "price":  
  "100.00000000", "size": "0.01000000", "side": "sell"
```

**Return type**

list

}}

**get\_products()**

Get a list of available currency pairs for trading.

**Returns****Info about all currency pairs. Example::**

```
[  
  { "id": "BTC-USD", "display_name": "BTC/USD", "base_currency": "BTC",  
    "quote_currency": "USD", "base_min_size": "0.01", "base_max_size":  
    "10000.00", "quote_increment": "0.01"  
  }  
]
```

**Return type** list**get\_time()**

Get the API server time.

**Returns****Server time in ISO and epoch format (decimal seconds****since Unix epoch). Example::**

```
{ "iso": "2015-01-07T23:47:25.201Z", "epoch": 1420674445.201  
}
```

**Return type** dict

## 1.2 Authenticated Client

**Contents**

- *Authenticated Client*
  - *Authenticated Client*

### 1.2.1 Authenticated Client

cbpro/AuthenticatedClient.py Daniel Paquin

For authenticated requests to the Coinbase exchange

```
class cbpro.authenticated_client.AuthenticatedClient (key, b64secret, passphrase,  
                                                    api_url='https://api.pro.coinbase.com')
```

Provides access to Private Endpoints on the cbpro API.

All requests default to the live *api\_url*: `'https://api.pro.coinbase.com'`. To test your application using the sandbox modify the *api\_url*.

**url**

The api url for this client instance to use.

**Type** str

**auth**

Custom authentication handler for each request.

**Type** *CBProAuth*

**session**

Persistent HTTP connection object.

**Type** requests.Session

**buy** (*product\_id, order\_type, \*\*kwargs*)

Place a buy order.

This is included to maintain backwards compatibility with older versions of cbpro-Python. For maximum support from docstrings and function signatures see the order type-specific functions `place_limit_order`, `place_market_order`, and `place_stop_order`.

**Parameters**

- **product\_id** (*str*) – Product to order (eg. 'BTC-USD')
- **order\_type** (*str*) – Order type ('limit', 'market', or 'stop')
- **\*\*kwargs** – Additional arguments can be specified for different order types.

**Returns** Order details. See `place_order` for example.

**Return type** dict

**cancel\_all** (*product\_id=None*)

With best effort, cancel all open orders.

**Parameters** **product\_id** (*Optional[str]*) – Only cancel orders for this product\_id

**Returns**

**A list of ids of the canceled orders. Example::**

```
[ "144c6f8e-713f-4682-8435-5280f8e8b2b4",      "debe4907-95dc-442f-af3b-  
  cec12f42ebda", "cf7aceee-7b08-4227-a76c-3858144323ab", "dfc5ae27-cadb-  
  4c0c-beef-8994936fde8a", "34fecfbf-de33-4273-b2c6-baf8e8948be4"  
]
```

**Return type** list

**cancel\_order** (*order\_id*)

Cancel a previously placed order.

If the order had no matches during its lifetime its record may be purged. This means the order details will not be available with `get_order(order_id)`. If the order could not be canceled (already filled or previously canceled, etc), then an error response will indicate the reason in the message field.

**Caution:** The order id is the server-assigned order id and not the optional `client_oid`.

**Parameters** `order_id` (*str*) – The `order_id` of the order you want to cancel

**Returns**

Containing the `order_id` of cancelled order. **Example::** [ “c5ab5eae-76be-480e-8961-00792dc7e138” ]

**Return type** list

**close\_position** (*repay\_only*)

Close position.

**Parameters** `repay_only` (*bool*) – Undocumented by cbpro.

**Returns** Undocumented

**coinbase\_deposit** (*amount, currency, coinbase\_account\_id*)

Deposit funds from a coinbase account.

You can move funds between your Coinbase accounts and your cbpro trading accounts within your daily limits. Moving funds between Coinbase and cbpro is instant and free.

See `AuthenticatedClient.get_coinbase_accounts()` to receive information regarding your coinbase\_accounts.

**Parameters**

- **amount** (*Decimal*) – The amount to deposit.
- **currency** (*str*) – The type of currency.
- **coinbase\_account\_id** (*str*) – ID of the coinbase account.

**Returns**

Information about the deposit. **Example::**

```
{ "id": "593533d2-ff31-46e0-b22e-ca754147a96a", "amount": "10.00", "currency":  
  "BTC",  
}
```

**Return type** dict

**coinbase\_withdraw** (*amount, currency, coinbase\_account\_id*)

Withdraw funds to a coinbase account.

You can move funds between your Coinbase accounts and your cbpro trading accounts within your daily limits. Moving funds between Coinbase and cbpro is instant and free.

See `AuthenticatedClient.get_coinbase_accounts()` to receive information regarding your coinbase\_accounts.

**Parameters**

- **amount** (*Decimal*) – The amount to withdraw.
- **currency** (*str*) – The type of currency (eg. ‘BTC’)
- **coinbase\_account\_id** (*str*) – ID of the coinbase account.

**Returns**

Information about the deposit. **Example::**

```
{ "id": "593533d2-ff31-46e0-b22e-ca754147a96a", "amount": "10.00", "currency":  
  "BTC",
```

```
}
```

**Return type** dict

**create\_report** (*report\_type*, *start\_date*, *end\_date*, *product\_id=None*, *account\_id=None*, *report\_format='pdf'*, *email=None*)

Create report of historic information about your account.

The report will be generated when resources are available. Report status can be queried via *get\_report(report\_id)*.

**Parameters**

- **report\_type** (*str*) – ‘fills’ or ‘account’
- **start\_date** (*str*) – Starting date for the report in ISO 8601
- **end\_date** (*str*) – Ending date for the report in ISO 8601
- **product\_id** (*Optional[str]*) – ID of the product to generate a fills report for. Required if *account\_type* is ‘fills’
- **account\_id** (*Optional[str]*) – ID of the account to generate an account report for. Required if *report\_type* is ‘account’.
- **report\_format** (*Optional[str]*) – ‘pdf’ or ‘csv’. Default is ‘pdf’.
- **email** (*Optional[str]*) – Email address to send the report to.

**Returns**

**Report details. Example::**

```
{ "id": "0428b97b-bec1-429e-a94c-59232926778d", "type": "fills", "status": "pending",
  "created_at": "2015-01-06T10:34:47.000Z", "completed_at": undefined,
  "expires_at": "2015-01-13T10:35:47.000Z", "file_url": undefined, "params": {
    "start_date": "2014-11-01T00:00:00.000Z", "end_date": "2014-11-30T23:59:59.000Z"
  }
}
```

**Return type** dict

**crypto\_withdraw** (*amount*, *currency*, *crypto\_address*)

Withdraw funds to a crypto address.

**Parameters**

- **amount** (*Decimal*) – The amount to withdraw
- **currency** (*str*) – The type of currency (eg. ‘BTC’)
- **crypto\_address** (*str*) – Crypto address to withdraw to.

**Returns**

**Withdraw details. Example::**

```
{ "id": "593533d2-ff31-46e0-b22e-ca754147a96a", "amount": "10.00", "currency": "BTC",
}
```

**Return type** dict

**deposit** (*amount*, *currency*, *payment\_method\_id*)

Deposit funds from a payment method.

See `AuthenticatedClient.get_payment_methods()` to receive information regarding payment methods.

**Parameters**

- **amount** (*Decimal*) – The amount to deposit.
- **currency** (*str*) – The type of currency.
- **payment\_method\_id** (*str*) – ID of the payment method.

**Returns**

**Information about the deposit. Example::**

```
{ "id": "593533d2-ff31-46e0-b22e-ca754147a96a", "amount": "10.00", "currency":
  "USD", "payout_at": "2016-08-20T00:31:09Z"
}
```

**Return type** dict

**get\_account** (*account\_id*)

Get information for a single account.

Use this endpoint when you know the `account_id`.

**Parameters** **account\_id** (*str*) – Account id for account you want to get.

**Returns**

**Account information. Example::**

```
{ "id": "a1b2c3d4", "balance": "1.100", "holds": "0.100", "available": "1.00", "cur-
  rency": "USD"
}
```

**Return type** dict

**get\_account\_history** (*account\_id*, *\*\*kwargs*)

List account activity. Account activity either increases or decreases your account balance.

Entry type indicates the reason for the account change. \* transfer: Funds moved to/from Coinbase to cbpro \* match: Funds moved as a result of a trade \* fee: Fee as a result of a trade \* rebate: Fee rebate as per our fee schedule

If an entry is the result of a trade (match, fee), the details field will contain additional information about the trade.

**Parameters**

- **account\_id** (*str*) – Account id to get history of.
- **kwargs** (*dict*) – Additional HTTP request parameters.

**Returns**

**History information for the account. Example::**

```
[
  { "id": "100", "created_at": "2014-11-07T08:19:27.028459Z", "amount":
    "0.001", "balance": "239.669", "type": "fee", "details": {
```

```

        "order_id": "d50ec984-77a8-460a-b958-66f114b0de9b", "trade_id": "74",
        "product_id": "BTC-USD"
    }
}, {
    ...
}
]

```

**Return type** list

**get\_account\_holds** (*account\_id*, *\*\*kwargs*)

Get holds on an account.

This method returns a generator which may make multiple HTTP requests while iterating through it.

Holds are placed on an account for active orders or pending withdraw requests.

As an order is filled, the hold amount is updated. If an order is canceled, any remaining hold is removed. For a withdraw, once it is completed, the hold is removed.

The *type* field will indicate why the hold exists. The hold type is 'order' for holds related to open orders and 'transfer' for holds related to a withdraw.

The *ref* field contains the id of the order or transfer which created the hold.

#### Parameters

- **account\_id** (*str*) – Account id to get holds of.
- **kwargs** (*dict*) – Additional HTTP request parameters.

#### Returns

**Hold information for the account. Example::**

```

[
    {
        "id": "82dcd140-c3c7-4507-8de4-2c529cd1a28f", "account_id":
        "e0b3f39a-183d-453e-b754-0c13e5bab0b3", "created_at": "2014-11-
        06T10:34:47.123456Z", "updated_at": "2014-11-06T10:40:47.123456Z",
        "amount": "4.23", "type": "order", "ref": "0a205de4-dd35-4370-a285-
        fe8fc375a273",
    }, { ... }
]

```

**Return type** generator(list)

**get\_accounts** ()

Get a list of trading all accounts.

When you place an order, the funds for the order are placed on hold. They cannot be used for other orders or withdrawn. Funds will remain on hold until the order is filled or canceled. The funds on hold for each account will be specified.

#### Returns

**Info about all accounts. Example::**

```

[

```

```
{ "id": "71452118-efc7-4cc4-8780-a5e22d4baa53", "currency": "BTC",  
  "balance": "0.0000000000000000", "available": "0.0000000000000000",  
  "hold": "0.0000000000000000", "profile_id": "75da88c5-05bf-4f54-bc85-  
5c775bd68254"  
}, {  
  ...  
}  
]
```

**Return type** list

- Additional info included in response for margin accounts.

**get\_coinbase\_accounts()**

Get a list of your coinbase accounts.

**Returns** Coinbase account details.

**Return type** list

**get\_fills**(*product\_id=None, order\_id=None, \*\*kwargs*)

Get a list of recent fills.

As of 8/23/18 - Requests without either *order\_id* or *product\_id* will be rejected

This method returns a generator which may make multiple HTTP requests while iterating through it.

Fees are recorded in two stages. Immediately after the matching engine completes a match, the fill is inserted into our datastore. Once the fill is recorded, a settlement process will settle the fill and credit both trading counterparties.

The 'fee' field indicates the fees charged for this fill.

The 'liquidity' field indicates if the fill was the result of a liquidity provider or liquidity taker. M indicates Maker and T indicates Taker.

**Parameters**

- **product\_id** (*str*) – Limit list to this *product\_id*
- **order\_id** (*str*) – Limit list to this *order\_id*
- **kwargs** (*dict*) – Additional HTTP request parameters.

**Returns**

**Containing information on fills. Example::**

```
[  
  { "trade_id": 74, "product_id": "BTC-USD", "price": "10.00", "size":  
    "0.01", "order_id": "d50ec984-77a8-460a-b958-66f114b0de9b", "cre-  
    ated_at": "2014-11-07T22:19:28.578544Z", "liquidity": "T", "fee":  
    "0.00025", "settled": true, "side": "buy"  
  }, {  
    ...  
  }  
]
```



**Return type** list

**get\_fundings** (*status=None*, *\*\*kwargs*)

Every order placed with a margin profile that draws funding will create a funding record.

This method returns a generator which may make multiple HTTP requests while iterating through it.

**Parameters**

- **status** (*list/str*) – Limit funding records to these statuses. **\*\* Options:** ‘out-standing’, ‘settled’, ‘rejected’
- **kwargs** (*dict*) – Additional HTTP request parameters.

**Returns**

Containing information on margin funding. Example::

```
[
    { "id": "b93d26cd-7193-4c8d-bfcc-446b2fe18f71", "order_id": "b93d26cd-
      7193-4c8d-bfcc-446b2fe18f71", "profile_id": "d881e5a6-58eb-47cd-b8e2-
      8d9f2e3ec6f6", "amount": "1057.6519956381537500", "status": "settled",
      "created_at": "2017-03-17T23:46:16.663397Z", "currency": "USD", "re-
      paid_amount": "1057.6519956381537500", "default_amount": "0", "re-
      paid_default": false
    }, {
      ...
    }
]
```

**Return type** list

**get\_order** (*order\_id*)

Get a single order by order id.

If the order is canceled the response may have status code 404 if the order had no matches.

**Caution:** Open orders may change state between the request and the response depending on market conditions.

**Parameters** **order\_id** (*str*) – The order to get information of.

**Returns**

Containing information on order. Example::

```
{ "created_at": "2017-06-18T00:27:42.920136Z", "executed_value":
  "0.0000000000000000", "fill_fees": "0.0000000000000000", "filled_size":
  "0.00000000", "id": "9456f388-67a9-4316-bad1-330c5353804f",
  "post_only": true, "price": "1.00000000", "product_id": "BTC-USD",
  "settled": false, "side": "buy", "size": "1.00000000", "status": "pending",
  "stp": "dc", "time_in_force": "GTC", "type": "limit"
}
```

**Return type** dict

**get\_orders** (*product\_id=None*, *status=None*, *\*\*kwargs*)

List your current open orders.

This method returns a generator which may make multiple HTTP requests while iterating through it.

Only open or un-settled orders are returned. As soon as an order is no longer open and settled, it will no longer appear in the default request.

Orders which are no longer resting on the order book, will be marked with the 'done' status. There is a small window between an order being 'done' and 'settled'. An order is 'settled' when all of the fills have settled and the remaining holds (if any) have been removed.

For high-volume trading it is strongly recommended that you maintain your own list of open orders and use one of the streaming market data feeds to keep it updated. You should poll the open orders endpoint once when you start trading to obtain the current state of any open orders.

#### Parameters

- **product\_id** (*Optional[str]*) – Only list orders for this product
- **status** (*Optional[list/str]*) – Limit list of orders to this status or statuses. Passing 'all' returns orders of all statuses. \*\* Options: 'open', 'pending', 'active', 'done', 'settled'  
\*\* default: ['open', 'pending', 'active']

#### Returns

Containing information on orders. Example::

```
[
  {
    "id": "d0c5340b-6d6c-49d9-b567-48c4bfca13d2",
    "price": "0.10000000",
    "size": "0.01000000",
    "product_id": "BTC-USD",
    "side": "buy",
    "stp": "dc",
    "type": "limit",
    "time_in_force": "GTC",
    "post_only": false,
    "created_at": "2016-12-08T20:02:28.53864Z",
    "fill_fees": "0.000000000000000000",
    "filled_size": "0.00000000",
    "executed_value": "0.000000000000000000",
    "status": "open",
    "settled": false
  }, {
    ...
  }
]
```

**Return type** list

**get\_payment\_methods()**

Get a list of your payment methods.

**Returns** Payment method details.

**Return type** list

**get\_position()**

Get An overview of your margin profile.

**Returns** Details about funding, accounts, and margin call.

**Return type** dict

**get\_report(report\_id)**

Get report status.

Use to query a specific report once it has been requested.

**Parameters** **report\_id** (*str*) – Report ID

**Returns** Report details, including file url once it is created.

**Return type** dict

**get\_trailing\_volume()**

Get your 30-day trailing volume for all products.

This is a cached value that's calculated every day at midnight UTC.

**Returns**

**30-day trailing volumes. Example::**

```
[
    { "product_id": "BTC-USD", "exchange_volume": "11800.00000000", "volume": "100.00000000", "recorded_at": "1973-11-29T00:05:01.123456Z" }, {
    ...
}]
```

**Return type** list

**margin\_transfer** (*margin\_profile\_id*, *transfer\_type*, *currency*, *amount*)

Transfer funds between your standard profile and a margin profile.

**Parameters**

- **margin\_profile\_id** (*str*) – Margin profile ID to withdraw or deposit from.
- **transfer\_type** (*str*) – ‘deposit’ or ‘withdraw’
- **currency** (*str*) – Currency to transfer (eg. ‘USD’)
- **amount** (*Decimal*) – Amount to transfer

**Returns**

**Transfer details. Example::**

```
{ "created_at": "2017-01-25T19:06:23.415126Z", "id": "80bc6b74-8b1f-4c60-a089-c61f9810d4ab", "user_id": "521c20b3d4ab09621f000011", "profile_id": "cda95996-ac59-45a3-a42e-30daeb061867", "margin_profile_id": "45fa9e3b-00ba-4631-b907-8a98cbdf21be", "type": "deposit", "amount": "2", "currency": "USD", "account_id": "23035fc7-0707-4b59-b0d2-95d0c035f8f5", "margin_account_id": "e1d9862c-a259-4e83-96cd-376352a9d24d", "margin_product_id": "BTC-USD", "status": "completed", "nonce": 25 }
```

**Return type** dict

**place\_limit\_order** (*product\_id*, *side*, *price*, *size*, *client\_oid*=None, *stp*=None, *time\_in\_force*=None, *cancel\_after*=None, *post\_only*=None, *overdraft\_enabled*=None, *funding\_amount*=None)

Place a limit order.

**Parameters**

- **product\_id** (*str*) – Product to order (eg. ‘BTC-USD’)
- **side** (*str*) – Order side (‘buy’ or ‘sell’)

- **price** (*Decimal*) – Price per cryptocurrency
- **size** (*Decimal*) – Amount of cryptocurrency to buy or sell
- **client\_oid** (*Optional[str]*) – User-specified Order ID
- **stp** (*Optional[str]*) – Self-trade prevention flag. See *place\_order* for details.
- **time\_in\_force** (*Optional[str]*) – Time in force. Options: ‘GTC’ Good till canceled ‘GTT’ Good till time (set by *cancel\_after*) ‘IOC’ Immediate or cancel ‘FOK’ Fill or kill
- **cancel\_after** (*Optional[str]*) – Cancel after this period for ‘GTT’ orders. Options are ‘min’, ‘hour’, or ‘day’.
- **post\_only** (*Optional[bool]*) – Indicates that the order should only make liquidity. If any part of the order results in taking liquidity, the order will be rejected and no part of it will execute.
- **overdraft\_enabled** (*Optional[bool]*) – If true funding above and beyond the account balance will be provided by margin, as necessary.
- **funding\_amount** (*Optional[Decimal]*) – Amount of margin funding to be provided for the order. Mutually exclusive with *overdraft\_enabled*.

**Returns** Order details. See *place\_order* for example.

**Return type** dict

**place\_market\_order** (*product\_id, side, size=None, funds=None, client\_oid=None, stp=None, overdraft\_enabled=None, funding\_amount=None*)

Place market order.

**Parameters**

- **product\_id** (*str*) – Product to order (eg. ‘BTC-USD’)
- **side** (*str*) – Order side (‘buy’ or ‘sell’)
- **size** (*Optional[Decimal]*) – Desired amount in crypto. Specify this or *funds*.
- **funds** (*Optional[Decimal]*) – Desired amount of quote currency to use. Specify this or *size*.
- **client\_oid** (*Optional[str]*) – User-specified Order ID
- **stp** (*Optional[str]*) – Self-trade prevention flag. See *place\_order* for details.
- **overdraft\_enabled** (*Optional[bool]*) – If true funding above and beyond the account balance will be provided by margin, as necessary.
- **funding\_amount** (*Optional[Decimal]*) – Amount of margin funding to be provided for the order. Mutually exclusive with *overdraft\_enabled*.

**Returns** Order details. See *place\_order* for example.

**Return type** dict

**place\_order** (*product\_id, side, order\_type, \*\*kwargs*)

Place an order.

The three order types (limit, market, and stop) can be placed using this method. Specific methods are provided for each order type, but if a more generic interface is desired this method is available.

**Parameters**

- **product\_id** (*str*) – Product to order (eg. ‘BTC-USD’)
- **side** (*str*) – Order side (‘buy’ or ‘sell’)
- **order\_type** (*str*) – Order type (‘limit’, ‘market’, or ‘stop’)
- **\*\*client\_oid** (*str*) – Order ID selected by you to identify your order. This should be a UUID, which will be broadcast in the public feed for *received* messages.
- **\*\*stp** (*str*) – Self-trade prevention flag. cbpro doesn’t allow self- trading. This behavior can be modified with this flag. Options: ‘dc’ Decrease and Cancel (default) ‘co’ Cancel oldest ‘cn’ Cancel newest ‘cb’ Cancel both
- **\*\*overdraft\_enabled** (*Optional[bool]*) – If true funding above and beyond the account balance will be provided by margin, as necessary.
- **\*\*funding\_amount** (*Optional[Decimal]*) – Amount of margin funding to be provided for the order. Mutually exclusive with *overdraft\_enabled*.
- **\*\*kwargs** – Additional arguments can be specified for different order types. See the limit/market/stop order methods for details.

## Returns

### Order details. Example::

```
{ "id": "d0c5340b-6d6c-49d9-b567-48c4bfca13d2", "price": "0.10000000",
  "size": "0.01000000", "product_id": "BTC-USD", "side": "buy", "stp": "dc",
  "type": "limit", "time_in_force": "GTC", "post_only": false, "created_at":
  "2016-12-08T20:02:28.53864Z", "fill_fees": "0.000000000000000000",
  "filled_size": "0.00000000", "executed_value": "0.000000000000000000",
  "status": "pending", "settled": false
}
```

### Return type dict

**place\_stop\_order** (*product\_id, side, price, size=None, funds=None, client\_oid=None, stp=None, overdraft\_enabled=None, funding\_amount=None*)

Place stop order.

### Parameters

- **product\_id** (*str*) – Product to order (eg. ‘BTC-USD’)
- **side** (*str*) – Order side (‘buy’ or ‘sell’)
- **price** (*Decimal*) – Desired price at which the stop order triggers.
- **size** (*Optional[Decimal]*) – Desired amount in crypto. Specify this or *funds*.
- **funds** (*Optional[Decimal]*) – Desired amount of quote currency to use. Specify this or *size*.
- **client\_oid** (*Optional[str]*) – User-specified Order ID
- **stp** (*Optional[str]*) – Self-trade prevention flag. See *place\_order* for details.
- **overdraft\_enabled** (*Optional[bool]*) – If true funding above and beyond the account balance will be provided by margin, as necessary.
- **funding\_amount** (*Optional[Decimal]*) – Amount of margin funding to be provided for the order. Mutually exclusive with *overdraft\_enabled*.

**Returns** Order details. See *place\_order* for example.

**Return type** dict

**repay\_funding** (*amount*, *currency*)

Repay funding. Repays the older funding records first.

**Parameters**

- **amount** (*int*) – Amount of currency to repay
- **currency** (*str*) – The currency, example USD

**Returns** Not specified by cbpro.

**sell** (*product\_id*, *order\_type*, *\*\*kwargs*)

Place a sell order.

This is included to maintain backwards compatibility with older versions of cbpro-Python. For maximum support from docstrings and function signatures see the order type-specific functions *place\_limit\_order*, *place\_market\_order*, and *place\_stop\_order*.

**Parameters**

- **product\_id** (*str*) – Product to order (eg. ‘BTC-USD’)
- **order\_type** (*str*) – Order type (‘limit’, ‘market’, or ‘stop’)
- **\*\*kwargs** – Additional arguments can be specified for different order types.

**Returns** Order details. See *place\_order* for example.

**Return type** dict

**withdraw** (*amount*, *currency*, *payment\_method\_id*)

Withdraw funds to a payment method.

See `AuthenticatedClient.get_payment_methods()` to receive information regarding payment methods.

**Parameters**

- **amount** (*Decimal*) – The amount to withdraw.
- **currency** (*str*) – Currency type (eg. ‘BTC’)
- **payment\_method\_id** (*str*) – ID of the payment method.

**Returns**

**Withdraw details. Example::**

```
{ "id": "593533d2-ff31-46e0-b22e-ca754147a96a", "amount": "10.00", "currency": "USD", "payout_at": "2016-08-20T00:31:09Z" }
```

**Return type** dict

## 1.3 Authentication

### Contents

- [Authentication](#)

– *Authentication*

### 1.3.1 Authentication

**class** cbpro.cbpro\_auth.CBProAuth(*api\_key*, *secret\_key*, *passphrase*)

## 1.4 Order Book

### Contents

- *Order Book*
  - *Order Book*

### 1.4.1 Order Book

## 1.5 Websocket Client

### Contents

- *Websocket Client*
  - *Websocket Client*

### 1.5.1 Websocket Client





### C

`cbpro.authenticated_client`, 6  
`cbpro.cbpro_auth`, 19  
`cbpro.order_book`, 19  
`cbpro.public_client`, 3  
`cbpro.websocket_client`, 19



## A

auth (cbpro.authenticated\_client.AuthenticatedClient attribute), 7  
AuthenticatedClient (class in cbpro.authenticated\_client), 6

## B

buy() (cbpro.authenticated\_client.AuthenticatedClient method), 7

## C

cancel\_all() (cbpro.authenticated\_client.AuthenticatedClient method), 7  
cancel\_order() (cbpro.authenticated\_client.AuthenticatedClient method), 7  
cbpro.authenticated\_client (module), 6  
cbpro.cbpro\_auth (module), 19  
cbpro.order\_book (module), 19  
cbpro.public\_client (module), 3  
cbpro.websocket\_client (module), 19  
CBProAuth (class in cbpro.cbpro\_auth), 19  
close\_position() (cbpro.authenticated\_client.AuthenticatedClient method), 8  
coinbase\_deposit() (cbpro.authenticated\_client.AuthenticatedClient method), 8  
coinbase\_withdraw() (cbpro.authenticated\_client.AuthenticatedClient method), 8  
create\_report() (cbpro.authenticated\_client.AuthenticatedClient method), 9  
crypto\_withdraw() (cbpro.authenticated\_client.AuthenticatedClient method), 9

## D

deposit() (cbpro.authenticated\_client.AuthenticatedClient method), 9

## G

get\_account() (cbpro.authenticated\_client.AuthenticatedClient method), 10  
get\_account\_history() (cbpro.authenticated\_client.AuthenticatedClient method), 10  
get\_account\_holds() (cbpro.authenticated\_client.AuthenticatedClient method), 11  
get\_accounts() (cbpro.authenticated\_client.AuthenticatedClient method), 11  
get\_coinbase\_accounts() (cbpro.authenticated\_client.AuthenticatedClient method), 12  
get\_currencies() (cbpro.public\_client.PublicClient method), 3  
get\_fills() (cbpro.authenticated\_client.AuthenticatedClient method), 12  
get\_fundings() (cbpro.authenticated\_client.AuthenticatedClient method), 13  
get\_order() (cbpro.authenticated\_client.AuthenticatedClient method), 13  
get\_orders() (cbpro.authenticated\_client.AuthenticatedClient method), 13  
get\_payment\_methods() (cbpro.authenticated\_client.AuthenticatedClient method), 14  
get\_position() (cbpro.authenticated\_client.AuthenticatedClient method), 14  
get\_product\_24hr\_stats() (cbpro.public\_client.PublicClient method), 4  
get\_product\_historic\_rates() (cbpro.public\_client.PublicClient method), 4  
get\_product\_order\_book() (cbpro.public\_client.PublicClient method), 4  
get\_product\_ticker()

```
        (cbpro.public_client.PublicClient    method),  
        5  
get_product_trades()  
        (cbpro.public_client.PublicClient    method),  
        5  
get_products() (cbpro.public_client.PublicClient  
               method), 6  
get_report() (cbpro.authenticated_client.AuthenticatedClient  
             method), 14  
get_time() (cbpro.public_client.PublicClient  
           method), 6  
get_trailing_volume()  
        (cbpro.authenticated_client.AuthenticatedClient  
        method), 15
```

## M

```
margin_transfer()  
        (cbpro.authenticated_client.AuthenticatedClient  
        method), 15
```

## P

```
place_limit_order()  
        (cbpro.authenticated_client.AuthenticatedClient  
        method), 15  
place_market_order()  
        (cbpro.authenticated_client.AuthenticatedClient  
        method), 16  
place_order() (cbpro.authenticated_client.AuthenticatedClient  
              method), 16  
place_stop_order()  
        (cbpro.authenticated_client.AuthenticatedClient  
        method), 17  
PublicClient (class in cbpro.public_client), 3
```

## R

```
repay_funding() (cbpro.authenticated_client.AuthenticatedClient  
               method), 18
```

## S

```
sell() (cbpro.authenticated_client.AuthenticatedClient  
       method), 18  
session(cbpro.authenticated_client.AuthenticatedClient  
       attribute), 7
```

## U

```
url (cbpro.authenticated_client.AuthenticatedClient at-  
    tribute), 7  
url (cbpro.public_client.PublicClient attribute), 3
```

## W

```
withdraw() (cbpro.authenticated_client.AuthenticatedClient  
           method), 18
```