**Public Rest API for Binance (2018-01-14)**

**General API Information**

* The base endpoint is: [**https://api.binance.com**](https://api.binance.com)
* All endpoints return either a JSON object or array.
* Data is returned in **ascending** order. Oldest first, newest last.
* All time and timestamp related fields are in milliseconds.
* HTTP 4XX return codes are used for for malformed requests; the issue is on the sender's side.
* HTTP 429 return code is used when breaking a request rate limit.
* HTTP 418 return code is used when an IP has been auto-banned for continuing to send requests after receiving 429 codes.
* HTTP 5XX return codes are used for internal errors; the issue is on Binance's side.
* HTTP 504 return code is used when the API successfully sent the message but not get a response within the timeout period. It is important to **NOT** treat this as a failure; the execution status is **UNKNOWN** and could have been a success.
* Any endpoint can retun an ERROR; the error payload is as follows:

{

"code": -1121,

"msg": "Invalid symbol."

}

* Specific error codes and messages defined in another document.
* For GET endpoints, parameters must be sent as a query string.
* For POST, PUT, and DELETE endpoints, the parameters may be sent as a query string or in the request body with content type application/x-www-form-urlencoded. You may mix parameters between both the query string and request body if you wish to do so.
* Parameters may be sent in any order.
* If a parameter sent in both the query string and request body, the query string parameter will be used.

**LIMITS**

* The /api/v1/exchangeInfo rateLimits array contains objects related to the exchange's REQUESTS and ORDER rate limits.
* A 429 will be returned when either rather limit is violated.
* Each route has a weight which determines for the number of requests each endpoint counts for. Heavier endpoints and endpoints that do operations on multiple symbols will have a heavier weight.
* When a 429 is recieved, it's your obligation as an API to back off and not spam the API.
* **Repeatedly violating rate limits and/or failing to back off after receiving 429s will result in an automated IP ban (http status 418).**
* IP bans are tracked and **scale in duration** for repeat offenders, **from 2 minutes to 3 days**.

**Endpoint security type**

* Each endpoint has a security type that determines the how you will interact with it.
* API-keys are passed into the Rest API via the X-MBX-APIKEY header.
* API-keys and secret-keys **are case sensitive**.
* API-keys can be configured to only access certain types of secure endpoints. For example, one API-key could be used for TRADE only, while another API-key can access everything except for TRADE routes.
* By default, API-keys can access all secure routes.

| **Security Type** | **Description** |
| --- | --- |
| NONE | Endpoint can be accessed freely. |
| TRADE | Endpoint requires sending a valid API-Key and signature. |
| USER\_DATA | Endpoint requires sending a valid API-Key and signature. |
| USER\_STREAM | Endpoint requires sending a valid API-Key. |
| MARKET\_DATA | Endpoint requires sending a valid API-Key. |

* TRADE and USER\_DATA endpoints are SIGNED endpoints.

**SIGNED (TRADE and USER\_DATA) Endpoint security**

* SIGNED endpoints require an additional parameter, signature, to be sent in the query string or request body.
* Endpoints use HMAC SHA256 signatures. The HMAC SHA256 signature is a keyed HMAC SHA256 operation. Use your secretKey as the key and totalParams as the value for the HMAC operation.
* The signature is **not case sensitive**.
* totalParams is defined as the query string concatenated with the request body.

**Timing security**

* A SIGNED endpoint also requires a parameter, timestamp, to be sent which should be the millisecond timestamp of when the request was created and sent.
* An additional parameter, recvWindow, may be sent to specific the number of milliseconds after timestamp the request is valid for. If recvWindow is not sent, **it defaults to 5000**.
* The logic is as follows:
* if (timestamp < (serverTime + 1000) && (serverTime - timestamp) <= recvWindow) {
* // process request
* } else {
* // reject request

}

**Serious trading is about timing.** Networks can be unstable and unreliable, which can lead to requests taking varying amounts of time to reach the servers. With recvWindow, you can specify that the request must be processed within a certain number of milliseconds or be rejected by the server.

**Tt recommended to use a small recvWindow of 5000 or less!**

**SIGNED Endpoint Examples for POST /api/v1/order**

Here is a step-by-step example of how to send a vaild signed payload from the Linux command line using echo, openssl, and curl.

| **Key** | **Value** |
| --- | --- |
| apiKey | vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A |
| secretKey | NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j |

| **Parameter** | **Value** |
| --- | --- |
| symbol | LTCBTC |
| side | BUY |
| type | LIMIT |
| timeInForce | GTC |
| quantity | 1 |
| price | 0.1 |
| recvWindow | 5000 |
| timestamp | 1499827319559 |

**Example 1: As a query string**

* **queryString:** symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559
* **HMAC SHA256 signature:**
* [linux]$ echo -n "symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559" | openssl dgst -sha256 -hmac "NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j"
* (stdin)= c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71
* **curl command:**
* (HMAC SHA256)
* [linux]$ curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X POST 'https://api.binance.com/api/v3/order?symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559&signature=c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71'

**Example 2: As a request body**

* **requestBody:** symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559
* **HMAC SHA256 signature:**
* [linux]$ echo -n "symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559" | openssl dgst -sha256 -hmac "NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j"
* (stdin)= c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71
* **curl command:**
* (HMAC SHA256)
* [linux]$ curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X POST 'https://api.binance.com/api/v3/order' -d 'symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=6000000&timestamp=1499827319559&signature=c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71'

**Example 3: Mixed query string and request body**

* **queryString:** symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC
* **requestBody:** quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559
* **HMAC SHA256 signature:**
* [linux]$ echo -n "symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTCquantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559" | openssl dgst -sha256 -hmac "NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j"
* (stdin)= 0fd168b8ddb4876a0358a8d14d0c9f3da0e9b20c5d52b2a00fcf7d1c602f9a77
* **curl command:**
* (HMAC SHA256)
* [linux]$ curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X POST 'https://api.binance.com/api/v3/order?symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC' -d 'quantity=1&price=0.1&recvWindow=6000000&timestamp=1499827319559&signature=0fd168b8ddb4876a0358a8d14d0c9f3da0e9b20c5d52b2a00fcf7d1c602f9a77'

Note that the signature is different in example 3. There is no & between "GTC" and "quantity=1".

**Public API Endpoints**

**Terminology**

* base asset refers to the asset that is the quantity of a symbol.
* quoate asset refers to the asset that is the price of a symbol.

**ENUM definitions**

**Symbol status:**

* PRE\_TRADING
* TRADING
* POST\_TRADING
* END\_OF\_DAY
* HALT
* AUCTION\_MATCH
* BREAK

**Symbol type:**

* SPOT

**Order status:**

* NEW
* PARTIALLY\_FILLED
* FILLED
* CANCELED
* PENDING\_CANCEL (currently unused)
* REJECTED
* EXPIRED

**Order types:**

* LIMIT
* MARKET
* STOP\_LOSS
* STOP\_LOSS\_LIMIT
* TAKE\_PROFIT
* TAKE\_PROFIT\_LIMIT
* LIMIT\_MAKER

**Order side:**

* BUY
* SELL

**Time in force:**

* GTC
* IOC
* FOK

**Kline/Candlestick chart intervals:**

m -> minutes; h -> hours; d -> days; w -> weeks; M -> months

* 1m
* 3m
* 5m
* 15m
* 30m
* 1h
* 2h
* 4h
* 6h
* 8h
* 12h
* 1d
* 3d
* 1w
* 1M

**Rate limiters (rateLimitType)**

* REQUESTS
* ORDERS

**Rate limit intervals**

* SECOND
* MINUTE
* DAY

**General endpoints**

**Test connectivity**

GET /api/v1/ping

Test connectivity to the Rest API.

**Weight:** 1

**Parameters:** NONE

**Response:**

{}

**Check server time**

GET /api/v1/time

Test connectivity to the Rest API and get the current server time.

**Weight:** 1

**Parameters:** NONE

**Response:**

{

"serverTime": 1499827319559

}

**Exchange information**

GET /api/v1/exchangeInfo

Current exchange trading rules and symbol information

**Weight:** 1

**Parameters:** NONE

**Response:**

{

"timezone": "UTC",

"serverTime": 1508631584636,

"rateLimits": [{

"rateLimitType": "REQUESTS",

"interval": "MINUTE",

"limit": 1200

},

{

"rateLimitType": "ORDERS",

"interval": "SECOND",

"limit": 10

},

{

"rateLimitType": "ORDERS",

"interval": "DAY",

"limit": 100000

}

],

"exchangeFilters": [],

"symbols": [{

"symbol": "ETHBTC",

"status": "TRADING",

"baseAsset": "ETH",

"baseAssetPrecision": 8,

"quoteAsset": "BTC",

"quotePrecision": 8,

"orderTypes": ["LIMIT", "MARKET"],

"icebergAllowed": false,

"filters": [{

"filterType": "PRICE\_FILTER",

"minPrice": "0.00000100",

"maxPrice": "100000.00000000",

"tickSize": "0.00000100"

}, {

"filterType": "LOT\_SIZE",

"minQty": "0.00100000",

"maxQty": "100000.00000000",

"stepSize": "0.00100000"

}, {

"filterType": "MIN\_NOTIONAL",

"minNotional": "0.00100000"

}]

}]

}

**Market Data endpoints**

**Order book**

GET /api/v1/depth

**Weight:** Adjusted based on the limit:

| **Limit** | **Weight** |
| --- | --- |
| 5, 10, 20, 50, 100 | 1 |
| 500 | 5 |
| 1000 | 10 |

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| limit | INT | NO | Default 100; max 1000. Valid limits:[5, 10, 20, 50, 100, 500, 1000] |

**Caution:** setting limit=0 can return a lot of data.

**Response:**

{

"lastUpdateId": 1027024,

"bids": [

[

"4.00000000", // PRICE

"431.00000000", // QTY

[] // Ignore.

]

],

"asks": [

[

"4.00000200",

"12.00000000",

[]

]

]

}

**Recent trades list**

GET /api/v1/trades

Get recent trades (up to last 500).

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| limit | INT | NO | Default 500; max 500. |

**Response:**

[

{

"id": 28457,

"price": "4.00000100",

"qty": "12.00000000",

"time": 1499865549590,

"isBuyerMaker": true,

"isBestMatch": true

}

]

**Old trade lookup (MARKET\_DATA)**

GET /api/v1/historicalTrades

Get older trades.

**Weight:** 5

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| limit | INT | NO | Default 500; max 500. |
| fromId | LONG | NO | TradeId to fetch from. Default gets most recent trades. |

**Response:**

[

{

"id": 28457,

"price": "4.00000100",

"qty": "12.00000000",

"time": 1499865549590,

"isBuyerMaker": true,

"isBestMatch": true

}

]

**Compressed/Aggregate trades list**

GET /api/v1/aggTrades

Get compressed, aggregate trades. Trades that fill at the time, from the same order, with the same price will have the quantity aggregated.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| fromId | LONG | NO | ID to get aggregate trades from INCLUSIVE. |
| startTime | LONG | NO | Timestamp in ms to get aggregate trades from INCLUSIVE. |
| endTime | LONG | NO | Timestamp in ms to get aggregate trades until INCLUSIVE. |
| limit | INT | NO | Default 500; max 500. |

* If both startTime and endTime are sent, limit should not be sent AND the distance between startTime and endTime must be less than 24 hours.
* If frondId, startTime, and endTime are not sent, the most recent aggregate trades will be returned.

**Response:**

[

{

"a": 26129, // Aggregate tradeId

"p": "0.01633102", // Price

"q": "4.70443515", // Quantity

"f": 27781, // First tradeId

"l": 27781, // Last tradeId

"T": 1498793709153, // Timestamp

"m": true, // Was the buyer the maker?

"M": true // Was the trade the best price match?

}

]

**Kline/Candlestick data**

GET /api/v1/klines

Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| interval | ENUM | YES |  |
| limit | INT | NO | Default 500; max 500. |
| startTime | LONG | NO |  |
| endTime | LONG | NO |  |

* If startTime and endTime are not sent, the most recent klines are returned.

**Response:**

[

[

1499040000000, // Open time

"0.01634790", // Open

"0.80000000", // High

"0.01575800", // Low

"0.01577100", // Close

"148976.11427815", // Volume

1499644799999, // Close time

"2434.19055334", // Quote asset volume

308, // Number of trades

"1756.87402397", // Taker buy base asset volume

"28.46694368", // Taker buy quote asset volume

"17928899.62484339" // Ignore

]

]

**24hr ticker price change statistics**

GET /api/v1/ticker/24hr

24 hour price change statistics. **Careful** when accessing this with no symbol.

**Weight:** 1 for a single symbol; **number of symbols that are TRADING / 2** when the symbol parameter is omitted

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | NO |  |

* If the symbol is not sent, tickers for all symbols will be returned in an array.
* When all symbols are returned, the number of requests counted against the rate limiter is equal to the number of symbols currently trading on the exchange.

**Response:**

{

"symbol": "BNBBTC",

"priceChange": "-94.99999800",

"priceChangePercent": "-95.960",

"weightedAvgPrice": "0.29628482",

"prevClosePrice": "0.10002000",

"lastPrice": "4.00000200",

"lastQty": "200.00000000",

"bidPrice": "4.00000000",

"askPrice": "4.00000200",

"openPrice": "99.00000000",

"highPrice": "100.00000000",

"lowPrice": "0.10000000",

"volume": "8913.30000000",

"quoteVolume": "15.30000000",

"openTime": 1499783499040,

"closeTime": 1499869899040,

"fristId": 28385, // First tradeId

"lastId": 28460, // Last tradeId

"count": 76 // Trade count

}

OR

[

{

"symbol": "BNBBTC",

"priceChange": "-94.99999800",

"priceChangePercent": "-95.960",

"weightedAvgPrice": "0.29628482",

"prevClosePrice": "0.10002000",

"lastPrice": "4.00000200",

"lastQty": "200.00000000",

"bidPrice": "4.00000000",

"askPrice": "4.00000200",

"openPrice": "99.00000000",

"highPrice": "100.00000000",

"lowPrice": "0.10000000",

"volume": "8913.30000000",

"quoteVolume": "15.30000000",

"openTime": 1499783499040,

"closeTime": 1499869899040,

"fristId": 28385, // First tradeId

"lastId": 28460, // Last tradeId

"count": 76 // Trade count

}

]

**Symbol price ticker**

GET /api/v3/ticker/price

Latest price for a symbol or symbols.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | NO |  |

* If the symbol is not sent, prices for all symbols will be returned in an array.

**Response:**

{

"symbol": "LTCBTC",

"price": "4.00000200"

}

OR

[

{

"symbol": "LTCBTC",

"price": "4.00000200"

},

{

"symbol": "ETHBTC",

"price": "0.07946600"

}

]

**Symbol order book ticker**

GET /api/v3/ticker/bookTicker

Best price/qty on the order book for a symbol or symbols.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | NO |  |

* If the symbol is not sent, bookTickers for all symbols will be returned in an array.

**Response:**

{

"symbol": "LTCBTC",

"bidPrice": "4.00000000",

"bidQty": "431.00000000",

"askPrice": "4.00000200",

"askQty": "9.00000000"

}

OR

[

{

"symbol": "LTCBTC",

"bidPrice": "4.00000000",

"bidQty": "431.00000000",

"askPrice": "4.00000200",

"askQty": "9.00000000"

},

{

"symbol": "ETHBTC",

"bidPrice": "0.07946700",

"bidQty": "9.00000000",

"askPrice": "100000.00000000",

"askQty": "1000.00000000"

}

]

**Account endpoints**

**New order (TRADE)**

POST /api/v3/order (HMAC SHA256)

Send in a new order.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| side | ENUM | YES |  |
| type | ENUM | YES |  |
| timeInForce | ENUM | NO |  |
| quantity | DECIMAL | YES |  |
| price | DECIMAL | NO |  |
| newClientOrderId | STRING | NO | A unique id for the order. Automatically generated if not sent. |
| stopPrice | DECIMAL | NO | Used with STOP\_LOSS, STOP\_LOSS\_LIMIT, TAKE\_PROFIT, and TAKE\_PROFIT\_LIMIT orders. |
| icebergQty | DECIMAL | NO | Used with LIMIT, STOP\_LOSS\_LIMIT, and TAKE\_PROFIT\_LIMIT to create an iceberg order. |
| newOrderRespType | ENUM | NO | Set the response JSON. ACK, RESULT, or FULL; default: RESULT. |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

Additional mandatory parameters based on type:

| **Type** | **Additional mandatory parameters** |
| --- | --- |
| LIMIT | timeInForce, quantity, price |
| MARKET | quantity |
| STOP\_LOSS | quantity, stopPrice |
| STOP\_LOSS\_LIMIT | timeInForce, quantity, price, stopPrice |
| TAKE\_PROFIT | quantity, stopPrice |
| TAKE\_PROFIT\_LIMIT | timeInForce, quantity, price, stopPrice |
| LIMIT\_MAKER | quantity, price |

Other info:

* LIMIT\_MAKER are LIMIT orders that will be rejected if they would immediately match and trade as a taker.
* STOP\_LOSS and TAKE\_PROFIT will execute a MARKET order when the stopPrice is reached.
* Any LIMIT or LIMIT\_MAKER type order can be made an iceberg order by sending an icebergQty.
* Any order with an icebergQty MUST have timeInForce set to GTC.

Trigger order price rules against market price for both MARKET and LIMIT versions:

* Price above market price: STOP\_LOSS BUY, TAKE\_PROFIT SELL
* Price below market price: STOP\_LOSS SELL, TAKE\_PROFIT BUY

**Response ACK:**

{

"symbol": "BTCUSDT",

"orderId": 28,

"clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",

"transactTime": 1507725176595

}

**Response RESULT:**

{

"symbol": "BTCUSDT",

"orderId": 28,

"clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",

"transactTime": 1507725176595,

"price": "0.00000000",

"origQty": "10.00000000",

"executedQty": "10.00000000",

"status": "FILLED",

"timeInForce": "GTC",

"type": "MARKET",

"side": "SELL"

}

**Response FULL:**

{

"symbol": "BTCUSDT",

"orderId": 28,

"clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",

"transactTime": 1507725176595,

"price": "0.00000000",

"origQty": "10.00000000",

"executedQty": "10.00000000",

"status": "FILLED",

"timeInForce": "GTC",

"type": "MARKET",

"side": "SELL",

"fills": [

{

"price": "4000.00000000",

"qty": "1.00000000",

"commission": "4.00000000",

"commissionAsset": "USDT"

},

{

"price": "3999.00000000",

"qty": "5.00000000",

"commission": "19.99500000",

"commissionAsset": "USDT"

},

{

"price": "3998.00000000",

"qty": "2.00000000",

"commission": "7.99600000",

"commissionAsset": "USDT"

},

{

"price": "3997.00000000",

"qty": "1.00000000",

"commission": "3.99700000",

"commissionAsset": "USDT"

},

{

"price": "3995.00000000",

"qty": "1.00000000",

"commission": "3.99500000",

"commissionAsset": "USDT"

}

]

}

**Test new order (TRADE)**

POST /api/v3/order/test (HMAC SHA256)

Test new order creation and signature/recvWindow long. Creates and validates a new order but does not send it into the matching engine.

**Weight:** 1

**Parameters:**

Same as /api/v3/order

**Response:**

{}

**Query order (USER\_DATA)**

GET /api/v3/order (HMAC SHA256)

Check an order's status.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| orderId | LONG | NO |  |
| origClientOrderId | STRING | NO |  |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

Either orderId or origClientOrderId must be sent.

**Response:**

{

"symbol": "LTCBTC",

"orderId": 1,

"clientOrderId": "myOrder1",

"price": "0.1",

"origQty": "1.0",

"executedQty": "0.0",

"status": "NEW",

"timeInForce": "GTC",

"type": "LIMIT",

"side": "BUY",

"stopPrice": "0.0",

"icebergQty": "0.0",

"time": 1499827319559,

"isWorking": true

}

**Cancel order (TRADE)**

DELETE /api/v3/order (HMAC SHA256)

Cancel an active order.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| orderId | LONG | NO |  |
| origClientOrderId | STRING | NO |  |
| newClientOrderId | STRING | NO | Used to uniquely identify this cancel. Automatically generated by default. |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

Either orderId or origClientOrderId must be sent.

**Response:**

{

"symbol": "LTCBTC",

"origClientOrderId": "myOrder1",

"orderId": 1,

"clientOrderId": "cancelMyOrder1"

}

**Current open orders (USER\_DATA)**

GET /api/v3/openOrders (HMAC SHA256)

Get all open orders on a symbol. **Careful** when accessing this with no symbol.

**Weight:** 1 for a single symbol; **number of symbols that are TRADING / 2** when the symbol parameter is omitted

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | NO |  |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

* If the symbol is not sent, orders for all symbols will be returned in an array.
* When all symbols are returned, the number of requests counted against the rate limiter is equal to the number of symbols currently trading on the exchange.

**Response:**

[

{

"symbol": "LTCBTC",

"orderId": 1,

"clientOrderId": "myOrder1",

"price": "0.1",

"origQty": "1.0",

"executedQty": "0.0",

"status": "NEW",

"timeInForce": "GTC",

"type": "LIMIT",

"side": "BUY",

"stopPrice": "0.0",

"icebergQty": "0.0",

"time": 1499827319559,

"isWorking": trueO

}

]

**All orders (USER\_DATA)**

GET /api/v3/allOrders (HMAC SHA256)

Get all account orders; active, canceled, or filled.

**Weight:** 5

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| orderId | LONG | NO |  |
| limit | INT | NO | Default 500; max 500. |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

* If orderId is set, it will get orders >= that orderId. Otherwise most recent orders are returned.

**Response:**

[

{

"symbol": "LTCBTC",

"orderId": 1,

"clientOrderId": "myOrder1",

"price": "0.1",

"origQty": "1.0",

"executedQty": "0.0",

"status": "NEW",

"timeInForce": "GTC",

"type": "LIMIT",

"side": "BUY",

"stopPrice": "0.0",

"icebergQty": "0.0",

"time": 1499827319559,

"isWorking": true

}

]

**Account information (USER\_DATA)**

GET /api/v3/account (HMAC SHA256)

Get current account information.

**Weight:** 5

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

**Response:**

{

"makerCommission": 15,

"takerCommission": 15,

"buyerCommission": 0,

"sellerCommission": 0,

"canTrade": true,

"canWithdraw": true,

"canDeposit": true,

"updateTime": 123456789,

"balances": [

{

"asset": "BTC",

"free": "4723846.89208129",

"locked": "0.00000000"

},

{

"asset": "LTC",

"free": "4763368.68006011",

"locked": "0.00000000"

}

]

}

**Account trade list (USER\_DATA)**

GET /api/v3/myTrades (HMAC SHA256)

Get trades for a specific account and symbol.

**Weight:** 5

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| symbol | STRING | YES |  |
| limit | INT | NO | Default 500; max 500. |
| fromId | LONG | NO | TradeId to fetch from. Default gets most recent trades. |
| recvWindow | LONG | NO |  |
| timestamp | LONG | YES |  |

**Response:**

[

{

"id": 28457,

"orderId": 100234,

"price": "4.00000100",

"qty": "12.00000000",

"commission": "10.10000000",

"commissionAsset": "BNB",

"time": 1499865549590,

"isBuyer": true,

"isMaker": false,

"isBestMatch": true

}

]

**User data stream endpoints**

Specifics on how user data streams work is in another document.

**Start user data stream (USER\_STREAM)**

POST /api/v1/userDataStream

Start a new user data stream. The stream will close after 60 minutes unless a keepalive is sent.

**Weight:** 1

**Parameters:** NONE

**Response:**

{

"listenKey": "pqia91ma19a5s61cv6a81va65sdf19v8a65a1a5s61cv6a81va65sdf19v8a65a1"

}

**Keepalive user data stream (USER\_STREAM)**

PUT /api/v1/userDataStream

Keepalive a user data stream to prevent a time out. User data streams will close after 60 minutes. It's recommended to send a ping about every 30 minutes.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| listenKey | STRING | YES |  |

**Response:**

{}

**Close user data stream (USER\_STREAM)**

DELETE /api/v1/userDataStream

Close out a user data stream.

**Weight:** 1

**Parameters:**

| **Name** | **Type** | **Mandatory** | **Description** |
| --- | --- | --- | --- |
| listenKey | STRING | YES |  |

**Response:**

{}

**Filters**

Filters define trading rules on a symbol or an exchange. Filters come in two forms: symbol filters and exchange filters.

**Symbol filters**

**PRICE\_FILTER**

The PRICE\_FILTER defines the price rules for a symbol. There are 3 parts:

* minPrice defines the minimum price/stopPrice allowed.
* maxPrice defines the maximum price/stopPrice allowed.
* tickSize defines the intervals that a price/stopPrice can be increased/decreased by.

In order to pass the price filter, the following must be true for price/stopPrice:

* price >= minPrice
* price <= maxPrice
* (price-minPrice) % tickSize == 0

**/exchangeInfo format:**

{

"filterType": "PRICE\_FILTER",

"minPrice": "0.00000100",

"maxPrice": "100000.00000000",

"tickSize": "0.00000100"

}

**LOT\_SIZE**

The LOT\_SIZE filter defines the quantity (aka "lots" in auction terms) rules for a symbol. There are 3 parts:

* minQty defines the minimum quantity/icebergQty allowed.
* maxQty defines the maximum quantity/icebergQty allowed.
* stepSize defines the intervals that a quantity/icebergQty can be increased/decreased by.

In order to pass the lot size, the following must be true for quantity/icebergQty:

* quantity >= minQty
* quantity <= maxQty
* (quantity-minQty) % stepSize == 0

**/exchangeInfo format:**

{

"filterType": "LOT\_SIZE",

"minQty": "0.00100000",

"maxQty": "100000.00000000",

"stepSize": "0.00100000"

}

**MIN\_NOTIONAL**

The MIN\_NOTIONAL filter defines the minimum notional value allowed for an order on a symbol. An order's notional value is the price \* quantity.

**/exchangeInfo format:**

{

"filterType": "MIN\_NOTIONAL",

"minNotional": "0.00100000"

}

**MAX\_NUM\_ORDERS**

The MAX\_NUM\_ORDERS filter defines the maximum number of orders an account is allowed to have open on a symbol. Note that both "algo" orders and normal orders are counted for this filter.

**/exchangeInfo format:**

{

"filterType": "MAX\_NUM\_ORDERS",

"limit": 25

}

**MAX\_ALGO\_ORDERS**

The MAX\_ALGO\_ORDERS filter defines the maximum number of "algo" orders an account is allowed to have open on a symbol. "Algo" orders are STOP\_LOSS, STOP\_LOSS\_LIMIT, TAKE\_PROFIT, and TAKE\_PROFIT\_LIMIT orders.

**/exchangeInfo format:**

{

"filterType": "MAX\_ALGO\_ORDERS",

"limit": 5

}

**Exchange Filters**

**EXCHANGE\_MAX\_NUM\_ORDERS**

The MAX\_NUM\_ORDERS filter defines the maximum number of orders an account is allowed to have open on the exchange. Note that both "algo" orders and normal orders are counted for this filter.

**/exchangeInfo format:**

{

"filterType": "EXCHANGE\_MAX\_NUM\_ORDERS",

"limit": 1000

}

**EXCHANGE\_MAX\_ALGO\_ORDERS**

The MAX\_ALGO\_ORDERS filter defines the maximum number of "algo" orders an account is allowed to have open on the exchange. "Algo" orders are STOP\_LOSS, STOP\_LOSS\_LIMIT, TAKE\_PROFIT, and TAKE\_PROFIT\_LIMIT orders.

**/exchangeInfo format:**

{

"filterType": "EXCHANGE\_MAX\_ALGO\_ORDERS",

"limit": 200

}