Trading API (1.0.0)

Introduction

Welcome to Robinhood Crypto API documentation for traders and developers! The APIs let you view crypto market data, access your account information, and place crypto orders programmatically.

Interested in using our API? Get started with creating your keys on the Crypto Account Settings Page, available only on a desktop web browser.

Your use of the Robinhood Crypto API is subject to the Robinhood Crypto Customer Agreement as well as all other terms and disclosures made available on Robinhood Crypto's about page.

Getting Started

Developing your own application to place trades with your Robinhood account is quick and simple. Start here with the code you'll need to access the API, authenticate credentials, and make supported API calls. These are essentially the building blocks of code for each API call, which you can easily build on based on your preferred strategies.

1. Create a script file

```
mkdir robinhood-api-trading && cd robinhood-api-trading
touch robinhood_api_trading.py
```

2. Install PyNaCl library

```
pip install pynacl
```

3. Copy the script below into the newly created `robinhood_api_trading.py` file. Make sure to add your API key and secret key into the `API_KEY` and `BASE64_PRIVATE_KEY` variables.

```python
import base64
import datetime
import json
from typing import Any, Dict, Optional
import uuid
import requests
from nacl.signing import SigningKey

API_KEY = "ADD YOUR API KEY HERE"
BASE64_PRIVATE_KEY = "ADD YOUR PRIVATE KEY HERE"

class CryptoAPITrading:
    def __init__(self):
        self.api_key = API_KEY
        private_key_seed = base64.b64decode(BASE64_PRIVATE_KEY)
        self.private_key = SigningKey(private_key_seed)
        self.base_url = "https://trading.robinhood.com"

    @staticmethod
    def _get_current_timestamp() -> int:
        return int(datetime.datetime.now(tz=datetime.timezone.utc).timestamp())

    @staticmethod
    def get_query_params(key: str, *args: Optional[str]) -> str:
        if not args:
            return ""

        params = []
        for arg in args:
            params.append(f"{key}={arg}")

        return "?" + "&".join(params)

    def make_api_request(self, method: str, path: str, body: str = "") -> Any:
        timestamp = self._get_current_timestamp()
        headers = self.get_authorization_header(method, path, body, timestamp)
        url = self.base_url + path

        try:
            response = {}
            if method == "GET":
                response = requests.get(url, headers=headers, timeout=10)
            elif method == "POST":
                response = requests.post(url, headers=headers, json=json.loads(body), timeout=10)
            return response.json()
        except requests.RequestException as e:
            print(f"Error making API request: {e}")
            return None

    def get_authorization_header(
            self, method: str, path: str, body: str, timestamp: int
    ) -> Dict[str, str]:
        message_to_sign = f"{self.api_key}{timestamp}{path}{method}{body}"
        signed = self.private_key.sign(message_to_sign.encode("utf-8"))

        return {
            "x-api-key": self.api_key,
            "x-signature": base64.b64encode(signed.signature).decode("utf-8"),
            "x-timestamp": str(timestamp),
        }

    def get_account(self) -> Any:
        path = "/api/v1/crypto/trading/accounts/"
        return self.make_api_request("GET", path)

    # The symbols argument must be formatted in trading pairs, e.g "BTC-USD", "ETH-USD". If no symbols are provided,
    # all supported symbols will be returned
    def get_trading_pairs(self, *symbols: Optional[str]) -> Any:
        query_params = self.get_query_params("symbol", *symbols)
        path = f"/api/v1/crypto/trading/trading_pairs/{query_params}"
        return self.make_api_request("GET", path)

    # The asset_codes argument must be formatted as the short form name for a crypto, e.g "BTC", "ETH". If no asset
    # codes are provided, all crypto holdings will be returned
    def get_holdings(self, *asset_codes: Optional[str]) -> Any:
        query_params = self.get_query_params("asset_code", *asset_codes)
        path = f"/api/v1/crypto/trading/holdings/{query_params}"
        return self.make_api_request("GET", path)

    # The symbols argument must be formatted in trading pairs, e.g "BTC-USD", "ETH-USD". If no symbols are provided,
    # the best bid and ask for all supported symbols will be returned
    def get_best_bid_ask(self, *symbols: Optional[str]) -> Any:
```

```
        query_params = self.get_query_params("symbol", *symbols)
        path = f"/api/v1/crypto/marketdata/best_bid_ask/{query_params}"
        return self.make_api_request("GET", path)

    # The symbol argument must be formatted in a trading pair, e.g "BTC-USD", "ETH-USD"
    # The side argument must be "bid", "ask", or "both".
    # Multiple quantities can be specified in the quantity argument, e.g. "0.1,1,1.999".
    def get_estimated_price(self, symbol: str, side: str, quantity: str) -> Any:
        path = f"/api/v1/crypto/marketdata/estimated_price/?symbol={symbol}&side={side}&quantity={quantity}"
        return self.make_api_request("GET", path)

    def place_order(
            self,
            client_order_id: str,
            side: str,
            order_type: str,
            symbol: str,
            order_config: Dict[str, str],
    ) -> Any:
        body = {
            "client_order_id": client_order_id,
            "side": side,
            "type": order_type,
            "symbol": symbol,
            f"{order_type}_order_config": order_config,
        }
        path = "/api/v1/crypto/trading/orders/"
        return self.make_api_request("POST", path, json.dumps(body))

    def cancel_order(self, order_id: str) -> Any:
        path = f"/api/v1/crypto/trading/orders/{order_id}/cancel/"
        return self.make_api_request("POST", path)

    def get_order(self, order_id: str) -> Any:
        path = f"/api/v1/crypto/trading/orders/{order_id}/"
        return self.make_api_request("GET", path)

    def get_orders(self) -> Any:
        path = "/api/v1/crypto/trading/orders/"
        return self.make_api_request("GET", path)


def main():
    api_trading_client = CryptoAPITrading()
    print(api_trading_client.get_account())

    """
    BUILD YOUR TRADING STRATEGY HERE

    order = api_trading_client.place_order(
            str(uuid.uuid4()),
            "buy",
            "market",
            "BTC-USD",
            {"asset_quantity": "0.0001"}
    )
    """


if __name__ == "__main__":
    main()
```

4. Run your script from the command line

```
python robinhood_api_trading.py
```

Authentication

Authenticated requests must include all three `x-api-key`, `x-signature`, and `x-timestamp` HTTP headers.

Creating an API key

To use the Crypto Trading API, you must visit the Robinhood API Credentials Portal to create credentials. After creating credentials, you will receive the API key associated with the credential. You can modify, disable, and delete credentials you created at any time.

The API key obtained from the credentials portal will be used as the `x-api-key` header you will need to pass during authentication when calling our API endpoints. Additionally, you will need the public key generated in the Creating a key pair section to create your API credentials.

Creating a key pair

Below are example scripts on how to generate the public and private key pair. You'll need the public key for creating an API credential and the private key for authenticating requests. Remember to never share your private key with anyone. Robinhood will never ask you to share it with us.

We highly recommend saving your private and public keys in an encrypted format to ensure the highest level of security. Encrypting your keys will protect them from unauthorized access or theft. Avoid saving them in plain text or any easily accessible location. Instead, consider using strong encryption algorithms or tools specifically designed for key storage. Remember to choose a strong passphrase for encryption and store it separately from your keys. By taking these precautions, you can significantly reduce the risk of compromising your keys and safeguard your sensitive information.

Node.js

Note that you'll need to have `tweetnacl` and `base64-js` installed to run the Node.js script. You can install them with the following npm command in your terminal.

```
npm install tweetnacl base64-js
```

```
const nacl = require('tweetnacl')
const base64 = require('base64-js')

// Generate an Ed25519 keypair
const keyPair = nacl.sign.keyPair()

// Convert keys to base64 strings
const private_key_base64 = base64.fromByteArray(keyPair.secretKey)
const public_key_base64 = base64.fromByteArray(keyPair.publicKey)

// Print keys in the base64 format
console.log("Private Key (Base64)")
console.log(private_key_base64)

console.log("Public Key (Base64):")
console.log(public_key_base64)
```

Python

Note that you'll need to have `pynacl` installed to run the Python script. You can install them with the following pip command in your terminal.

```
pip install pynacl
```

```
import nacl.signing
import base64

# Generate an Ed25519 keypair
private_key = nacl.signing.SigningKey.generate()
public_key = private_key.verify_key

# Convert keys to base64 strings
private_key_base64 = base64.b64encode(private_key.encode()).decode()
public_key_base64 = base64.b64encode(public_key.encode()).decode()

# Print the keys in base64 format
print("Private Key (Base64):")
print(private_key_base64)

print("Public Key (Base64):")
print(public_key_base64)
```

Headers and Signature

API Key

The " `x-api-key` " header should contain your API key. This API key is obtained from the Robinhood API Credentials Portal when enrolling in the Robinhood Crypto API program.

**API keys issued after August 13, 2024 will be formatted as "`rh-api-[uuid]`." Functionality will remain the same, and older keys will keep the original formatting (no "`rh-api`" prepend).*

Security Scheme Type: API Key

Header parameter name: `x-api-key`

API Signature

Authenticated requests should be signed with the "`x-signature`" header, using a signature generated with the following: private key, API key, timestamp, path, method, and body. Here's how the message signature should be defined:

```
message = f"{api_key}{current_timestamp}{path}{method}{body}"
```

**Note that for requests without a body, the body can be omitted from the message signature.*

Example Signature

The following is an example of a signature that corresponds to a cancel order request. You may use the example values below to ensure your code implementation is generating the same (`x-signature`) header signature value. The code snippet is for generating the signature in Python.

Field	Value
Private Key	xQnTJVeQLmw1/Mg2YimEViSpw/SdJcgNXZ5kQkAXNPU=
Public Key	jPItx4TLjcnSUnmnXQQyAKL4eJj3+oWNNMmmm2vATqk=
API Key	rh-api-6148effc-c0b1-486c-8940-a1d099456be6
Method	POST
Path	/api/v1/crypto/trading/orders/
Body	{"client_order_id":"131de903-5a9c-4260-abc1-28d562a5dcf0","side":"buy","type":"market","symbol":"BTC-USD","market_order_config":{"asset_quantity":"0.1"}}
Timestamp	1698708981
API Signature Header	**q/nEtxp/P2Or3hph3KejBqnw5o9qeuQ+hYRnB56FaHbjDsNUY9KhB1asMxohDnzdVFSD7StaTqjSd9U9HvaRAw==**

Code Example

Python

```
import nacl.signing
import base64
import time

# Replace these with your base64-encoded keys
private_key_base64 = "xQnTJVeQLmw1/Mg2YimEViSpw/SdJcgNXZ5kQkAXNPU="
public_key_base64 = "jPItx4TLjcnSUnmnXQQyAKL4eJj3+oWNNMmmm2vATqk="

api_key = "rh-api-6148effc-c0b1-486c-8940-a1d099456be6"

# Get the current Unix timestamp in seconds
# You can get the current_timestamp with the following code:
# current_timestamp = int(time.time())
# This value is hardcoded for demonstration purposes to match the example in the documentation
current_timestamp = "1698708981"

path = f"/api/v1/crypto/trading/orders/"

method = "POST"

body = {
    "client_order_id": "131de903-5a9c-4260-abc1-28d562a5dcf0",
    "side": "buy",
    "symbol": "BTC-USD",
    "type": "market",
    "market_order_config": {
        "asset_quantity": "0.1"
    },
}

# Convert base64 strings to seed (for private key) and bytes (for public key)
private_key_seed = base64.b64decode(private_key_base64)
public_key_bytes = base64.b64decode(public_key_base64)

# Create private key (from seed) and public key (from bytes)
```

```
private_key = nacl.signing.SigningKey(private_key_seed)
public_key = nacl.signing.VerifyKey(public_key_bytes)

# Create the message to sign
message = f"{api_key}{current_timestamp}{path}{method}{body}"

# Sign the message
signed = private_key.sign(message.encode("utf-8"))

base64_signature = base64.b64encode(signed.signature).decode("utf-8")
print(base64_signature)

# Verify the signature
public_key.verify(signed.message, signed.signature)
```

Security Scheme Type: API Key

Header parameter name: `x-signature`

Timestamp

The " `x-timestamp` " header should contain the timestamp of the request.

To protect the security and integrity of your API interactions, timestamps are only valid for 30 seconds after they're generated, and an expired timestamp will be rejected. Therefore, ensure that your requests are made promptly after generating the timestamp.

Below is an example in Python on how to generate the timestamp.

```
import time

current_timestamp = int(time.time())
print(current_timestamp)
```

Security Scheme Type: API Key

Header parameter name: `x-timestamp`

Pagination

Endpoints that return a list of results allow you to paginate to have more control over how many and which results to display.

Pagination Parameters

Parameter	Description
cursor	The cursor is a unique identifier for each page in a list of results. The cursor is used to paginate through the pages of results.
limit	The limit is the number of items to return in a single page. Some of our endpoints support this query parameter. You can view support by checking the query parameters in the documentation of each endpoint.

Pagination Response

Field	Description
next	The API request endpoint that includes the next cursor query parameter to use for pagination.
previous	The API request endpoint that includes the previous cursor query parameter to use for pagination.
results	The list of response items for the current cursor.

Rate Limiting

Rate Limits

- Requests per minute per user account: 100
- Requests per minute per user account in bursts: 300

Rate limiting is applied using a token bucket implementation. The burst size or `capacity` is the number of tokens you can use to call an endpoint. This capacity is initialized at the maximum capacity and will be refilled using a `refill amount` at a timed interval called `refill interval` until the max capacity is once again reached.

Rate Limiting Terms

Term	Description
Max capacity	The maximum amount of tokens allowed. Will no longer continue refilling if this amount is reached.
Remaining amount	The number of tokens remaining that can be consumed to call an endpoint.
Refill amount	The number of tokens that are refilled at each refill interval.
Refill interval	The timed interval at which the tokens are refilled.

The actual values of the configuration will fluctuate depending on the availability of our service and our current expected volume at the time of service. Rate limits are applied per endpoint and may differ among each endpoint depending on their expected use case.

Example rate limiting configuration:

Term	Value
Max capacity	5
Remaining amount	2
Refill amount	1
Refill interval	1 second

Action	Time (in seconds)	Remaining amount	Description
Initialize	0	5	Initial state of the token bucket.
Endpoint call 1	0.5	4	One token consumed for calling endpoint 1.
Endpoint call 2	0.7	3	One token consumed for calling endpoint 2.
Refill	1	4	One token refilled at refill interval.
Refill	2	5	One token refilled at refill interval.
No refill	3	5	No refill since max capacity has been reached and no endpoints were called.
Endpoint call 3	3.5	4	One token consumed for calling endpoint 3.

Error Responses

Error response format

Type

The `type` field in the error response will be mapped to the following:

Error type	Status codes
validation_error	400
client_error	4XX, excluding 400
server_error	5XX

Errors

The `errors` field will contain a list of error details, each item will contain a nested `attr` and `detail` field.

Field	Description
attr	Error types of `validation_error` will specify the field name or `non_field_errors` if the error cannot be attributed to a field. Will be `null` for error types of either `client_error` and `server_error`.
detail	Will contain a human readable string describing the error.

Example Error Response

Here's a sample error response where the `client_order_id` field in the payload was a value that was not expected when calling the ***Add Crypto Order*** endpoint. The `detail` field for each error in the `errors` list will help understand why the `validation_error` was thrown. The `attr` field will indicate which field name in the request body or query parameter the error was thrown for if applicable.

```
{
  "type": "validation_error"
  "errors": [
    {
      "detail": "Must be a valid UUID.",
      "attr": "client_order_id"
    }
  ]
}
```


Common Error Status Codes

Status Code	Error
400	Bad request
401	Unauthorized
403	Forbidden
404	Not found
405	Method not allowed
406	Not acceptable
415	Unsupported media type
429	Too many requests
500	Internal server error
503	Service unavailable

Account

Get Crypto Trading Account Details

Fetches the Robinhood Crypto account details for the current user.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

Responses

> 200

GET /api/v1/crypto/trading/accounts/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/api/v1/crypto/trading/accounts/" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200

Content type
application/json

Copy

```
{
    "account_number": "string",
    "status": "active",
    "buying_power": "string",
    "buying_power_currency": "string"
}
```


Market Data

Get Best Price

Fetch a single bid and ask price per symbol, representing our current best price. This price does not take into account the order size, and may not be the final execution price.

The bid and ask prices are the prices our trading venues provide for an order, inclusive of a spread. The buy spread is the percent difference between the ask and the mid price. The sell spread is the percent difference between the bid and the mid price. Robinhood receives 0.45% of every executed order from its trading venues which is included in the spread.

AUTHORIZATIONS: >

(*API Key* and *API Signature* and *Timestamp*)

QUERY PARAMETERS

`symbol`

string
Example: `symbol=BTC-USD`
List of trading pair symbol(s) to retrieve data for.

Multiple symbols can be provided by using the "symbol" parameter with different values. Ensure that the symbol(s) are provided in all uppercase.

For example, **?symbol=BTC-USD&symbol=ETH-USD**

Responses

> **200**

> **400**

GET /api/v1/crypto/marketdata/best_bid_ask/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/api/v1/crypto/marketdata/best_bid_ask/?symbol=BTC-USD" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 400

Content type
application/json

Copy Expand all Collapse all

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

Get Estimated Price

This endpoint returns the estimated cost for a particular symbol, book side, and asset quantity. You can include a list of quantities in a single request to retrieve the price for various hypothetical order sizes.

The estimated price represents the expected execution price if you were to subsequently place an order. To estimate the cost for a Buy order, request an Ask quote. If you are preparing to place a Sell order, request a Bid quote. The execution price may vary due to market volatility and once executed the transaction may not be undone.

The bid and ask prices are the best prices our trading venues provide for an order, inclusive of a spread. The buy spread is the percent difference between the ask and the mid price. The sell spread is the percent difference between the bid and the mid price. Robinhood receives 0.45% of every executed order from its trading venues which is included in the spread.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

QUERY PARAMETERS

`symbol` required	string Example: `symbol=BTC-USD` A string matching a valid currency pair, for example BTC-USD. Ensure that the symbol is provided in all uppercase. Only USD symbols are accepted.
`side` required	string Enum: `"bid"` `"ask"` `"both"` Example: `side=bid` Order side. e.g. bid, ask, or both.
`quantity` required	string Example: `quantity=0.1,1,1.999` Quantity must be between `min_order_size` and `max_order_size` as defined in our `Get Crypto Trading Pairs` endpoint. A max of 10 quantities can be specified for each request.

Responses

> 200

> 400

GET /api/v1/crypto/marketdata/estimated_price/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/marketdata/api/v1/estimated_price/?symbol=BTC-USD&side=ask&quantity=0.1,1,1.999" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 400

Content type
application/json

Copy Expand all Collapse all

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

Trading

Get Crypto Trading Pairs

Fetch a list of trading pairs.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

QUERY PARAMETERS

`symbol`	string Example: `symbol=BTC-USD` List of trading pair symbol(s) to retrieve data for. Multiple symbols can be provided by using the "symbol" parameter with different values. Ensure that the symbol(s) are provided in all uppercase. For example, **?symbol=BTC-USD&symbol=ETH-USD**
`limit`	integer Limit the number of results per page size
`cursor`	string

Responses

> 200

> 400

> 404

GET /api/v1/crypto/trading/trading_pairs/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/api/v1/crypto/trading/trading_pairs/?symbol=BTC-USD" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 400 404

Content type
application/json

Copy Expand all Collapse all


```
{
   "next": "https://trading.robinhood.com/api/v1/crypto/trading/trading_pairs/?cursor={CURSOR_ID}",
   "previous": "https://trading.robinhood.com/api/v1/crypto/trading/trading_pairs/?cursor={CURSOR_ID}",
 - "results": [
    + { … }
   ]
}
```


Get Crypto Holdings

Fetch a list of holdings for the current user.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

QUERY PARAMETERS

asset_code	string Example: `asset_code=BTC` The asset code(s) to retrieve data for. Multiple asset codes can be provided by using the "asset_code" parameter with different values. Ensure that the asset codes are provided in all uppercase. For example, **?asset_code=BTC&asset_code=ETH**
limit	integer Limit the number of results per page size
cursor	string

Responses

> 200

> 400

> 404

GET /api/v1/crypto/trading/holdings/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/api/v1/crypto/trading/holdings/?asset_code=BTC" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
```

```
  -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
  -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 | 400 | 404

Content type
application/json

Copy Expand all Collapse all


```
{
   "next": "https://trading.robinhood.com/api/v1/crypto/trading/holdings?cursor={CURSOR_ID}",
   "previous": "https://trading.robinhood.com/api/v1/crypto/trading/holdings?cursor={CURSOR_ID}",
 - "results": [
     + { … }
   ]
}
```


Get Crypto Orders

Fetch a list of orders for the current user.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

QUERY PARAMETERS

`created_at_start`	string Example: `created_at_start=2023-10-31T20:57:50Z` Filter by created at start time range (greater than or equal to) in ISO 8601 format
`created_at_end`	string Example: `created_at_end=2023-10-31T20:57:50Z` Filter by created at end time range (less than or equal to) in ISO 8601 format
`symbol`	string Example: `symbol=BTC-USD` Currency pair symbol. Ensure that the symbol is provided in all uppercase.
`id`	string <uuid> Order ID, unique per order
`side`	string Enum: `"buy"` `"sell"` Buy or sell
`state`	string Enum: `"open"` `"canceled"` `"partially_filled"` `"filled"` `"failed"` State of the order
`type`	string Enum: `"limit"` `"market"` `"stop_limit"` `"stop_loss"` Type of order
`updated_at_start`	string Example: `updated_at_start=2023-10-31T20:57:50Z` Filter by updated at start time range (greater than or equal to) in ISO 8601 format
`updated_at_end`	string Example: `updated_at_end=2023-10-31T20:57:50Z` Filter by updated at end time range (less than or equal to) in ISO 8601 format
`cursor`	string
`limit`	integer Limit the number of results per page size

Responses

> 200

> 400

› 404

GET /api/v1/crypto/trading/orders/

Request samples

cURL

Copy

```
curl -X "GET" "https://trading.robinhood.com/api/v1/crypto/trading/orders/" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 400 404

Content type
application/json

Copy Expand all Collapse all


```
{
   "next": "https://trading.robinhood.com/api/v1/crypto/trading/orders/?cursor={CURSOR_ID}",
   "previous": "https://trading.robinhood.com/api/v1/crypto/trading/orders/?cursor={CURSOR_ID}",
 - "results": [
    + { … }
   ]
}
```


Place New Crypto Order

Places a new crypto trading order with an order type.

Note: Depending on the type used in the request body, you must include the respective order configuration in the request body.

For order configurations that support both `asset_quantity` or `quote_amount`, only one can be present in the request body.

AUTHORIZATIONS: › (*API Key* and *API Signature* and *Timestamp*)

REQUEST BODY SCHEMA: application/json
required

`symbol` required	string (Symbol) Example: **BTC-USD** Currency pair symbol. Ensure that the symbol is provided in all uppercase.
`client_order_id` required	string <uuid> (Client order ID) User input order id for idempotency validation
`side` required	string (Side) Enum: `"buy"` `"sell"` Buy or sell
`type` required	string Enum: `"limit"` `"market"` `"stop_limit"` `"stop_loss"` Type of order
`market_order_config` ›	object Market order configuration ***Required for market orders**
`limit_order_config` ›	object Limit order configuration ***Required for limit orders**
`stop_loss_order_config` ›	object Stop loss order configuration

*Required for stop loss orders

stop_limit_order_config ›

object
Stop limit order configuration

*Required for stop limit orders

Responses

› 201

› 400

POST /api/v1/crypto/trading/orders/

Request samples

Payload | cURL

Content type
application/json

Copy Expand all Collapse all


```
{
   "symbol": "string",
   "client_order_id": "11299b2b-61e3-43e7-b9f7-dee77210bb29",
   "side": "buy",
   "type": "limit",
 - "market_order_config": {
       "asset_quantity": 0
   },
 - "limit_order_config": {
       "quote_amount": 0,
       "asset_quantity": 0,
       "limit_price": 0,
       "time_in_force": "gtc"
   },
 - "stop_loss_order_config": {
       "quote_amount": 0,
       "asset_quantity": 0,
       "stop_price": 0,
       "time_in_force": "gtc"
   },
 - "stop_limit_order_config": {
       "quote_amount": 0,
       "asset_quantity": 0,
       "limit_price": 0,
       "stop_price": 0,
       "time_in_force": "gtc"
   }
}
```


Response samples

201 | 400

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "497f6eca-6276-4993-bfeb-53cbbbba6f08",
   "account_number": "string",
   "symbol": "string",
   "client_order_id": "11299b2b-61e3-43e7-b9f7-dee77210bb29",
   "side": "buy",
 - "executions": [
     + { … }
   ],
   "type": "limit",
   "state": "open",
```

```
    "average_price": 0,
    "filled_asset_quantity": 0,
    "created_at": "string",
    "updated_at": "string",
  - "market_order_config": {
        "asset_quantity": 0
    },
  - "limit_order_config": {
        "quote_amount": 0,
        "asset_quantity": 0,
        "limit_price": 0,
        "time_in_force": "gtc"
    },
  - "stop_loss_order_config": {
        "quote_amount": 0,
        "asset_quantity": 0,
        "stop_price": 0,
        "time_in_force": "gtc"
    },
  - "stop_limit_order_config": {
        "quote_amount": 0,
        "asset_quantity": 0,
        "limit_price": 0,
        "stop_price": 0,
        "time_in_force": "gtc"
    }
}
```


Cancel Open Crypto Order

Cancels an open crypto trading order.

AUTHORIZATIONS: > (*API Key* and *API Signature* and *Timestamp*)

PATH PARAMETERS

id required	string <uuid> Order ID, unique per order

Responses

> 200

> 400

> 404

POST /api/v1/crypto/trading/orders/{id}/cancel/

Request samples

cURL

Copy

```
curl -X "POST" "https://trading.robinhood.com/api/v1/crypto/trading/orders/{id}/cancel/" \
    -H 'x-api-key: <YOUR-API-KEY>' \
    -H 'x-timestamp: <timestamp unix>' \
    -H 'x-signature: <YOUR-MSG-SIGNATURE>' \
    -H 'Content-Type: application/json; charset=utf-8'
```

Response samples

200 400 404

Content type

text/plain

Copy

```
Cancel request was submitted for order {id}
```