Introduction

Welcome to the Binance.US API Documentation!

Our **REST APIs** offer access to:

- Exchange data
- Market trade and price data
- User account data
- Trade order management
- Wallet management

Our **WebSocket APIs** offer access to:

- Market data
- Trade order management
- User account data
- User data stream

Our **WebSocket Steams** offer access to:

- Market data streams
- User data streams

Authentication

Get API Keys

IMPORTANT REMINDER FOR BINANCE.US API KEY USERS

- Do not disclose your API Key to anyone to avoid asset losses. It is recommended to bind IP for API Key to increase your account security.
- Be aware that your API Key may be disclosed by authorizing it to a third-party platform.
- You will not be able to create an API if KYC is not completed.
- Learn more about API key best practices and safety tips.

API KEY TYPES

Binance.US currently offers three API key types: Exchange API Keys, Custodial Solution API Keys, and Credit Line API Keys. Please read on for more information on the differences and instructions on how to set up your key type.

EXCHANGE API KEYS

- Private API keys **for the majority of API users** to interact with Binance.US API endpoints.
- Provides access to markets and real-time trading services on Binance.US via a third-party site or application.

Get Exchange API Keys

To create this API key type:

1. Log into Binance.US with your account details

2. From the profile icon drop-down menu > select 'API Management'

3. Enter a name for your API key for reference.

4. Click 'Create.' Enter your 2FA code to confirm when prompted

CUSTODIAL SOLUTION API KEYS

- Private API keys **only available** to users who have entered into a Custody Exchange Network agreement between a participating custody partner and Binance.US.
- Provides access to Custodial Solution related API endpoints only. To access other types of API endpoints, please generate other corresponding API keys.

Get Custodial Solution API Keys

After entering into a Custody Exchange Network agreement between a participating custody partner and Binance.US, users can create a Custodial Solution API key:

1. Log into Binance.US with your account details

2. From the profile icon drop-down menu > select 'API Management'

3. Select 'Custodial Solution API'' and give your API key a label for reference

4. Click 'Create.' Enter your 2FA code to confirm when prompted

CREDIT LINE API KEYS

- Private API keys **only available** to institutional users who have signed a credit line agreement with Binance.US.
- Provides access to Credit Line related API endpoints only. To access other types of API endpoints, please generate other corresponding API keys.

Get Credit Line API Keys

After signing a credit line agreement with Binance.US, users can create a Credit Line API key:

1. Log into Binance.US with your account details

2. From the profile icon drop-down menu > select 'API Management'

3. Select 'Credit Line API' and give your API key a label for reference

4. Click 'Create.' Enter your 2FA code to confirm when prompted

---

Authentication Types

- Each endpoint has a security type that determines how you will interact with it. This is stated next to the NAME of the endpoint.
  - If no security type is stated, assume the security type is NONE.
- 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

Authentication Timing

```
if (timestamp < (serverTime + 1000) && (serverTime - timestamp) <= recvWindow) {
  // process request
} else {
  // reject request
}
```

- A `SIGNED` endpoint also requires a parameter and `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 specify the number of milliseconds after the `timestamp` that the request is valid for. If `recvWindow` is not sent, **it defaults to 5,000**.
- The exact timing authentication logic can be viewed in the code sample on the right panel (or below on a mobile device).

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.

> It is recommended to use a small recvWindow of 5000 or less! The max cannot go beyond 60,000!

Signature Authentication

Example 1 As a request body

```
# request body
# symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559

# Get HMAC SHA256 signature
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 result
# c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71

# Request signed endpoints, /api/v3/order as an example
curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X POST 'https://api.binance.us/api/v3/order' -d 'symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC&quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559&sig
```

Example 2 As a query string

```
# query string
# symbol=BTCUSDT&timestamp=1499827319559

# Get HMAC SHA256 signature
echo -n "symbol=BTCUSDT&timestamp=1499827319559" | openssl dgst -sha256 -hmac "NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j"

# stdin result
# c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71

# Request signed endpoints, /api/v3/order as an example
curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X GET 'https://api.binance.us/api/v3/openOrders?symbol=BTCUSDT&timestamp=1499827319559&signature=c8db56825ae71d6d79447849e617115f4a920fa2acdcab2b053c4b2838bd6b71'
```

Example 3 Mixed query string and request body

```
# query string
# symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC

# request body
# quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559

# Get HMAC SHA256 signature
shell echo -n "symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTCquantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559" | openssl dgst -sha256 -hmac "NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j"

# stdin result
# 0fd168b8ddb4876a0358a8d14d0c9f3da0e9b20c5d52b2a00fcf7d1c602f9a77

# Request signed endpoints, /api/v3/order as an example
curl -H "X-MBX-APIKEY: vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A" -X POST 'https://api.binance.us/api/v3/order?symbol=LTCBTC&side=BUY&type=LIMIT&timeInForce=GTC' -d 'quantity=1&price=0.1&recvWindow=5000&timestamp=1499827319559&sig
```

- `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`.

Here is a step-by-step example of how to send a valid 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

Error Responses

Error Responses

```
{
  "code":-1121,
  "msg":"Invalid symbol."
}
```

Errors consist of two parts: an error code and a message. Codes are universal, but messages can vary. Here is the error JSON payload:

HTTP Errors

- HTTP `4XX` return codes are used for malformed requests; the issue is on the sender's side.
- HTTP `403` return code is used when the WAF (Web Application Firewall) Limit has been violated.
- HTTP `409` return code is used when a cancelReplace order partially succeeds. (i.e. if the cancellation of the order fails but the new order placement succeeds.)
- 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. It is important to **NOT** treat this as a failure operation; the execution status is **UNKNOWN** and could have been a success.

Server/Network Errors

- 1000 UNKNOWN

- An unknown error occurred while processing the request.

- 1001 DISCONNECTED

- Internal error; unable to process your request. Please try again.

- 1002 UNAUTHORIZED

- You are not authorized to execute this request.

- 1003 TOO_MANY_REQUESTS

- Too many requests queued.
- Too much request weight used; current limit is %s request weight per %s. Please use WebSocket Streams for live updates to avoid polling the API.
- Way too much request weight used; IP banned until %s. Please use WebSocket Streams for live updates to avoid bans.

- 1006 UNEXPECTED_RESP

- An unexpected response was received from the message bus. Execution status is unknown.

- 1007 TIMEOUT

- Timeout waiting for a response from the backend server. Send status unknown; execution status unknown.

- 1008 SERVER_BUSY

- Spot server is currently overloaded with other requests. Please try again in a few minutes.

- 1014 UNKNOWN_ORDER_COMPOSITION

- Unsupported order combination.

- 1015 TOO_MANY_ORDERS

- Too many new orders.
- Too many new orders; current limit is %s orders per %s.

- 1016 SERVICE_SHUTTING_DOWN

- This service is no longer available.

- 1020 UNSUPPORTED_OPERATION

- This operation is not supported.

- 1021 INVALID_TIMESTAMP

- Timestamp for this request is outside of the recvWindow.
- Timestamp for this request was 1000ms ahead of the server's time.

- 1022 INVALID_SIGNATURE

- Signature for this request is not valid.

Request Errors

- 1100 ILLEGAL_CHARS

- Illegal characters found in a parameter.
- Illegal characters found in parameter '%s'; the legal range is '%s'.

- 1101 TOO_MANY_PARAMETERS

- Too many parameters sent for this endpoint.
- Too many parameters; expected '%s' and received '%s'.
- Duplicate values for a parameter detected.

- 1102 MANDATORY_PARAM_EMPTY_OR_MALFORMED

- A mandatory parameter was not sent, was empty/null, or was malformed.
- Mandatory parameter '%s' was not sent, was empty/null, or was malformed.
- Param '%s' or '%s' must be sent, but both were empty/null.

- 1103 UNKNOWN_PARAM

- An unknown parameter was sent.

- 1104 UNREAD_PARAMETERS

- Not all sent parameters were read.
- Not all sent parameters were read; read '%s' parameter(s) but was sent '%s'.

- 1105 PARAM_EMPTY

- A parameter was empty.
- Parameter '%s' was empty.

- 1106 PARAM_NOT_REQUIRED

- A parameter was sent when not required.
- Parameter '%s' sent when not required.

- 1108 PARAM_OVERFLOW

- Parameter '%s' overflowed.

- 1111 BAD_PRECISION

- Precision is over the maximum defined for this asset.

- 1112 NO_DEPTH

- No orders on the book for this symbol.

- 1114 TIF_NOT_REQUIRED

- TimeInForce parameter sent when not required.

- 1115 INVALID_TIF

- Invalid timeInForce.

- 1116 INVALID_ORDER_TYPE

- Invalid orderType.

- 1117 INVALID_SIDE

- Invalid side.

- 1118 EMPTY_NEW_CL_ORD_ID

- New client order ID was empty.

- 1119 EMPTY_ORG_CL_ORD_ID

- Original client order ID was empty.

- 1120 BAD_INTERVAL

- Invalid interval.

- 1121 BAD_SYMBOL

- Invalid symbol.

- 1125 INVALID_LISTEN_KEY

- This listenKey does not exist.

- 1127 MORE_THAN_XX_HOURS

- Lookup interval is too big.
- More than %s hours between startTime and endTime.

- 1128 OPTIONAL_PARAMS_BAD_COMBO

- Combination of optional parameters invalid.

- 1130 INVALID_PARAMETER

- Invalid data sent for a parameter.
- Data sent for p arameter '%s' is not valid.

- 1135 INVALID_JSON

- Invalid JSON Request
- JSON sent for parameter '%s' is not valid

- 1145 INVALID_CANCEL_RESTRICTIONS

- `cancelRestrictions` has to be either `ONLY_NEW` or `ONLY_PARTIALLY_FILLED`.

- 2010 NEW_ORDER_REJECTED

- NEW_ORDER_REJECTED

- 2011 CANCEL_REJECTED

- CANCEL_REJECTED

- 2013 NO_SUCH_ORDER

- Order does not exist.

- 2014 BAD_API_KEY_FMT

- API-key format invalid.

- 2015 REJECTED_MBX_KEY

- Invalid API-key, IP, or permissions for action.

- 2016 NO_TRADING_WINDOW

- No trading window could be found for the symbol. Try ticker/24hrs instead.

- 2021 Order cancel-replace partially failed

- This code is sent when either the cancellation of the order failed or the new order placement failed but not both.

- 2022 Order cancel-replace failed.

- This code is sent when both the cancellation of the order failed and the new order placement failed.

- 2026 ORDER_ARCHIVED

- Order was canceled or expired with no executed qty over 90 days ago and has been archived.

Matching Engine Errors

Messages for -1010 ERROR_MSG_RECEIVED, -2010 NEW_ORDER_REJECTED, and -2011 CANCEL_REJECTED

This code is sent when an error has been returned by the matching engine. The following messages will indicate the specific error:

Error message	Description
"Unknown order sent"	The order (by either `orderId`, `clOrdId`, `origClOrdId`) could not be found
"Duplicate order sent"	The `clOrdId` is already in use
"Market is closed"	The symbol is not trading
"Account has insufficient balance for requested action"	Not enough funds to complete the action
"Market orders are not supported for this symbol"	`MARKET` is not enabled on the symbol
"Iceberg orders are not supported for this symbol"	`icebergQty` is not enabled on the symbol
"Stop loss orders are not supported for this symbol"	`STOP_LOSS` is not enabled on the symbol
"Stop loss limit orders are not supported for this symbol"	`STOP_LOSS_LIMIT` is not enabled on the symbol
"Take profit orders are not supported for this symbol"	`TAKE_PROFIT` is not enabled on the symbol
"Take profit limit orders are not supported for this symbol"	`TAKE_PROFIT_LIMIT` is not enabled on the symbol
"Price * QTY is zero or less"	`Price` * `quantity` is too low
"IcebergQty exceeds QTY"	`icebergQty` must be less than the order quantity
"This action is disabled on this account"	Contact customer support; some actions have been disabled on the account

Error message	Description
"Unsupported order combination"	The `orderType`, `timeInForce`, `stopPrice`, and/or `icebergQty` combination isn't allowed
"Order would trigger immediately"	The order's stop price is not valid compared to the last traded price
"Cancel order is invalid. Check origClOrdId and orderId."	No `origClOrdId` or `orderId` was sent in
"Order would immediately match and take"	`LIMIT_MAKER` order type would immediately match and trade, and not be a pure maker order
"The relationship of the prices for the orders is not correct"	The prices set in the `OCO` are breaking the Price rules. The rules are: `SELL Orders`: Limit Price > Last Price > Stop Price `BUY Orders`: Limit Price < Last Price < Stop Price
"OCO orders are not supported for this symbol"	`OCO` is not enabled on the symbol
"Quote order qty market orders are not support for this symbol"	`MARKET` orders using the parameter `quoteOrderQty` are not enabled on this symbol
"Trailing stop orders are not supported for this symbol."	Orders using `trailingDelta` are not enabled on the symbol.
"Order cancel-replace is not supported for this symbol."	`POST /api/v3/order/cancelReplace` is not enabled for the symbol.
"This symbol is not permitted for this account."	Account does not have permission to trade on this symbol.
"This symbol is restricted for this account."	Account does not have permission to trade on this symbol.
"Order was not canceled due to cancel restrictions."	Either `cancelRestrictions` was set to `ONLY_NEW` but the order status was not `NEW` or `cancelRestrictions` was set to `ONLY_PARTIALLY_FILLED` but the order status was not `PARTIALLY_FILLED`.

Filter Failure Errors

Error message	Description
"Filter failure: PRICE_FILTER"	`Price` is too high, too low, and/or not following the tick size rule for the symbol
"Filter failure: PERCENT_PRICE"	`Price` is X% too high or X% too low from the average weighted price over the last Y minutes
"Filter failure: LOT_SIZE"	`Quantity` is too high, too low, and/or not following the step size rule for the symbol
"Filter failure: MIN_NOTIONAL"	`Price` * `Quantity` is too low to be a valid order for the symbol
"Filter failure: ICEBERG_PARTS"	`ICEBERG` order would break into too many parts; icebergQty is too small
"Filter failure: MARKET_LOT_SIZE"	`MARKET` order's `quantity` is too high, too low, and/or not following the step size rule for the symbol
"Filter failure: MAX_NUM_ORDERS"	Account has too many open orders on the symbol
"Filter failure: MAX_ALGO_ORDERS"	Account has too many open stop loss and/or take profit orders on the symbol
"Filter failure: MAX_NUM_ICEBERG_ORDERS"	Account has too many open iceberg orders on the symbol
"Filter failure: TRAILING_DELTA"	`trailingDelta` is not within the defined range of the filter for that order type
"Filter failure: EXCHANGE_MAX_NUM_ORDERS"	Account has too many open orders on the exchange
"Filter failure: EXCHANGE_MAX_ALGO_ORDERS"	Account has too many open stop loss and/or take profit orders on the exchange

Changelog

13/10/2023

Removed 'Quick Enable Crypto Withdrawal' and 'Quick Disable Crypto Withdrawal' endpoints.

20/9/2023

- Added 'Withdraw Fiat via BITGO' endpoint

13/9/2023

- Updated 'Get Credit Line Account Information (C.L.)' and 'Get Alert History (C.L.)' endpoints to support multi-asset loan.

6/9/2023

- Removed 1s candlestick interval

20/7/2023

- Removed order related Credit Line endpoints

26/6/2023

- Added 'API Partner Endpoints' section and updated 'Get Asset Distribution History' endpoint for API Partner Program.

9/6/2023

- Removed USD from Convert Dust

12/5/2023

- Spot WebSocket APIs are now available for Binance US.
    - WebSocket API allows placing orders, canceling orders, etc. through a WebSocket connection.
    - WebSocket API is a **separate** service from WebSocket Market Data streams. i.e., placing orders and listening to market data requires two separate WebSocket connections.
    - WebSocket API is subject to the same Filter and Rate Limit rules as REST API.
    - WebSocket API and REST API are functionally equivalent: they provide the same features, accept the same parameters, return the same status and error codes.
- The full documentation can be found here.

11/5/2023

- Trading parameters for 3 trading pairs have been updated. Click here to learn more.

17/4/2023

- Updated convert dust endpoints to support dust conversion to BNB/BTC/ETH/USD.

3/4/2023

- Removed 'Withdraw Fiat (via SIGNET)' endpoint

16/3/2023

- Improved error messages for certain issues for easier troubleshooting.

Situation	Old Error Message	New Error Message
Account trading ability disabled (cannot place or cancel an order).	This action is disabled on this account.	This account may not place or cancel orders.
Permissions configured on the symbol do not match the permissions on the account.		This symbol is not permitted for this account.
Account tries to place an order on a symbol it has no permissions for.		This symbol is restricted for this account.
Placing an order when `symbol` is not `TRADING`.	Unsupported order combination.	This order type is not possible in this trading phase.
Placing an order with `timeinForce=IOC` or `FOK` on a non-supported trading phase.		Limit orders require GTC for this phase.

- Fixed error message for querying archived orders(status `CANCELED` or `EXPIRED` where `executedQty` == 0 in the last 90 days):
  - Previous error message:
    `{`
    `"code": -2013,`
    `"msg": "Order does not exist."`
    `}`
  - Now error message:
    `{`
    `"code": -2026,`
    `"msg": "Order was canceled or expired with no executed qty over 90 days ago and has been archived."`
    `}`

- Behavior for API requests with `startTime` and `endTime`:
  - Previously some requests failed if the `startTime` == `endTime`.
  - Now, all API requests that accept `startTime` and `endTime` allow the parameters to be equal. This applies to the following requests:
    - `GET /api/v3/aggTrades`
    - `GET /api/v3/klines`
    - `GET /api/v3/allOrderList`
    - `GET /api/v3/allOrders`
    - `GET /api/v3/myTrades`

- Changes to Filter Evaluation:
  - Previous behavior: `LOT_SIZE` and `MARKET_LOT_SIZE` required that (`quantity` - `minQty`) % `stepSize` == 0.
  - New behavior changed to: (`quantity` % `stepSize`) == 0.

- Bug fix with reverse `MARKET` orders (i.e. `MARKET` using `quoteOrderQty`):
  - Previous behavior: Reverse market orders would have the status `FILLED` even if the order was not fully filled.
  - New behavior: If the reverse market order did not fully fill due to low liquidity, the order status will be `EXPIRED`, and `FILLED` only if completely filled.

REST API

- Changes to `DELETE /api/v3/order` and `POST /api/v3/order/cancelReplace`:
  - New optional parameter `cancelRestrictions` that determines whether the cancel will succeed if the order status is `NEW` or `PARTIALLY_FILLED`.
  - If the order cancellation fails due to `cancelRestrictions`, error will be:
    `{`
    `"code": -2011,`
    `"msg": "Order was not canceled due to cancel restrictions."`
    `}`
- Added a new endpoint to get all orders `GET /api/v3/allOrders`

2/3/2023

Trading parameters for 153 trading pairs have been updated. Click here to learn more.

23/2/2023

- **New API Key Type (Credit Line):** Added a new API Key type (Credit Line) and instructions for generating this key type in 'Get API Keys.'
- **Added Credit Line Endpoints:** Added 'Credit Line Endpoints' section for new Credit Line API.

20/2/2023

- `Withdraw Fiat (via SIGNET)` updated to remove SEN channel.

24/1/2023

The changes to the system will take place on **January 31, 2023**.

Additional details on the functionality of STP is explained in the STP FAQ document.

Rest API

- Self-Trade Prevention (aka STP) has been added to the system. STP is a measure to prevent users from trading against their own account or other accounts that share the same `tradeGroupId` (such as parent and sub-accounts which belong to the same entity). The default and allowed modes of STP are as follows, and can be confirmed using `GET /api/v3/exchangeInfo`:
  `"defaultSelfTradePreventionMode": "EXPIRE_MAKER", //If selfTradePreventionMode not provided, this will be the value passed to the engine`
  `"allowedSelfTradePreventionModes": [//What the allowed modes of`
  `selfTradePrevention are`
  `"EXPIRE_MAKER",`
  `"EXPIRE_TAKER",`
  `"EXPIRE_BOTH"`
  `]`

- New order status: `EXPIRED_IN_MATCH` - This means that the order expired due to STP being triggered.

- New endpoints:
  - `GET /api/v3/myPreventedMatches` - This queries the orders that expired due to STP being triggered.

- New optional parameter `selfTradePreventionMode` has been added to the following endpoints:
  - `POST /api/v3/order`
  - `POST /api/v3/order/oco`
  - `POST /api/v3/order/cancelReplace`

- New responses that will appear for all order placement endpoints if there was a prevented match (i.e. if an order could have matched with an order of the same account, or the accounts are in the same `tradeGroupId`):
  - `tradeGroupId` - This will only appear if account is configured to a `tradeGroupId` and if there was a prevented match.

- `preventedQuantity` - Only appears if there was a prevented match
- An array `preventedMatches` with the following fields:
  - `preventedMatchId`
  - `makerOrderId`
  - `price`
  - `takerPreventedQuantity` - This will only appear if `selfTradePreventionMode` set is `EXPIRE_TAKER` or `EXPIRE_BOTH` .
  - `makerPreventedQuantity` - This will only appear if `selfTradePreventionMode` set is `EXPIRE_MAKER` or `EXPIRE_BOTH` .

- New fields `preventedMatchId` and `preventedQuantity` that can appear in the order query endpoints if the order had expired due to STP :

  - `GET /api/v3/order`
  - `GET /api/v3/openOrders`
  - `GET /api/v3/allOrders`

- New field `tradeGroupId` will appear in the `GET /api/v3/account` response.

USER DATA STREAM

- New execution Type: TRADE_PREVENTION
- New fields for `executionReport` (These fields will only appear if the order has expired due to STP)
  - `u` - `tradeGroupId`
  - `v` - `preventedMatchId`
  - `U` - `counterOrderId`
  - `A` - `preventedQuantity`
  - `B` - `lastPreventedQuantity`

18/1/2023

- 'Withdraw Fiat (via SEN or SIGNET)' updated to support signet.
- 'Get All OTC Trade Orders' and 'Get All OCBS Trade Orders' parameters updated. 90-day limit removed from both.
- 'Get User's Spot Asset Snapshot' API removed.

30/11/2022

WEBSOCKET

- !bookTicker removed.
  - Please use Individual Book Ticker streams (@bookTicker) instead.
  - Multiple streams can be subscribed to over one connection. (E.g. wss://stream.binance.us:9443/stream?streams=btcusdt@bookTicker/bnbbtc@bookTicker)

REST API

- New error code -1135 occurs if a parameter requiring a JSON object is invalid.
- New error code -1108 occurs if a value sent to a parameter is too large.
- Changes to GET /api/v3/aggTrades
  - startTime and endTime can now be used individually and the 1-hour limit has been removed.
- Changes to GET /api/v3/myTrades
  - Bug fixed: The combination of symbol + orderId no longer returns all trades beyond the 500 default limit.
  - Sending an unsupported combination of optional parameters now responds with generic error: { "code": -1128, "msg": "Combination of optional parameters invalid." }.
- defaultSelfTradePreventionMode and allowedSelfTradePreventionModes fields will appear in GET /api/v3/exchangeInfo
- selfTradePreventionMode field will appear in the response for several order endpoints.
- requireSelfTradePrevention field will appear in the response for GET /api/v3/account
- workingTime field indicating when the order started working on the order book, will appear in several order endpoints.
- trailingTime field will appear in order types: (TAKE_PROFIT, TAKE_PROFIT_LIMIT, STOP_LOSS, STOP_LOSS_LIMIT if trailingDelta parameter was provided), for several order endpoints.
- commissionRates field will appear in the GET /api/v3/acccount response.

USER DATA STREAM

- eventType executionReport has new fields:
  - V - selfTradePreventionMode
  - D - trailing_time (Appears if the trailing stop order is active)
  - W - workingTime (Appears if the order is working on the order book)

16/11/2022

- **Staking Endpoints:** Added four new staking-focused endpoints, including:
  - **Stake Asset:** Initiate staking for a supported asset.
  - **Unstake Asset:** Unstake an asset that is currently staking.
  - **Get Staking Asset Information:** Information on staking asset(s) including reward asset received, APR, APY, unstaking period (hrs.), minimum and maximum staking amounts, and whether auto restaking is enabled.
  - **Get Staking History:** History of staking transactions for an asset in a given time period, including transaction amount, type, and initiation time.
- **Revised Error Messages:** Updated several API error messages related to staking for increased clarity.

15/11/2022

- **New API Key Type (Custodial Solution):** Added a new API Key type (Custodial Solution) and instructions for generating this key type in 'Get API Keys.'
- **Added Custodial Solution API & Endpoints:** Added 'Custodial Solution Endpoints' section for new Custodial Solution API. Contains four endpoint categories and 18 endpoints in total:
  - **User Account Data (Custodial):** Two endpoints.
  - **Transfer (Custodial):** Five endpoints.
  - **Trade Order (Custodial):** Nine endpoints.
  - **Settlement (Custodial):** Two endpoints.

11/4/2022

- **Enabled trailing stop order:** Updated three endpoints: POST /api/v3/order, POST /api/v3/order/test, POST /api/v3/order/oco, and added one new filter: TRAILING_DELTA, to support trailing stop orders. This type of stop order activates based on the percentage of a price change in the market using the new parameter: trailingDelta. This can be used with STOP_LOSS_LIMIT, or TAKE_PROFIT_LIMIT.
- **Enabled replace order:** Added one endpoint to cancel an existing order and place a new order with the same symbol.
- **Scheduled changes:** The All Book Tickers stream (!bookTicker) is set to be removed in late November 2022. Please use the Individual Book Ticker Streams instead. (<symbol>@bookTicker). Multiple <symbol>@bookTicker streams can be subscribed to over one connection. For example: wss://stream.binance.us:9443/stream?streams=btcusdt@bookTicker/bnbbtc@bookTicker
- **Market Data:** Added a new optional parameter type in two endpoints: GET /api/v3/ticker and GET /api/v3/ticker/24hr. Also removed Individual Symbol Ticker Streams as it duplicates with Ticker Order Book Stream, supported new candlestick chart interval: 1s.
- **Get Asset Distribution History endpoint:** Updated this endpoint to also query the rebate distribution record.
- **Get Exchange Information endpoint:** Added a service line permission parameter to display all symbols with the permission matching the value provided.

9/20/2022

- Updated trade fee API to reflect the manual fee adjustment.

8/12/2022

- **User data endpoints:** Added two endpoints to get trading fees and trading volume for the past 30 days.
- **Convert dust to BNB endpoints:** Added three endpoints to convert dust to BNB and query the conversion history and convertible assets.
- **Staking endpoints:** Added two endpoints to get staking balance and staking reward history.
- **Market data endpoints:** Added one endpoint to get price change data within a requested time window and updated four endpoints to support more parameter options.
- **Market data streams:** Added two ticker streams with 1h and 4h windows, individual symbol ticker streams, and all market ticker streams.
- **Filters:** Added three filters (percent price by side, notional, exchange maximum number iceberg orders) and updated the rules of price filters.
- **Usability improvements:** Fixed some language errors and improved the content.

6/15/2022

- **User data endpoints:** Migrated seven endpoints from WAPI to SAPI to improve performance and added five endpoints for status query and crypto withdrawals.
- **Wallet endpoints:** Added two endpoints to get sub-account deposit addresses and history.
- **OTC endpoints:** Added one OTC endpoint for users to query all OTC order details.
- **Referral endpoints:** Added one referral endpoint for users to get referral rewards history.
- **Usability improvements:** Made several improvements to content and readability.
- **Miscellaneous updates:** Temporarily removed the Get User Maker/Taker Rates endpoint. The endpoint will return in a future update.

3/3/2022

- Updated five wallet endpoints related to crypto withdrawals and deposits.
- Added a new trade order endpoint for querying rate limit usage.

- Removed convert dust to BNB endpoints.
- Improved content readability and descriptions.

1/21/2022

- Added new OTC trade endpoints which support larger buy and sell order quotes, placements, and queries, as well as crypto-to-crypto conversion (e.g. KSHIB/SHIB).

1/22/2021

- Published new API documentation interface and added Python code samples.

Support

GET API SUPPORT

Have questions about our APIs? Contact customer support here.

REST API

General REST API Information

The base endpoint is: **https://api.binance.us**

All endpoints return either a JSON object or array.

Data is returned in **ascending** order: oldest first, newest last.

All times for the fields of staking, referrals, airdrops, etc. are in **milliseconds**.

Making Requests

- 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 is sent in both the `query string` and `request body`, the `query string` parameter will be used.

Data Sources(REST)

- The API system is asynchronous, so some delay in the response is normal.
- Each endpoint has a data source indicating where the data is being retrieved, and thus which endpoints have the most up-to-date response.

These are the three sources ordered from the most up-to-date response to the one with potential delays in updates:

- **Matching Engine** - the data is from the matching engine
- **Memory** - the data is from a server's local or external memory
- **Database** - the data is taken directly from a database

Some endpoints can have more than one data source(e.g. Memory => Database). This means that the endpoint will check the first data source. If it cannot find the value it's looking for it will check the next one etc.

API Terminology

These terms will be used throughout the documentation, so it is recommended that you read them to enhance your understanding of the API (especially for new users).

- `Base asset` refers to the asset that is the `quantity` of a symbol; for the symbol BTCUSDT, BTC would be the `base asset.`
- `Quote asset` refers to the asset that is the `price` of a symbol; for the symbol BTCUSDT, USDT would be the `quote asset`.

Enum Definitions(REST)

Symbol status (status):

- `PRE_TRADING`
- `TRADING`
- `POST_TRADING`
- `END_OF_DAY`
- `HALT`
- `AUCTION_MATCH`
- `BREAK`

Symbol type:

- `SPOT`

Order status (status):

Status	Description
`NEW`	The order has been accepted by the engine
`PARTIALLY_FILLED`	Part of the order has been filled

Status	Description
`FILLED`	The order has been completed
`CANCELED`	The order has been canceled by the user
`PENDING_CANCEL`	This is currently unused
`REJECTED`	The order was not accepted by the engine and not processed
`EXPIRED`	The order was canceled according to the order type's rules (e.g., LIMIT FOK orders with no fill, LIMIT IOC, or MARKET orders that partially fill), or by the exchange(e.g., orders canceled during liquidation or orders canceled during maintenance)
`EXPIRED_IN_MATCH`	The order was canceled by the exchange due to STP. (e.g. an order with EXPIRE_TAKER will match with existing orders on the book with the same account or same tradeGroupId)

OCO Status (listStatusType):

Status	Description
`RESPONSE`	This is used when the ListStatus is responding to a failed action (e.g., Orderlist placement or cancelation)
`EXEC_STARTED`	The order list has been placed, or there is an update to the order list status
`ALL_DONE`	The order list has finished executing and is thus no longer active

OCO Order Status (listOrderStatus):

Status	Description
`EXECUTING`	Either an order list has been placed, or there is an update to the status of the list
`ALL_DONE`	An order list has completed execution and is thus no longer active
`REJECT`	The List Status is responding to a failed action during order placement, or the order was canceled

ContingencyType

- `OCO`

Order types (orderTypes, type):

- `LIMIT`
- `MARKET`
- `STOP_LOSS_LIMIT`
- `TAKE_PROFIT_LIMIT`
- `LIMIT_MAKER`

Order side (side):

- `BUY`
- `SELL`

Time in force (timeInForce):

Status	Description
`GTC`	"Good Till Canceled." An order will be on the book unless the order is canceled
`IOC`	"Immediate or Cancel." An order will try to fill the order as much as it can before the order expires
`FOK`	"Fill or Kill." An order will expire if the full order cannot be filled upon execution

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)

> Example - REQUEST_WEIGHT

```
{
  "rateLimitType": "REQUEST_WEIGHT",
  "interval": "MINUTE",
  "intervalNum": 1,
  "limit": 1200
}
```

> Example - ORDERS

```
{
  "rateLimitType": "ORDERS",
  "interval": "SECOND",
  "intervalNum": 1,
  "limit": 10
}
```

Example - RAW_REQUESTS

```
{
  "rateLimitType": "RAW_REQUESTS",
  "interval": "MINUTE",
  "intervalNum": 5,
  "limit": 5000
}
```

- REQUEST_WEIGHT
- ORDERS
- RAW_REQUESTS

Rate limit intervals (interval)

- SECOND
- MINUTE
- DAY

Rate Limits(REST)

- The following are `intervalLetter` values for headers:
  - SECOND = S
  - MINUTE = M
  - HOUR = H
  - DAY = D
- The `/api/v3/exchangeInfo` `rateLimits` array contains objects related to the exchange's `RAW_REQUEST`, `REQUEST_WEIGHT`, and `ORDER` rate limits. These are further defined in the `ENUM definitions` section under `Rate limiters (rateLimitType)`.
- A 429 will be returned when either rate limit is violated.
- Each route has a `weight` that determines the number of requests each endpoint counts for. Heavier endpoints and endpoints that do operations on multiple symbols will have a heavier `weight`.
- REST API and WebSocket API are subject to the same Rate Limit rules.

IP Limits(REST)

- Every request will contain an `X-MBX-USED-WEIGHT-(intervalNum)(intervalLetter)` header which has the currently used weight for the IP of all request rate limiters defined.
- Every successful order will contain an `X-MBX-ORDER-COUNT-(intervalNum)(intervalLetter)` header which has the current order count for the IP of all order rate limiters defined. Rejected/unsuccessful orders are not guaranteed to have a `X-MBX-ORDER-COUNT-**` header in the response.
- When a 429 is received, it's your obligation as an API user/trader 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**.
- A `Retry-After` header is sent with a 418 or 429 response and will give the **number of seconds** required to wait to prevent a ban (for a 418) or until the ban is over (for a 429).
- **The limits on the API are based on the IPs, not the API keys.**

Order Rate Limits(REST)

- Every successful order response will contain an `X-MBX-ORDER-COUNT-(intervalNum)(intervalLetter)` header which has the current order count for the account for all order rate limiters defined.
- Rejected/unsuccessful orders are not guaranteed to have a `X-MBX-ORDER-COUNT-**` header in the response.
- **The order rate limit is counted against each account**.

General Data Endpoints

System Information

TEST CONNECTIVITY

Example

```
curl 'https://api.binance.us/api/v3/ping'
```

Response

```
{}
```

`GET /api/v3/ping`

Use this endpoint to test connectivity to the exchange.

Weight: 1

Parameters: NONE

Data Source: Memory

GET SERVER TIME

Example

```
curl 'https://api.binance.us/api/v3/time'
```

Response

```
{
  "serverTime": 1499827319559
}
```

`GET /api/v3/time`

Use this endpoint to get the exchange's server time.

Weight: 1

Parameters: NONE

Data Source: Memory

GET SYSTEM STATUS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/system/status?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "status": 0 // 0: normal, 1: system maintenance
}
```

`GET /sapi/v1/system/status (HMAC SHA256)`

Use this endpoint to fetch whether the system status is normal or under maintenance.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
timestamp	LONG	YES	

Data Source: Memory

Exchange Information

GET EXCHANGE INFORMATION

Example

```
curl 'https://api.binance.us/api/v3/exchangeInfo'
```

Response

```
{
  "timezone": "UTC",
  "serverTime": 1565246363776,
  "rateLimits": [
    {
      //These are defined in the `ENUM definitions` section under `Rate Limiters (rateLimitType)`.
      //All limits are optional
    }
  ],
  "exchangeFilters": [
    //These are the defined filters in the `Filters` section.
    //All filters are optional.
  ],
  "symbols": [
    {
      "symbol": "ETHBTC",
      "status": "TRADING",
      "baseAsset": "ETH",
      "baseAssetPrecision": 8,
      "quoteAsset": "BTC",
      "quotePrecision": 8,
      "quoteAssetPrecision": 8,
      "baseCommissionPrecision": 8,
      "quoteCommissionPrecision": 8,
      "orderTypes": [
        "LIMIT",
        "LIMIT_MAKER",
        "MARKET",
        "STOP_LOSS",
        "STOP_LOSS_LIMIT",
        "TAKE_PROFIT",
        "TAKE_PROFIT_LIMIT"
      ],
      "icebergAllowed": true,
      "ocoAllowed": true,
      "quoteOrderQtyMarketAllowed": true,
      "allowTrailingStop": false
      "cancelReplaceAllowed": true,
      "isSpotTradingAllowed": true,
      "isMarginTradingAllowed": false,
      "filters": [
        //These are defined in the Filters section.
        //All filters are optional
      ]
    }
  ],
  "permissions": [
     "SPOT"
  ],
```

```
  "defaultSelfTradePreventionMode": "EXPIRE_MAKER", //If selfTradePreventionMode not provided, this will be the value passed to the engine
  "allowedSelfTradePreventionModes": [ //What the allowed modes of selfTradePrevention are
    "EXPIRE_MAKER",
    "EXPIRE_TAKER",
    "EXPIRE_BOTH"
    ]
}
```

`GET /api/v3/exchangeInfo`

Use this endpoint to get the current exchange trading rules and trading pair information.

Weight: 10

Parameters:

There are 4 possible options:

Options	Example
No parameter	curl -X GET "https://api.binance.us/api/v3/exchangeInfo"
symbol	curl -X GET "https://api.binance.us/api/v3/exchangeInfo?symbol=BNBBTC"
symbols	curl -X GET curl -X GET "https://api.binance.us/api/v3/exchangeInfo?symbols=%5B%22BNBBTC%22,%22BTCUSDT%22%5D" or curl -g GET 'https://api.binance.us/api/v3/exchangeInfo?symbols=["BTCUSDT","BNBBTC"]'
permissions	curl -X GET "https://api.binance.us/api/v3/exchangeInfo?permissions=SPOT"

Notes:

- If the value provided to `symbol` or `symbols` do not exist, the endpoint will throw an error saying the symbol is invalid.
- All parameters are optional.
- If `permissions` parameter not provided, the default values will be `["SPOT"]`.

Data Source: Memory

Market Data Endpoints

Trade Data

GET RECENT TRADES

Example

```
curl -X "GET" "https://api.binance.us/api/v3/trades?symbol=LTCBTC"
```

Response

```
[
  {
    "id": 981492,
    "price": "0.00380100",
    "qty": "0.22000000",
    "quoteQty": "0.00083622",
    "time": 1637128016269,
    "isBuyerMaker": false,
    "isBestMatch": true
  },
]
```

`GET /api/v3/trades`

Use this endpoint to get the recent trades. Please note the maximum limit is 1,000 trades.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
limit	INT	NO	Default 500; max 1000

Data Source: Memory

GET HISTORICAL TRADES (MARKET_DATA)

Example

```
curl -X "GET" "https://api.binance.us/api/v3/historicalTrades?symbol=<symbol>" \
    -H "X-MBX-APIKEY: <your_api_key>"
```

Response

```
[
  {
    "id": 17,
    "price": "0.06800000",
    "qty": "1.00000000",
    "quoteQty": "0.06800000",
    "time": 1635489737109,
    "isBuyerMaker": false,
    "isBestMatch": true
  }
]
```

`GET /api/v3/historicalTrades`

Use this endpoint to get older trades. Please note the maximum limit is 1,000 trades.

Weight: 5

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
limit	INT	NO	Default 500; max 1000
fromId	LONG	NO	TradeId to fetch from. Default gets most recent trades

Data Source: Database

GET AGGREGATE TRADES

Example

```
curl -X "GET" "https://api.binance.us/api/v3/aggTrades?symbol=LTCBTC"
```

Response


```
[
  {
    "a": 874844,
    "p": "0.00379700",
    "q": "0.05000000",
    "f": 981493,
    "l": 981493,
    "T": 1637128220041,
    "m": true,
    "M": true
  }
]
```


`GET /api/v3/aggTrades`

Use this endpoint to get compressed, aggregate trades. Trades that fill at the same time, from the same order, with the same price, will have the quantity aggregated. Please note the maximum limit is 1,000 trades.

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 1000

- If fromId, startTime, and endTime are not sent, the most recent aggregate trades will be returned.

Data Source: Database

GET ORDER BOOK DEPTH

Example

```
curl -X "GET" "https://api.binance.us/api/v3/depth?symbol=LTCBTC"
```

Response


```
{
  "lastUpdateId": 1027024,
  "bids": [
    [
      "0.00379200",
      "31.26000000"
    ]
  ],
  "asks": [
    [
      "0.00380100",
      "32.37000000"
    ]
  ]
}
```


`GET /api/v3/depth`

Use this endpoint to get order book depth (prices and quantities of bids and asks).

Weight(IP): Adjusted based on the limit:

Limit	Weight
1-100	1
101-500	5
501-1000	10
1001-5000	50

Parameters:

Name	Type	Mandatory	Description

Name	Type	Mandatory	Description
symbol	STRING	YES	
limit	INT	NO	Default 100; max 5000. If limit > 5000, the response will truncate to 5000

Data Source: Memory

GET CANDLESTICK DATA

Example

```
curl -X "GET" "https://api.binance.us/api/v3/klines?symbol=LTCBTC&interval=1m"
```

Response

```
[
  [
    1499040000000,      // Open time
    "0.00386200",       // Open
    "0.00386200",       // High
    "0.00386200",       // Low
    "0.00386200",       // Close
    "0.47000000",  // Volume
    1499644799999,      // Close time
    "0.00181514",    // Quote asset volume
    1,                // Number of trades
    "0.47000000",    // Taker buy base asset volume
    "0.00181514",      // Taker buy quote asset volume
    "0" // Ignore.
  ]
]
```

`GET /api/v3/klines`

Use this endpoint to get Kline/candlestick bars for a token symbol. Klines are uniquely identified by their open time. Please note the maximum limit is 1,000 bars.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
interval	ENUM	YES	
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	Default 500; max 1000

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

Data Source: Database

Price Data

GET LIVE TICKER PRICE

Example

```
# Example A, symbol param provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/price?symbol=LTCBTC"

# Example B, no symbol provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/price"
```

Response A

```
{
  "symbol": "LTCBTC",
  "price": "0.00378800"
}
```

Response B

```
[
  {
    "symbol": "BTCUSD",
    "price": "59705.0700"
  },
  {
    "symbol": "ETHUSD",
    "price": "4178.7200"
  }
]
```

`GET /api/v3/ticker/price`

Use this endpoint to get the live ticker price.

Weight(IP):

Parameter	Symbols Provided	Weight
symbol	1	1
	symbol parameter is omitted	2

Parameter	Symbols Provided	Weight
symbols	Any	2

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	NO	Parameter symbol and symbols cannot be used in combination. If neither parameter is sent, prices for all symbols will be returned in an array
symbols	STRING	NO	Exaples of accepted format for the symbols parameter: ["BTCUSDT", "BNBUSDT"] or %5B%22BTCUSDT%22, %22BNBUSDT%22%5D

Data Source: Memory

GET AVERAGE PRICE

Example

```
curl -X "GET" "https://api.binance.us/api/v3/avgPrice?symbol=LTCBTC"
```

Response

```
{
  "mins": 5,
  "price": "0.00378906"
}
```

`GET /api/v3/avgPrice`

Use this endpoint to get the current average price for a symbol.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	

Data Source: Memory

GET BEST ORDER BOOK PRICE

Example

```
# Example A, symbol param provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/bookTicker?symbol=LTCBTC"

# Example B, no symbol provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/bookTicker"
```

Response A

```
{
  "symbol": "LTCBTC",
  "bidPrice": "0.00378600",
  "bidQty": "3.50000000",
  "askPrice": "0.00379100",
  "askQty": "26.69000000"
}
```

Response B

```
[
  {
    "symbol": "BTCUSD",
    "bidPrice": "59614.7400",
    "bidQty": "0.07100000",
    "askPrice": "59630.2500",
    "askQty": "0.07100000"
  },
  {
    "symbol": "ETHUSD",
    "bidPrice": "4171.2800",
    "bidQty": "0.72000000",
    "askPrice": "4172.3700",
    "askQty": "5.40000000"
  },
]
```

`GET /api/v3/ticker/bookTicker`

Use this endpoint to get the best available order book price.

Weight(IP):

Parameter	Symbols Provided	Weight
symbol	1	1
	symbol parameter is omitted	2
symbols	Any	2

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	NO	Parameter symbol and symbols cannot be used in combination. If neither parameter is sent, bookTickers for all symbols will be returned in an array
symbols	STRING	NO	Exaples of accepted format for the symbols parameter: ["BTCUSDT", "BNBUSDT"] or %5B%22BTCUSDT%22, %22BNBUSDT%22%5D

Data Source: Memory

Example

```
# Example A, symbol param provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/24hr?symbol=BNBBTC"

# Example B, no symbol provided
curl -X "GET" "https://api.binance.us/api/v3/ticker/24hr"
```

Response A


```
{
  "symbol": "BNBBTC",
  "priceChange": "-0.00046450",
  "priceChangePercent": "-4.659",
  "weightedAvgPrice": "0.00978390",
  "prevClosePrice": "0.00997050",
  "lastPrice": "0.00950520",
  "lastQty": "0.09000000",
  "bidPrice": "0.00950060",
  "bidQty": "0.25000000",
  "askPrice": "0.00950520",
  "askQty": "13.74000000",
  "openPrice": "0.00996970",
  "highPrice": "0.01012120",
  "lowPrice": "0.00950500",
  "volume": "7936.25000000",
  "quoteVolume": "77.64747556",
  "openTime": 1637042984994,
  "closeTime": 1637129384994,
  "firstId": 1851686,
  "lastId": 1856703,
  "count": 5018
}
```


Response B


```
[
  {
    "symbol": "BTCUSD",
    "priceChange": "-1267.1100",
    "priceChangePercent": "-2.083",
    "weightedAvgPrice": "60099.2142",
    "prevClosePrice": "60842.0600",
    "lastPrice": "59566.4300",
    "lastQty": "0.03454900",
    "bidPrice": "59525.6600",
    "bidQty": "0.07100000",
    "askPrice": "59543.0900",
    "askQty": "0.42600000",
    "openPrice": "60833.5400",
    "highPrice": "61406.0700",
    "lowPrice": "58585.4900",
    "volume": "1675.88065900",
    "quoteVolume": "100719110.6761",
    "openTime": 1637043019764,
    "closeTime": 1637129419764,
    "firstId": 25821891,
    "lastId": 25894571,
    "count": 72681
  },
  {
    "symbol": "ETHUSD",
    "priceChange": "-160.0800",
    "priceChangePercent": "-3.701",
    "weightedAvgPrice": "4233.5077",
    "prevClosePrice": "4326.0600",
    "lastPrice": "4165.8000",
    "lastQty": "2.42310000",
    "bidPrice": "4165.2100",
    "bidQty": "0.00484000",
    "askPrice": "4165.4100",
    "askQty": "6.12000000",
    "openPrice": "4325.8800",
    "highPrice": "4349.1400",
    "lowPrice": "4065.6400",
    "volume": "20975.27292000",
    "quoteVolume": "88798979.5292",
    "openTime": 1637043020441,
    "closeTime": 1637129420441,
    "firstId": 23606820,
    "lastId": 23673128,
    "count": 66309
  },
]
```


`GET /api/v3/ticker/24hr`

Use this endpoint to get price change data for the past 24hrs.

Weight(IP):

Parameter	Symbols Provided	Weight
symbol	1	1
	symbol parameter is omitted	40
symbols	1-20	1
	21-100	20
	101 or more	40
	symbols parameter is omitted	40

Parameters:

Name	Type	Mandatory	Description

Name	Type	Mandatory	Description
symbol	STRING	NO	Parameter symbol and symbols cannot be used in combination. If neither parameter is sent, tickers for all symbols will be returned in an array
symbols	STRING	NO	Exaples of accepted format for the symbols parameter: ["BTCUSDT", "BNBUSDT"] or %5B%22BTCUSDT%22, %22BNBUSDT%22%5D
type	ENUM	NO	Supported values: `FULL` or `MINI`. If none provided,the default is `FULL`. `FULL` is the default value and the response that is currently being returned from the endpoint. `MINI` omits the following fields from the response: `priceChangePercent`, `weightedAvgPrice`,`bidPrice`, `bidQty`, `askPrice`, `askQty`, and `lastQty`

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

Data Source: Memory

GET ROLLING WINDOW PRICE CHANGE STATISTICS

Example

```
# Example A, symbol param provided
curl -X "GET" "https://api.binance.us/api/v3/ticker?symbol=BNBBTC"

# Example A, symbols param provided
curl -X "GET" "https://api.binance.us/api/v3/ticker?symbols=%5B%22BTCUSDT%22,%22BNBBTC%22%5D"
```

Response A

```
{
  "symbol":             "BNBBTC",
  "priceChange":        "-8.00000000",  // Absolute price change
  "priceChangePercent": "-88.889",      // Relative price change in percent
  "weightedAvgPrice":   "2.60427807",   // QuoteVolume / Volume
  "openPrice":          "9.00000000",
  "highPrice":          "9.00000000",
  "lowPrice":           "1.00000000",
  "lastPrice":          "1.00000000",
  "volume":             "187.00000000",
  "quoteVolume":        "487.00000000", // Sum of (price * volume) for all trades
  "openTime":           1641859200000,  // Open time for ticker window
  "closeTime":          1642031999999,  // Current Time of the Request
  "firstId":            0,              // Trade IDs
  "lastId":             60,
  "count":              61              // Number of trades in the interval
}
```

Response B

```
[
  {
    "symbol": "BTCUSDT",
    "priceChange": "-154.13000000",
    "priceChangePercent": "-0.740",
    "weightedAvgPrice": "20677.46305250",
    "openPrice": "20825.27000000",
    "highPrice": "20972.46000000",
    "lowPrice": "20327.92000000",
    "lastPrice": "20671.14000000",
    "volume": "72.65112300",
    "quoteVolume": "1502240.91155513",
    "openTime": 1655432400000,
    "closeTime": 1655446835460,
    "firstId": 11147809,
    "lastId": 11149775,
    "count": 1967
  },
  {
    "symbol": "BNBBTC",
    "priceChange": "0.00008530",
    "priceChangePercent": "0.823",
    "weightedAvgPrice": "0.01043129",
    "openPrice": "0.01036170",
    "highPrice": "0.01049850",
    "lowPrice": "0.01033870",
    "lastPrice": "0.01044700",
    "volume": "166.67000000",
    "quoteVolume": "1.73858301",
    "openTime": 1655432400000,
    "closeTime": 1655446835460,
    "firstId": 2351674,
    "lastId": 2352034,
    "count": 361
  }
]
```

`GET /api/v3/ticker`

Use this endpoint to get the price change data within a requested window of time.

Note: `openTime` reverts to the start of the minute (e.g. 09:17:00 UTC, instead of 09:17:47:99). `closeTime` is the current time of the request (including seconds and milliseconds). Therefore, the effective window can be up to 59999ms (59 seconds) longer than the specified `windowSize`.

E.g. If the `closeTime` is 1641287867099 (January 04, 2022 09:17:47:099 UTC), and the `windowSize` is 1d. the `openTime` will be: 1641201420000 (January 3, 2022, 09:17:00 UTC).

Weight:

2 for each requested symbol regardless of `windowSize`.

The weight for this request will cap at 100 once the number of `symbols` in the request is more than 50.

Parameters:

Name	Type	Mandatory	Description
symbol symbols	STRING	YES	Either `symbol` or `symbols` must be provided Examples of accepted format for the `symbols` parameter: ["BTCUSDT","BNBUSDT"] or %5B%22BTCUSDT%22,%22BNBUSDT%22%5D The maximum number of `symbols` allowed in a request is 100

Name	Type	Mandatory	Description
windowSize	ENUM	NO	Defaults to `1d` if no parameter provided Supported `windowSize` values: 1m, 2m ... 59m for minutes 1h, 2h ... 23h - for hours 1d ... 7d - for days Units cannot be combined (e.g. `1d2h` is not allowed)
type	ENUM	NO	Supported values: `FULL` or `MINI`. If none provided,the default is `FULL`. `FULL` is the default value and the response that is currently being returned from the endpoint. `MINI` omits the following fields from the response: `priceChangePercent`, `weightedAvgPrice`,`bidPrice`, `bidQty`, `askPrice`, `askQty`, and `lastQty`

Data Source: Database

User Data Endpoints

User Account Data

GET USER ACCOUNT INFORMATION (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/account?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "makerCommission":15,
    "takerCommission":15,
    "buyerCommission":0,
    "sellerCommission":0,
    "commissionRates":{
        "maker":"0.00150000",
        "taker":"0.00150000",
        "buyer":"0.00000000",
        "seller":"0.00000000"
    },
    "canTrade":true,
    "canWithdraw":true,
    "canDeposit":true,
    "brokered":false,
    "requireSelfTradePrevention":false,
    "updateTime":123456789,
    "accountType":"SPOT",
    "balances":[
        {
            "asset":"BTC",
            "free":"4723846.89208129",
            "locked":"0.00000000"
        },
        {
            "asset":"LTC",
            "free":"4763368.68006011",
            "locked":"0.00000000"
        }
    ],
    "permissions":[
        "SPOT"
    ]
}
```

`GET /api/v3/account (HMAC SHA256)`

Use this endpoint to get current account information.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

GET USER ACCOUNT STATUS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"
```

```
signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v3/accountStatus?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
// Response A
{
    "msg": "Order failed:Low Order fill rate! Will be reactivated after 5 minutes.",
    "success": true,
    "objs": [
        "5"
    ]
}

// Response B
{
    "msg": "Normal",
    "success": true
}
```

`Get /sapi/v3/accountStatus (HMAC SHA256)`

Use this endpoint to fetch account status details.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
timestamp	LONG	YES	

Data Source: Database

GET USER API TRADING STATUS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v3/apiTradingStatus?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "success": true,     // Query result
    "status": {          // API trading status detail
        "isLocked": false,   // API trading function is locked or not
        "plannedRecoverTime": 0,  // If API trading function is locked, this is the planned recover time
        "triggerCondition": {
            "GCR": 150,  // Number of GTC orders
            "IFER": 150, // Number of FOK/IOC orders
            "UFR": 300   // Number of orders
        },
        "indicators": {  // The indicators updated every 30 seconds
           "BTCUSDT": [  // The symbol
            {
            "i": "UFR",  // Unfilled Ratio (UFR)
            "c": 20,     // Count of all orders
            "v": 0.05,   // Current UFR value
            "t": 0.995   // Trigger UFR value
            },
            {
            "i": "IFER", // IOC/FOK Expiration Ratio (IFER)
            "c": 20,     // Count of FOK/IOC orders
            "v": 0.99,   // Current IFER value
            "t": 0.99    // Trigger IFER value
            },
            {
            "i": "GCR",  // GTC Cancellation Ratio (GCR)
            "c": 20,     // Count of GTC orders
            "v": 0.99,   // Current GCR value
            "t": 0.99    // Trigger GCR value
            }
            ],
            "ETHUSDT": [
            {
            "i": "UFR",
            "c": 20,
            "v": 0.05,
            "t": 0.995
            },
            {
            "i": "IFER",
            "c": 20,
            "v": 0.99,
            "t": 0.99
            },
            {
            "i": "GCR",
            "c": 20,
            "v": 0.99,
            "t": 0.99
            }
            ]
        },
        "updateTime": 1547630471725   // The query result return time
    }
}
```

`GET /sapi/v3/apiTradingStatus (HMAC SHA256)`

Use this endpoint to fetch account API trading status details.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Database

GET ASSET DISTRIBUTION HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/asset/assetDistributionHistory?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
   "rows":[
      {
         "amount": "10.00000000",
         "asset": "BHFT",
         "divTime": 1563189166000,
         "category": "BHFT distribution",
         "tranId": 2968885920
      },
      {
         "amount": "10.00000000",
         "asset": "BHFT",
         "divTime": 1563189165000,
         "category": "BHFT distribution",
         "tranId": 2968885920
      }
   ],
   "total": 2
}
```

`GET /sapi/v1/asset/assetDistributionHistory (HMAC SHA256)`

Use this endpoint to query asset distribution records, including Market Maker Rebate, MM Streaks Rebate, API Partner Rebate and airdrop, etc.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
asset	STRING	NO	Distribution asset
category	STRING	NO	Distribution category (e.g., Market Maker Rebate, MM Streaks Rebate, API Partner Rebate, airdrop)
startTime	LONG	NO	Distribution start time
endTime	LONG	NO	Distribution end time
limit	INT	NO	Limit rows (default: 20, max: 500)
timestamp	LONG	YES	Current timestamp

Data Source: database

GET TRADE FEE

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X GET "$api_url/sapi/v1/asset/query/trading-fee?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
    {
        "symbol": "1INCHUSD",
        "makerCommission": "0.004",
        "takerCommission": "0.006"
    },
    {
        "symbol": "1INCHUSDT",
        "makerCommission": "0.004",
        "takerCommission": "0.006"
     }
]
```


`GET /sapi/v1/asset/query/trading-fee (HMAC SHA256)`

Use this endpoint to get your current maker & taker fee rates for spot trading based on your VIP level or manual fee adjustment. Discount for using BNB to pay fees (25% off) is not factored in.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	NO, if not specified, will return all	Symbol name

Data Source: Database

GET PAST 30 DAYS TRADE VOLUME

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X GET "$api_url/sapi/v1/asset/query/trading-volume?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "past30DaysTradingVolume": 100
}
```


`GET /sapi/v1/asset/query/trading-volume (HMAC SHA256)`

Use this endpoint to get total trade volume for the past 30 days, calculated on a rolling basis every day at 0:00 AM (UTC).

Weight: 1

Parameters: NONE

Data Source: Database

Sub-account Data

GET SUB-ACCOUNT INFORMATION

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v3/sub-account/list?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "success": true,
    "subAccounts": [
        {
            "email": "123@test.com",
            "status": "enabled",
            "activated": true,
            "mobile": "91605290",
            "gAuth": true,
            "createTime": 1544433328000
        },
        {
            "email": "321@test.com",
            "status": "disabled",
            "activated": true,
            "mobile": "22501238",
            "gAuth": true,
            "createTime": 1544433328000
        }
    ]
}
```


`GET /sapi/v3/sub-account/list (HMAC SHA256)`

Use this endpoint to get your sub-account list.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	NO	Sub-account email
status	STRING	NO	Sub-account status: enabled or disabled
page	INT	NO	Default value: 1
limit	INT	NO	Default value: 500
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Database

GET SUB-ACCOUNT TRANSFER HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v3/sub-account/transfer/history?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "msg": "string",
  "success": true,
  "transfers": [
    {
      "asset": "BNB",
      "from": "aa.email.com",
      "qty": "10",
      "time": 1637789299,
      "to": "bb.email.com"
    }
  ]
}
```

`GET /sapi/v3/sub-account/transfer/history (HMAC SHA256)`

Use this endpoint to fetch sub-account asset transfer history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	NO	Sub-account email
startTime	LONG	NO	
endTime	LONG	NO	
page	INT	NO	The transfer history batch number (each batch contains at most 500 transfer history records)
limit	INT	NO	Default value: 500
timestamp	LONG	YES	

Data Source Database

EXECUTE SUB-ACCOUNT TRANSFER

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

fromEmail=<your_from_email>
toEmail=<your_to_email>
asset=<asset>
amount=<amount>

api_url="https://api.binance.us"

signature=`echo -n "fromEmail=$fromEmail&toEmail=$toEmail&asset=$asset&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v3/sub-account/transfer?fromEmail=$fromEmail&toEmail=$toEmail&asset=$asset&amount=$amount&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "msg": "Success",
  "success": true,
  "txnId": "2966662589"
}
```

`POST /sapi/v3/sub-account/transfer(HMAC SHA256)`

Use this endpoint to execute an asset transfer between the master account and a sub-account.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
fromEmail	STRING	YES	Sender email
toEmail	STRING	YES	Recipient email
asset	STRING	YES	
amount	DECIMAL	YES	
timestamp	LONG	YES	

Data Source: Database

GET SUB-ACCOUNT ASSETS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

email="ios@mt.com"

signature=`echo -n "email=$email&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v3/sub-account/assets?email=$email&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "balances":
    [
        {
            "asset":"BTC", //asset
            "Free":0, //free
            "Locked":0 //locked
        },
        {
            "asset":"BNB",
            "free":98,
            "locked":0
        },
        {
            "asset":"ETH",
            "free":0,
            "locked":0
        },
        {
            "asset":"LTC",
            "free":0,
            "locked":0
        },
        {
            "asset":"USDT",
            "free":0,
            "locked":0
        }
    ],
    "success":true
}
```

`GET /sapi/v3/sub-account/assets (HMAC SHA256)`

Use this endpoint to fetch sub-account assets.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	YES	Sub-account Email
timestamp	LONG	YES	

Data Source: Database

GET MASTER ACCOUNT'S TOTAL USD VALUE

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
email=<email>
page=<page>
size=<size>

api_url="https://api.binance.us"

signature=`echo -n "email=$email&page=$page&size=$size&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/sub-account/spotSummary?email=$email&page=$page&size=$size&timestamp=$timestamp&signature=$signature" \
```

```
  -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "totalCount": 2,
    "masterAccountTotalAsset": 401,
    "spotSubUserAssetBtcVoList": [
        {
            "email": "test001@123.com",
            "totalAsset": 201
        },
        {
            "email": "test002@123.com",
            "totalAsset": 200
        }
    ]
}
```


`GET /sapi/v1/sub-account/spotSummary (HMAC SHA256)`

Use this endpoint to get the total value of assets in the master account in USD.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	NO	
page	INT	NO	
size	INT	NO	
timestamp	LONG	YES	

Data Source: Database

GET SUB-ACCOUNT STATUS LIST

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
email=<email>


api_url="https://api.binance.us"

signature=`echo -n "email=$email&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/sub-account/status?email=$email&timestamp=$timestamp&signature=$signature" \
  -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
    {
        "email": "test001@123.com",
        "insertTime": 1652944721676,
        "mobile": "XXXXXXXXXX",
        "isUserActive": true,
        "isMarginEnabled": true,
        "isSubUserEnabled": false,
        "isFutureEnabled": false
    },
    {
        "email": "test002@123.com",
        "insertTime": 1652944721676,
        "mobile": "XXXXXXXXXX",
        "isUserActive": true,
        "isMarginEnabled": false,
        "isSubUserEnabled": true,
        "isFutureEnabled": false
    }
]
```


`GET /sapi/v1/sub-account/status (HMAC SHA256)`

Use this endpoint to get a status list of sub-accounts.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	YES	
timestamp	LONG	YES	

Data Source: Database

Trade Order Endpoints

General Orders

GET ORDER RATE LIMITS (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

recvWindow=<recvWindow>

api_url="https://api.binance.us"

signature=`echo -n "recvWindow=$recvWindow&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/rateLimit/order?recvWindow=$recvWindow&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "rateLimitType": "ORDERS",
    "interval": "SECOND",
    "intervalNum": 10,
    "limit": 100,
    "count": 0
  },
  {
    "rateLimitType": "ORDERS",
    "interval": "DAY",
    "intervalNum": 1,
    "limit": 200000,
    "count": 0
  }
]
```

`GET /api/v3/rateLimit/order (HMAC SHA256)`

Get the current trade order count rate limits for all time intervals.

Weight: 20

Parameters:

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than 60000
timestamp	LONG	YES	

Data Source: Memory

CREATE NEW ORDER (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
side=<side>
type=<type>
quantity=<quantity>

api_url="https://api.binance.us"

signature=`echo -n "symbol=$symbol&side=$side&type=$type&quantity=$quantity&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/api/v3/order?symbol=$symbol&side=$side&type=$type&quantity=$quantity&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response ACK:

```
{
  "symbol": "BTCUSDT",
  "orderId": 28,
  "orderListId": -1, //Unless OCO, value will be -1
  "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
  "transactTime": 1507725176595
}
```

Response RESULT:

```
{
  "symbol": "BTCUSDT",
  "orderId": 28,
  "orderListId": -1, //Unless OCO, value will be -1
  "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
  "transactTime": 1507725176595,
  "price": "0.00000000",
  "origQty": "10.00000000",
  "executedQty": "10.00000000",
  "cummulativeQuoteQty": "10.00000000",
  "status": "FILLED",
  "timeInForce": "GTC",
  "type": "MARKET",
  "side": "SELL",
  "workingTime":1507725176595,
  "selfTradePreventionMode": "NONE"
}
```

Response FULL:

```
{
  "symbol": "BTCUSDT",
```

```
    "orderId": 28,
    "orderListId": -1, //Unless OCO, value will be -1
    "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
    "transactTime": 1507725176595,
    "price": "0.00000000",
    "origQty": "10.00000000",
    "executedQty": "10.00000000",
    "cummulativeQuoteQty": "10.00000000",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "MARKET",
    "side": "SELL",
    "workingTime":1507725176595,
    "selfTradePreventionMode": "NONE"
    "fills": [
      {
        "price": "4000.00000000",
        "qty": "1.00000000",
        "commission": "4.00000000",
        "commissionAsset": "USDT",
        "tradeId": 56
      },
      {
        "price": "3999.00000000",
        "qty": "5.00000000",
        "commission": "19.99500000",
        "commissionAsset": "USDT",
        "tradeId": 57
      },
      {
        "price": "3998.00000000",
        "qty": "2.00000000",
        "commission": "7.99600000",
        "commissionAsset": "USDT",
        "tradeId": 58
      },
      {
        "price": "3997.00000000",
        "qty": "1.00000000",
        "commission": "3.99700000",
        "commissionAsset": "USDT",
        "tradeId": 59
      },
      {
        "price": "3995.00000000",
        "qty": "1.00000000",
        "commission": "3.99500000",
        "commissionAsset": "USDT",
        "tradeId": 60
      }
    ]
  }
```

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

Use this endpoint to place a new trade order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD)
side	ENUM	YES	Order side (e.g., BUY, SELL)
type	ENUM	YES	Order type (e.g., LIMIT, MARKET, STOP_LOSS_LIMIT, TAKE_PROFIT_LIMIT, LIMIT_MAKER)
timeInForce	ENUM	NO	
quantity	DECIMAL	NO	
quoteOrderQty	DECIMAL	NO	
price	DECIMAL	NO	Order price
newClientOrderId	STRING	NO	A unique ID among open orders. Automatically generated if not sent. Orders with the same `newClientOrderID` can be accepted only when the previous one is filled, otherwise the order will be rejected. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: "ABCD1234-...".
stopPrice	DECIMAL	NO	Used with `STOP_LOSS_LIMIT`, and `TAKE_PROFIT_LIMIT` orders
trailingDelta	LONG	NO	Used with `STOP_LOSS_LIMIT`, and `TAKE_PROFIT_LIMIT` orders. For more details on `SPOT` implementation on `trailing stops`, please refer to Trailing Stop FAQ
icebergQty	DECIMAL	NO	Used with `LIMIT`, `STOP_LOSS_LIMIT`, and `TAKE_PROFIT_LIMIT` to create an iceberg order
selfTradePreventionMode	ENUM	NO	The configured default mode is `EXPIRE_MAKER`. The supported values currently are `EXPIRE_TAKER`, `EXPIRE_MAKER`, `EXPIRE_BOTH`.
newOrderRespType	ENUM	NO	Set the response JSON. `ACK`, `RESULT`, or `FULL`; `MARKET` and `LIMIT` order types default to `FULL`; all other orders default to `ACK`
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Some additional mandatory parameters based on order `type`:

Type	Additional Mandatory Parameters	Additional Information
`LIMIT`	`timeInForce`, `quantity`, `price`	
`MARKET`	`quantity` or `quoteOrderQty`	`MARKET` orders using the `quantity` field specifies the amount of the `base asset` the user wants to buy or sell at the market price. E.g., a MARKET order on BTCUSDT will specify how much BTC the user is buying or selling `MARKET` orders using `quoteOrderQty` specify the amount the user wants to spend (when buying) or receive (when selling) the `quote` asset; the correct `quantity` will be determined based on the market liquidity and `quoteOrderQty`. E.g., Using the symbol BTCUSDT: `BUY` side, the order will buy as many BTC as `quoteOrderQty` USDT can. `SELL` side, the order will sell as much BTC needed to receive `quoteOrderQty` USDT

Type	Additional Mandatory Parameters	Additional Information
`STOP_LOSS_LIMIT`	`timeInForce`, `quantity`, `price`, `stopPrice`, `trailingDelta`	This will execute a LIMIT order when the stopPrice is reached
`TAKE_PROFIT_LIMIT`	`timeInForce`, `quantity`, `price`, `stopPrice`, `trailingDelta`	This will execute a LIMIT order when the stopPrice is reached
`LIMIT_MAKER`	`quantity`, `price`	This is a `LIMIT` order that will be rejected if the order immediately matches and trades as a taker. This is also known as a POST-ONLY order

Other info:

- 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`.
- `MARKET` orders using `quoteOrderQty` will not break `LOT_SIZE` filter rules; the order will execute a `quantity` with a notional value as close as possible to `quoteOrderQty`.

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`

Data Source: Matching Engine

TEST NEW ORDER (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
side=<side>
type=<type>
quantity=<quantity>

api_url="https://api.binance.us"

signature=`echo -n "symbol=$symbol&side=$side&type=$type&quantity=$quantity&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/api/v3/order/test?symbol=$symbol&side=$side&type=$type&quantity=$quantity&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{}
```

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

Use this endpoint to test new order creation and signature/recvWindow long. The endpoint creates and validates a new order but does not send it into the matching engine.

Weight: 1

Parameters:

Same as `POST /api/v3/order`

Data Source: Memory

GET ORDER (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
orderId=<orderId>
symbol=<symbol>

api_url="https://api.binance.us"

signature=`echo -n "orderId=$orderId&symbol=$symbol&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/order?orderId=$orderId&symbol=$symbol&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "symbol": "LTCBTC",
  "orderId": 1,
  "orderListId": -1 //Unless part of an OCO, the value will always be -1.
  "clientOrderId": "myOrder1",
  "price": "0.1",
  "origQty": "1.0",
  "executedQty": "0.0",
  "cummulativeQuoteQty": "0.0",
  "status": "NEW",
  "timeInForce": "GTC",
  "type": "LIMIT",
  "side": "BUY",
  "stopPrice": "0.0",
  "icebergQty": "0.0",
  "time": 1499827319559,
  "updateTime": 1499827319559,
  "isWorking": true,
  "origQuoteOrderQty": "0.000000",
  "workingTime":1507725176595,
  "selfTradePreventionMode": "NONE"
}
```


`GET /api/v3/order (HMAC SHA256)`

Use this endpoint to check a trade order's status.

Weight: 2

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
orderId	LONG	NO	
origClientOrderId	STRING	NO	
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Notes:

- Either `orderId` or `origClientOrderId` must be sent.
- For some historical orders `cummulativeQuoteQty` will be < 0, meaning the data is not available at this time.

Data Source: Database

GET ALL OPEN ORDERS (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/openOrders?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "symbol": "LTCBTC",
    "orderId": 1,
    "orderListId": -1, //Unless OCO, the value will always be -1
    "clientOrderId": "myOrder1",
    "price": "0.1",
    "origQty": "1.0",
    "executedQty": "0.0",
    "cummulativeQuoteQty": "0.0",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "stopPrice": "0.0",
    "icebergQty": "0.0",
    "time": 1499827319559,
    "updateTime": 1499827319559,
    "isWorking": true,
    "origQuoteOrderQty": "0.000000",
    "selfTradePreventionMode": "NONE"
  }
]
```

`GET /api/v3/openOrders (HMAC SHA256)`

Use this endpoint to get all open trade orders for a token symbol. Do not access this without a token symbol as this would return all pair data.

Weight: 3 for a single symbol; 40 when the symbol parameter is omitted

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	NO	
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

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

Data Source: Database

CANCEL ORDER (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
orderId=<orderId>
symbol=<symbol>

api_url="https://api.binance.us"

signature=`echo -n "orderId=$orderId&symbol=$symbol&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "DELETE" "$api_url/api/v3/order?orderId=$orderId&symbol=$symbol&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "symbol": "LTCBTC",
  "origClientOrderId": "myOrder1",
  "orderId": 4,
  "orderListId": -1, //Unless part of an OCO, the value will always be -1.
  "clientOrderId": "cancelMyOrder1",
  "price": "2.00000000",
  "origQty": "1.00000000",
  "executedQty": "0.00000000",
  "cummulativeQuoteQty": "0.00000000",
  "status": "CANCELED",
  "timeInForce": "GTC",
  "type": "LIMIT",
  "side": "BUY",
  "selfTradePreventionMode": "NONE"
}
```

`DELETE /api/v3/order (HMAC SHA256)`

Use this endpoint to cancel an active trade 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. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: "ABCD1234-…".
cancelRestrictions	ENUM	NO	Supported values: `ONLY_NEW` - Cancel will succeed if the order status is `NEW`. `ONLY_PARTIALLY_FILLED` - Cancel will succeed if order status is `PARTIALLY_FILLED`.
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Either `orderId` or `origClientOrderId` must be sent.

Data Source: Matching Engine

REGARDING `CANCELRESTRICTIONS`

- If the `cancelRestrictions` value is not any of the supported values, the error will be: `{ "code": -1145, "msg": "Invalid cancelRestrictions" }`
- If the order did not pass the conditions for `cancelRestrictions`, the error will be: `{ "code": -2011, "msg": "Order was not canceled due to cancel restrictions." }`

CANCEL OPEN ORDERS FOR SYMBOL (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>

api_url="https://api.binance.us"

signature=`echo -n "symbol=$symbol&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "DELETE" "$api_url/api/v3/openOrders?symbol=$symbol&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "symbol": "BTCUSDT",
    "origClientOrderId": "KZJijIsAFf5BU5oxkWTAU3",
    "orderId": 0,
    "orderListId": -1,
    "clientOrderId": "8epSIUMvNCknntXcSzWs7H",
    "price": "0.10000000",
    "origQty": "1.00000000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY"
  }
]
```

`DELETE /api/v3/openOrders (HMAC SHA256)`

Use this endpoint to cancels all active trade orders on a token symbol (this includes OCO orders).

Weight: 1

Parameters

Name	Type	Mandatory	Description
symbol	STRING	YES	
side	ENUM	YES	

Name	Type	Mandatory	Description
type	ENUM	YES	
cancelReplaceMode	ENUM	YES	The allowed values are: `STOP_ON_FAILURE` - If the cancel request fails, the new order placement will not be attempted. `ALLOW_FAILURE` - new order placement will be attempted even if cancel request fails.
timeInForce	ENUM	NO	
quantity	DECIMAL	NO	
quoteOrderQty	DECIMAL	NO	
price	DECIMAL	NO	
cancelNewClientOrderId	STRING	NO	Used to uniquely identify this cancel. Automatically generated by default.
cancelOrigClientOrderId	STRING	NO	Either the `cancelOrigClientOrderId` or `cancelOrderId` must be provided. If both are provided, `cancelOrderId` takes precedence.
cancelOrderId	LONG	NO	Either the `cancelOrigClientOrderId` or `cancelOrderId` must be provided. If both are provided, `cancelOrderId` takes precedence.
newClientOrderId	STRING	NO	Used to identify the new order. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: “ABCD1234-…”.
stopPrice	DECIMAL	NO	
trailingDelta	LONG	NO	
icebergQty	DECIMAL	NO	

newOrderRespType	ENUM	NO	Allowed values:
			`ACK`, `RESULT`, `FULL`
			`MARKET` and `LIMIT` orders types default to `FULL`; all other orders default to `ACK`

selfTradePreventionMode	ENUM	NO	The allowed enums is dependent on what is configured on the symbol. The possible supported values are `EXPIRE_TAKER`, `EXPIRE_MAKER`, `EXPIRE_BOTH`, `NONE`.
cancelRestrictions	ENUM	NO	Supported values: `ONLY_NEW` - Cancel will succeed if the order status is `NEW`. `ONLY_PARTIALLY_FILLED` - Cancel will succeed if order status is `PARTIALLY_FILLED`. For more information please refer to Regarding `cancelRestrictions`
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Matching Engine

GET TRADES

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "symbol=BNBBTC&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/myTrades?symbol=BNBBTC&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "symbol": "BNBBTC",
    "id": 28457,
    "orderId": 100234,
    "orderListId": -1,
    "price": "4.00000100",
    "qty": "12.00000000",
    "quoteQty": "48.000012",
    "commission": "10.10000000",
    "commissionAsset": "BNB",
    "time": 1499865549590,
    "isBuyer": true,
    "isMaker": false,
    "isBestMatch": true
  }
]
```

`GET /api/v3/myTrades (HMAC SHA256)`

Use this endpoint to get trade data for a specific account and token symbol.

Weight: 10 with symbol

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
orderId	LONG	NO	This can only be used in combination with `symbol`
startTime	LONG	NO	
endTime	LONG	NO	
fromId	LONG	NO	TradeId to fetch from. Default gets most recent trades
limit	INT	NO	Default 500; max 1000

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Notes:

- If `fromId` is set, it will get orders >= than `fromId`. Otherwise most recent orders are returned.
- The time between `startTime` and `endTime` can't be longer than 24 hours.
- These are the supported combinations of optional parameters:
  - symbol
  - symbol + orderId
  - symbol + fromId
  - symbol + startTime
  - symbol + endTime
  - symbol + startTime + endTime
  - symbol + orderId + fromId

Data Source: Memory => Database

REPLACE ORDER (TRADE)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
side=<side>
type=<type>
cancelReplaceMode=<cancelReplaceMode>
cancelOrderId=<cancelOrderId>
timeInForce=<timeInForce>
quantity=<quantity>
price=<price>
api_url="https://api.binance.us"
parameter_concat="symbol=$symbol&side=$side&type=$type&cancelReplaceMode=$cancelReplaceMode&cancelOrderId=$cancelOrderId&timeInForce=$timeInForce&quantity=$quantity&price=$price&timestamp=$timestamp"
signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/api/v3/order/cancelReplace?$parameter_concat&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
//Both the cancel order placement and new order placement succeeded.
{
  "cancelResult": "SUCCESS",
  "newOrderResult": "SUCCESS",
  "cancelResponse": {
    "symbol": "BTCUSDT",
    "origClientOrderId": "DnLo3vTAQcjha43lAZhZ0y",
    "orderId": 9,
    "orderListId": -1,
    "clientOrderId": "osxN3JXAtJvKvCqGeMWMVR",
    "price": "0.01000000",
    "origQty": "0.000100",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "SELL",
    "selfTradePreventionMode": "NONE"
  },
  "newOrderResponse": {
    "symbol": "BTCUSDT",
    "orderId": 10,
    "orderListId": -1,
    "clientOrderId": "wOceeeOzNORyLiQfw7jd8S",
    "transactTime": 1652928801803,
    "price": "0.02000000",
    "origQty": "0.040000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "workingTime": 1652928801803,
    "fills": [],
    "selfTradePreventionMode": "NONE"
  }
}
```

`POST /api/v3/order/cancelReplace (HMAC SHA256)`

Cancels an existing order and places a new order on the same symbol.

Filters and Order Count are evaluated before the processing of the cancellation and order placement occurs.

A new order that was not attempted (i.e. when `newOrderResult: NOT_ATTEMPTED`), will still increase the order count by 1.

Weight(IP):1

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
side	ENUM	YES	
type	ENUM	YES	
cancelReplaceMode	ENUM	YES	The allowed values are: `STOP_ON_FAILURE` - If the cancel request fails, the new order placement will not be attempted. `ALLOW_FAILURE` - new order placement will be attempted even if cancel request fails.

Name	Type	Mandatory	Description
timeInForce	ENUM	NO	
quantity	DECIMAL	NO	
quoteOrderQty	DECIMAL	NO	
price	DECIMAL	NO	
cancelNewClientOrderId	STRING	NO	Used to uniquely identify this cancel. Automatically generated by default.
cancelOrigClientOrderId	STRING	NO	Either the cancelOrigClientOrderId or cancelOrderId must be provided. If both are provided, cancelOrderId takes precedence.
cancelOrderId	LONG	NO	Either the cancelOrigClientOrderId or cancelOrderId must be provided. If both are provided, cancelOrderId takes precedence.
newClientOrderId	STRING	NO	Used to identify the new order. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: "ABCD1234-…".
strategyId	INT	NO	
strategyType	INT	NO	The value cannot be less than 1000000.
stopPrice	DECIMAL	NO	
trailingDelta	LONG	NO	
icebergQty	DECIMAL	NO	
selfTradePreventionMode	ENUM	NO	The configured default mode is `EXPIRE_MAKER`. The supported values currently are `EXPIRE_TAKER`, `EXPIRE_MAKER`, `EXPIRE_BOTH`.
newOrderRespType	ENUM	NO	Allowed values:`ACK`, `RESULT`, `FULL MARKET` and `LIMIT` orders types default to `FULL`; all other orders default to `ACK`
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Notes:

Similar to POST `/api/v3/order`, additional mandatory parameters are determined by type.

Response format varies depending on whether the processing of the message succeeded, partially succeeded, or failed.

Data Source: Matching Engine

QUERY PREVENTED MATCHES (USER_DATA)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
preventedMatchId=<preventedMatchId>
api_url="https://api.binance.us"

parameter_concat="symbol=$symbol&preventedMatchId=$preventedMatchId&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "GET" "$api_url/api/v3/myPreventedMatches?$parameter_concat&signature=$signature" \
  -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
{
"symbol": "BTCUSDT",
"preventedMatchId": 1,
"takerOrderId": 5,
"makerOrderId": 3,
"tradeGroupId": 1,
"selfTradePreventionMode": "EXPIRE_MAKER",
"price": "1.100000",
"makerPreventedQuantity": "1.300000",
"transactTime": 1669101687094
}
]
```


`GET /api/v3/myPreventedMatches (HMAC SHA256)`

Displays the list of orders that were expired because of STP. These are the combinations supported:

- symbol + preventedMatchId
- symbol + orderId
- symbol + orderId + fromPreventedMatchId (limit will default to 500)
- symbol + orderId + fromPreventedMatchId + limit

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
preventedMatchId	LONG	NO	
orderId	LONG	NO	
fromPreventedMatchId	LONG	NO	
limit	INT	NO	Default: `500`; Max: `1000`
recvWindow	LONG	NO	The value cannot be greater than 60000
timestamp	LONG	YES	

Weight

Case	Weight
If symbol is invalid	1
Querying by preventedMatchId	1
Querying by orderId	10

Data Source: Database

ALL ORDERS (USER_DATA)

Example

```
#!/bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"
symbol=BTCUSD
parameter_concat="symbol=$symbol&timestamp=$timestamp"
signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/allOrders?$parameter_concat&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "symbol": "BTCUSD",
    "orderId": 221505,
    "orderListId": -1,
    "clientOrderId": "web_13fc402e44d2448c88c04ce4094fb9c6",
    "price": "0.00000000",
    "origQty": "1.00000000",
    "executedQty": "1.00000000",
    "cummulativeQuoteQty": "7155.37000000",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "MARKET",
    "side": "BUY",
    "stopPrice": "0.00000000",
    "icebergQty": "0.00000000",
    "time": 1678951342382,
    "updateTime": 1678951342382,
    "isWorking": true,
    "workingTime": 1678951342382,
    "origQuoteOrderQty": "0.00000000",
    "selfTradePreventionMode": "NONE"
  }
]
```

`GET /api/v3/allOrders (HMAC SHA256)`

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

Weight(IP): 10 with symbol

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
orderId	LONG	NO	
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	Default 500; max 1000
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Notes:

- If `orderId` is set, it will get orders >= that `orderId`. Otherwise most recent orders are returned.
- For some historical orders `cummulativeQuoteQty` will be < 0, meaning the data is not available at this time.
- If `startTime` and/or `endTime` provided, `orderId` is not required.

Data Source: Database

OCO Orders

CREATE NEW OCO ORDER (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
side=<side>
quantity=<quantity>
```

```
price=<price>
stopPrice=<stopPrice>
stopLimitPrice=<stopLimitPrice>
stopLimitTimeInForce=<stopLimitTimeInForce>

api_url="https://api.binance.us"

signature=`echo -n "symbol=$symbol&side=$side&quantity=$quantity&price=$price&stopPrice=$stopPrice&stopLimitPrice=$stopLimitPrice&stopLimitTimeInForce=$stopLimitTimeInForce&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/api/v3/order/oco?symbol=$symbol&side=$side&quantity=$quantity&price=$price&stopPrice=$stopPrice&stopLimitPrice=$stopLimitPrice&stopLimitTimeInForce=$stopLimitTimeInForce&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "orderListId": 0,
  "contingencyType": "OCO",
  "listStatusType": "EXEC_STARTED",
  "listOrderStatus": "EXECUTING",
  "listClientOrderId": "JYVpp3F0f5CAG15DhtrqLp",
  "transactionTime": 1563417480525,
  "symbol": "LTCBTC",
  "orders": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "clientOrderId": "xTXKaGYd4bluPVp78IVRvl"
    }
  ],
  "orderReports": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "orderListId": 0,
      "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos",
      "transactTime": 1563417480525,
      "price": "0.000000",
      "origQty": "0.624363",
      "executedQty": "0.000000",
      "cummulativeQuoteQty": "0.000000",
      "status": "NEW",
      "timeInForce": "GTC",
      "type": "STOP_LOSS",
      "side": "BUY",
      "stopPrice": "0.960664",
      "workingTime": -1,
      "selfTradePreventionMode": "NONE"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "orderListId": 0,
      "clientOrderId": "xTXKaGYd4bluPVp78IVRvl",
      "transactTime": 1563417480525,
      "price": "0.036435",
      "origQty": "0.624363",
      "executedQty": "0.000000",
      "cummulativeQuoteQty": "0.000000",
      "status": "NEW",
      "timeInForce": "GTC",
      "type": "LIMIT_MAKER",
      "side": "BUY",
      "workingTime": 1563417480525,
      "selfTradePreventionMode": "NONE"
    }
  ]
}
```


`POST /api/v3/order/oco (HMAC SHA256)`

Weight: 1

Use this endpoint to place a new OCO(one-cancels-the-other) order.

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
listClientOrderId	STRING	NO	A unique ID for the entire orderList
side	ENUM	YES	
quantity	DECIMAL	YES	
limitClientOrderId	STRING	NO	A unique ID for the limit order
price	DECIMAL	YES	
limitIcebergQty	DECIMAL	NO	
trailingDelta	LONG	NO	
stopClientOrderId	STRING	NO	A unique ID for the stop loss/stop loss limit leg
stopPrice	DECIMAL	YES	
stopLimitPrice	DECIMAL	NO	If provided, `stopLimitTimeInForce` is required
stopIcebergQty	DECIMAL	NO	
stopLimitTimeInForce	ENUM	NO	Valid values are `GTC`/`FOK`/`IOC`
newOrderRespType	ENUM	NO	Set the response JSON
selfTradePreventionMode	ENUM	NO	The configured default mode is `EXPIRE_MAKER`. The supported values currently are `EXPIRE_TAKER`, `EXPIRE_MAKER`, `EXPIRE_BOTH`.

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Other Info:

- Price Restrictions:
  - `SELL`: Limit Price > Last Price > Stop Price
  - `BUY`: Limit Price < Last Price < Stop Price
- Quantity Restrictions:
  - Both legs must have the same quantity.
  - `ICEBERG` quantities however do not have to be the same
- Order Rate Limit
  - `OCO` counts as 2 orders against the order rate limit.

Data Source: Matching Engine

GET OCO ORDER (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
orderListId=<orderListId>

api_url="https://api.binance.us"

signature=`echo -n "orderListId=$orderListId&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/orderList?orderListId=$orderListId&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "orderListId": 27,
  "contingencyType": "OCO",
  "listStatusType": "EXEC_STARTED",
  "listOrderStatus": "EXECUTING",
  "listClientOrderId": "h2USkA5YQpaXHPIrkd96xE",
  "transactionTime": 1565245656253,
  "symbol": "LTCBTC",
  "orders": [
    {
      "symbol": "LTCBTC",
      "orderId": 4,
      "clientOrderId": "qD1gy3kc3Gx0rihm9Y3xwS"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 5,
      "clientOrderId": "ARzZ9I00CPM8i3NhmU9Ega"
    }
  ]
}
```


`GET /api/v3/orderList (HMAC SHA256)`

Weight: 2

Use this endpoint to retrieve a specific OCO order based on provided optional parameters.

Parameters:

Name	Type	Mandatory	Description
orderListId	LONG	NO	Either `orderListId` or `listClientOrderId` must be provided
origClientOrderId	STRING	NO	Either `orderListId` or `listClientOrderId` must be provided
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

GET ALL OCO ORDER (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/allOrderList?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
  {
    "orderListId": 29,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "amEEAXryFzFwYF1FeRpUoZ",
    "transactionTime": 1565245913483,
```

```
    "symbol": "LTCBTC",
    "orders": [
      {
        "symbol": "LTCBTC",
        "orderId": 4,
        "clientOrderId": "oD7aesZqjEGlZrbtRpy5zB"
      },
      {
        "symbol": "LTCBTC",
        "orderId": 5,
        "clientOrderId": "Jr1h6xirOxgeJOUuYQS7V3"
      }
    ]
  },
  {
    "orderListId": 28,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "hG7hFNxJV6cZy3Ze4AUT4d",
    "transactionTime": 1565245913407,
    "symbol": "LTCBTC",
    "orders": [
      {
        "symbol": "LTCBTC",
        "orderId": 2,
        "clientOrderId": "j6lFOfbmFMRjTYA7rRJ0LP"
      },
      {
        "symbol": "LTCBTC",
        "orderId": 3,
        "clientOrderId": "z0KCjOdditiLS5ekAFtK81"
      }
    ]
  }
]
```

`GET /api/v3/allOrderList (HMAC SHA256)`

Weight: 10

Use this endpoint to retrieve all OCO orders based on provided optional parameters. Please note the maximum limit is 1,000 orders.

Parameters

Name	Type	Mandatory	Description
fromId	LONG	NO	If supplied, neither `startTime` nor `endTime` can be provided
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	Default Value: 500; Max Value: 1000
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

GET OPEN OCO ORDERS (USER_DATA)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/api/v3/openOrderList?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "orderListId": 31,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "wuB13fmulKj3YjdqWEcsnp",
    "transactionTime": 1565246080644,
    "symbol": "1565246079109",
    "orders": [
      {
        "symbol": "LTCBTC",
        "orderId": 4,
        "clientOrderId": "r3EH2N76dHfLoSZWIUw1bT"
      },
      {
        "symbol": "LTCBTC",
        "orderId": 5,
        "clientOrderId": "Cv1SnyPD3qhqpbjpYEHbd2"
      }
    ]
  }
]
```

`GET /api/v3/openOrderList (HMAC SHA256)`

Use this endpoint to query open OCO orders.

Weight: 3

Parameters

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

CANCEL OCO ORDER (TRADE)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
symbol=<symbol>
orderListId=<orderListId>

api_url="https://api.binance.us"

signature=`echo -n "symbol=$symbol&orderListId=$orderListId&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "DELETE" "$api_url/api/v3/orderList?symbol=$symbol&orderListId=$orderListId&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "orderListId": 0,
  "contingencyType": "OCO",
  "listStatusType": "ALL_DONE",
  "listOrderStatus": "ALL_DONE",
  "listClientOrderId": "C3wyj4WVEktd7u9aVBRXcN",
  "transactionTime": 1574040868128,
  "symbol": "LTCBTC",
  "orders": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "clientOrderId": "pO9ufTiFGg3nw2fOdgeOXa"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "clientOrderId": "TXOvglzXuaubXAaENpaRCB"
    }
  ],
  "orderReports": [
    {
      "symbol": "LTCBTC",
      "origClientOrderId": "pO9ufTiFGg3nw2fOdgeOXa",
      "orderId": 2,
      "orderListId": 0,
      "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
      "price": "1.00000000",
      "origQty": "10.00000000",
      "executedQty": "0.00000000",
      "cummulativeQuoteQty": "0.00000000",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "STOP_LOSS_LIMIT",
      "side": "SELL",
      "stopPrice": "1.00000000",
      "selfTradePreventionMode": "NONE"
    },
    {
      "symbol": "LTCBTC",
      "origClientOrderId": "TXOvglzXuaubXAaENpaRCB",
      "orderId": 3,
      "orderListId": 0,
      "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
      "price": "3.00000000",
      "origQty": "10.00000000",
      "executedQty": "0.00000000",
      "cummulativeQuoteQty": "0.00000000",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "LIMIT_MAKER",
      "side": "SELL",
      "selfTradePreventionMode": "NONE"
    }
  ]
}
```


`DELETE /api/v3/orderList (HMAC SHA256)`

Weight: 1

Use this endpoint to cancel an entire order list.

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
orderListId	LONG	NO	Either `orderListId` or `listClientOrderId` must be provided
listClientOrderId	STRING	NO	Either `orderListId` or `listClientOrderId` must be provided
newClientOrderId	STRING	NO	Used to uniquely identify this cancel. Automatically generated by default. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: "ABCD1234-…".
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Additional notes:

- Canceling an individual leg will cancel the entire OCO

Data Source: Matching Engine

OTC Endpoints

Get Supported Coin Pairs

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/otc/coinPairs?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
    {
        "fromCoin": "BTC",
        "toCoin": "USDT",
        "fromCoinMinAmount": 0.1,
        "fromCoinMaxAmount": 100,
        "toCoinMinAmount": 1000,
        "toCoinMaxAmount": 500000
    },
    {
        "fromCoin": "USDT",
        "toCoin": "BNB",
        "fromCoinMinAmount": 2000,
        "fromCoinMaxAmount": 600000,
        "toCoinMinAmount": 10,
        "toCoinMaxAmount": 4000
    }
]
```

`GET /sapi/v1/otc/coinPairs (HMAC SHA256)`

Use this endpoint to get a list of supported coin pairs.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
fromCoin	STRING	NO	From coin name, e.g. BTC, SHIB
toCoin	STRING	NO	To coin name, e.g. USDT, KSHIB
timestamp	LONG	YES	

Data Source: Database

Request for Quote

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
from_coin=BTC
to_coin=USDT
request_coin=BTC
request_amount=1

api_url="https://api.binance.us"

signature=`echo -n "fromCoin=$from_coin&toCoin=$to_coin&requestCoin=$request_coin&requestAmount=$request_amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/otc/quotes" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    --data-urlencode "fromCoin=$from_coin&toCoin=$to_coin&requestCoin=$request_coin&requestAmount=$request_amount&timestamp=$timestamp&signature=$signature"
```

Response

```
{
    "symbol": "BTCUSDT",
    "ratio": 50550.26,
    "inverseRatio": 0.00001978,
    "validTimestamp": 1641806714,
    "toAmount": 50550.26,
    "fromAmount": 1
}
```

`POST /sapi/v1/otc/quotes (HMAC SHA256)`

Use this endpoint to request a quote for a from-to coin pair.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
fromCoin	STRING	YES	From coin name, e.g. SHIB
toCoin	STRING	YES	To coin name, e.g. KSHIB
requestCoin	STRING	YES	Request coin name, e.g. SHIB
requestAmount	DECIMAL	YES	Amount of request coin, e.g. 50000
timestamp	LONG	YES	

Data Source: Database

Place OTC Trade Order

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
quote_id=<quoteId>

api_url="https://api.binance.us"

signature=`echo -n "quoteId=$quote_id&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/otc/orders" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    --data-urlencode "quoteId=$quote_id&timestamp=$timestamp&signature=$signature"
```

Response

```
{
    "orderId": "10002349",
    "createTime": 1641906714,
    "orderStatus": "PROCESS"  // Status: PROCESS / ACCEPT_SUCCESS / SUCCESS / FAIL
}
```

`POST /sapi/v1/otc/orders (HMAC SHA256)`

Use this endpoint to place an order using an acquired quote.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
quoteId	STRING	YES	Quote ID, e.g. 15848701022
timestamp	LONG	YES	

Data Source: Database

Get OTC Trade Order

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
orderid=<your_order_id>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/otc/orders/$orderid?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "quoteId": "4e5446f2cc6f44ab86ab02abf19a2fd2",
    "orderId": "10002349",
    "orderStatus": "SUCCESS",
    "fromCoin": "BTC",
    "fromAmount": 1,
    "toCoin": "USDT",
    "toAmount": 50550.26,
    "ratio": 50550.26,
    "inverseRatio": 0.00001978,
    "createTime": 1641806714
}
```

`GET /sapi/v1/otc/orders/{orderId} (HMAC SHA256)`

Use this endpoint to query OTC trade order details.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
orderId	STRING	YES	Order ID, e.g., 10002349
timestamp	LONG	YES	

Data Source: Database

Get All OTC Trade Orders

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/otc/orders?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "total": 2,
  "rows": [
          {
              "quoteId": "4e5446f2cc6f44ab86ab02abf19a2fd2",
              "orderId": "10002349",
              "orderStatus": "SUCCESS",
              "fromCoin": "BTC",
              "fromAmount": 1,
              "toCoin": "USDT",
              "toAmount": 50550.26,
              "ratio": 50550.26,
              "inverseRatio": 0.00001978,
            "createTime": 1641806714
        },
        {
            "quoteId": "15848645308",
            "orderId": "10002380",
            "orderStatus": "PROCESS",
            "fromCoin": "SHIB",
            "fromAmount": 10000,
            "toCoin": "KSHIB",
            "toAmount": 10,
            "ratio": 0.001,
            "inverseRatio": 1000,
            "createTime": 1641916714
        }
    ]
}
```

`GET /sapi/v1/otc/orders (HMAC SHA256)`

Use this endpoint to query OTC trade orders by condition.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
orderId	STRING	NO	Order ID
fromCoin	STRING	NO	From coin name, e.g., BTC, KSHIB
toCoin	STRING	NO	To coin name, e.g., USDT, SHIB
startTime	LONG	NO	Start timestamp
endTime	LONG	NO	End timestamp
page	INT	NO	Set the number of pages, depending on the number of records and the record limit for each page. No maximum value of pages.
limit	INT	NO	Number of records per page. Default: 10, Max: 100.
timestamp	LONG	YES	

Data Source: Database

Get All OCBS Trade Orders

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/ocbs/orders?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "total": 2,
  "dataList": [
    {
      "quoteId": "4e5446f2cc6f44ab86ab02abf19a7654",
      "orderId": "10090821212",
      "orderStatus": "SUCCESS",
      "fromCoin": "BTC",
      "fromAmount": 1,
      "toCoin": "USD",
      "toAmount": 50550.26,
      "feeCoin": "USD",
      "feeAmount": 0.26,
      "ratio": 50550.26,
      "createTime": 1641806714
    },
    {
      "quoteId": "4e5446f2cc6f44ab86ab02abf19abvd",
      "orderId": "1000238000",
      "orderStatus": "FAIL",
      "fromCoin": "USD",
      "fromAmount": 1000.5,
      "toCoin": "ETH",
      "toAmount": 0.5,
      "feeCoin": "USD",
      "feeAmount": 0.5,
      "ratio": 2000,
      "createTime": 1641916714
    }
  ]
}
```


`GET /sapi/v1/ocbs/orders (HMAC SHA256)`

Use this endpoint to query all OCBS orders by condition.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
orderId	STRING	NO	Order ID
startTime	LONG	NO	Start timestamp
endTime	LONG	NO	End timestamp
page	INT	NO	Set the number of pages, depending on the number of records and the record limit for each page. No maximum value of pages.
limit	INT	NO	Number of records per page. Default: 10, Max: 100.
timestamp	LONG	YES	

Data Source: Database

Wallet Endpoints

Asset Fees & Wallet Status

GET ASSET FEES & WALLET STATUS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/capital/config/getall?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
    {
        "coin": "BCH",
        "depositAllEnable": true,
        "withdrawAllEnable": true,
        "name": "BCH",
        "free": "0",
        "locked": "0",
        "freeze": "0",
        "withdrawing": "0",
        "ipoing": "0",
        "ipoable": "0",
        "storage": "0",
        "isLegalMoney": false,
        "trading": true,
        "networkList": [
            {
                "network": "BCH",
                "coin": "BCH",
                "withdrawIntegerMultiple": "0.00000001",
                "isDefault": true,
                "depositEnable": true,
                "withdrawEnable": true,
                "depositDesc": "",
                "withdrawDesc": "",
                "name": "Bitcoin Cash",
                "resetAddressStatus": false,
                "withdrawFee": "0",
                "withdrawMin": "0",
                "withdrawMax": "9999999"
            },
            {
                "network": "BNB",
                "coin": "BCH",
                "withdrawIntegerMultiple": "0.00000001",
                "isDefault": false,
                "depositEnable": true,
                "withdrawEnable": true,
                "depositDesc": "",
                "withdrawDesc": "",
                "name": "BEP222ee",
                "resetAddressStatus": false,
                "addressRegex": "^(bnb1)[0-9a-z]{38}$",
                "memoRegex": "^[0-9A-Za-z\\-_]{1,120}$",
                "withdrawFee": "0",
                "withdrawMin": "0",
                "withdrawMax": "9999999",
                "minConfirm": 15,
                "unLockConfirm": 3
            },
            {
                "network": "BSC",
                "coin": "BCH",
                "withdrawIntegerMultiple": "0.00000001",
                "isDefault": false,
                "depositEnable": true,
                "withdrawEnable": true,
                "depositDesc": "",
                "withdrawDesc": "",
                "name": "BSC (BEP20)",
                "resetAddressStatus": false,
                "addressRegex": "^(0x)[0-9A-Fa-f]{40}$",
                "memoRegex": "",
                "withdrawFee": "0",
                "withdrawMin": "0",
                "withdrawMax": "9999999",
                "minConfirm": 0,
                "unLockConfirm": 0
            }
        ]
    },
    {
    "coin": "PAX",
    "depositAllEnable": true,
    "withdrawAllEnable": true,
    "name": "PAX",
    "free": "0",
    "locked": "0",
    "freeze": "0",
    "withdrawing": "0",
    "ipoing": "0",
    "ipoable": "0",
    "storage": "0",
    "isLegalMoney": false,
    "trading": false,
    "networkList": [
        {
            "network": "BNB",
            "coin": "PAX",
            "withdrawIntegerMultiple": "0",
            "isDefault": false,
            "depositEnable": true,
            "withdrawEnable": true,
            "depositDesc": "",
            "withdrawDesc": "",
            "specialTips": "",
            "name": "BEP222ee",
            "resetAddressStatus": false,
            "addressRegex": "^(bnb1)[0-9a-z]{38}$",
            "memoRegex": "^[0-9A-Za-z\\-_]{1,120}$",
            "withdrawFee": "0",
            "withdrawMin": "0",
            "withdrawMax": "9999999",
            "minConfirm": 15,
            "unLockConfirm": 3
        }
    ]
    }
]
```

`GET /sapi/v1/capital/config/getall (HMAC SHA256)`

Use this endpoint to fetch the details of all crypto assets including fees, withdrawal limits, and network status.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Memory

Withdrawals

WITHDRAW FIAT VIA BITGO

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
paymentMethod=<paymentMethod>
paymentAccount=<paymentAccount>
amount=<amount>

api_url="https://api.binance.us"

signature=`echo -n "paymentMethod=$paymentMethod&paymentAccount=$paymentAccount&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/fiatpayment/withdraw/apply" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    --data-urlencode "paymentMethod=$paymentMethod&paymentAccount=$paymentAccount&amount=$amount&timestamp=$timestamp&signature=$signature"
```

Response

```
{
    "orderId": "6c2ff984890145fdac2b7160299062f0",
    "channelCode": "BITGO",
    "currencyCode": "USD",
    "amount": "100.00000000",
    "orderStatus": "INIT"
}
```

`POST /sapi/v1/fiatpayment/withdraw/apply (HMAC SHA256)`

Use this endpoint to submit a USD withdraw request via BITGO

Weight: 1

Parameters:

Name	Type	Mandatory	Description
paymentMethod	STRING	YES	default value="BITGO"
paymentAccount	STRING	YES	The account to which the user wants to withdraw funds
fiatCurrency	STRING	NO	default value="USD"
amount	DECIMAL	YES	The amount
recvWindow	LONG	NO	
timestamp	LONG	YES	

WITHDRAW CRYPTO

Example

```
#!/bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
coin=<coin>
network=<network>
address=<address>
amount=<amount>

api_url="https://api.binance.us"

signature=`echo -n "coin=$coin&network=$network&address=$address&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

payload="coin=$coin&network=$network&address=$address&amount=$amount&timestamp=$timestamp&signature=$signature"

curl -X "POST" "$api_url/sapi/v1/capital/withdraw/apply" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    -d $payload
```

Response

```
{"id":"4e7f4f31560041ee880b5386f06d4053"}
```

`POST /sapi/v1/capital/withdraw/apply (HMAC SHA256)`

Use this endpoint to submit a crypto withdrawal request.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
coin	STRING	YES	
network	STRING	YES	Specify the withdrawal network (e.g. 'ERC20' or 'BEP20'). Please ensure the address type is correct for the chosen network
withdrawOrderId	STRING	NO	Client ID for withdraw
address	STRING	YES	
addressTag	STRING	NO	Memo: Acts as a secondary address identifier for coins like XRP, XMR etc.
amount	DECIMAL	YES	
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Memory

GET CRYPTO WITHDRAWAL HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/capital/withdraw/history?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
  {
    "id": "4e7f4f31560041ee880b5386f06d4053",
    "amount": "0.9",
    "transactionFee": "0.1",
    "coin": "BNB",
    "status": 2,
    "address": "0xd709f9d0bbc6b0e746a13142dfe353086edf87c2",
    "applyTime": "2022-02-18 03:36:04",
    "network": "BSC",
    "transferType": 0
  }
]
```


`GET /sapi/v1/capital/withdraw/history (HMAC SHA256)`

Use this endpoint to fetch your crypto withdrawal history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
coin	STRING	YES	
withdrawOrderId	STRING	NO	Client ID for withdraw
status	INT	NO	0: email sent, 1: canceled, 2: awaiting approval, 3: rejected, 4: processing, 5: failure, 6: completed
startTime	LONG	NO	Default: 90 days from current timestamp
endTime	LONG	NO	Default: present timestamp
offset	INT	NO	Default: 0
limit	INT	NO	Default: 1000, max: 1000
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Database

GET FIAT WITHDRAWAL HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/fiatpayment/query/withdraw/history?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "assetLogRecordList": [
        {
            "orderId":"6c2ff984890145fdac2b7160299062f0",
```

```
            "paymentAccount": "4a992541-c12d-4cca-bbd6-df637f801526",
            "paymentChannel": "PRIMETRUST",
            "paymentMethod": "WIRE_INTERNATIONAL",
            "orderStatus": "Processing",
            "amount": "65",
        "transactionFee": "20",
            "platformFee": "0"
        }
    ]
}
```


`GET /sapi/v1/fiatpayment/query/withdraw/history (HMAC SHA256)`

Use this endpoint to fetch your fiat (USD) withdrawal history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
fiatCurrency	STRING	NO	
orderId	STRING	NO	
offset	INT	NO	
paymentChannel	STRING	NO	
paymentMethod	STRING	NO	
startTime	LONG	NO	Default to 90 days from current timestamp
endTime	LONG	NO	Default to current timestamp
recvWindow	Long	NO	
timestamp	Long	YES	

- Please pay attention to the default value of startTime and endTime.
- If both startTime and endTime are sent, the duration between startTime and endTime must be greater than 0 day and less than 90 days.

Deposits

GET CRYPTO DEPOSIT ADDRESS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
coin=<coin>

api_url="https://api.binance.us"

signature=`echo -n "coin=$coin&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/capital/deposit/address?coin=$coin&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "coin": "BNB",
  "address": "0xcf1988d27e402086941284313b632466fdf28a22",
  "tag": "",
  "url": "0xcf1988d27e402086941284313b632466fdf28a22"
}
```


`GET /sapi/v1/capital/deposit/address (HMAC SHA256)`

Use this endpoint to fetch a deposit address for a particular crypto asset.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
coin	STRING	YES	
network	STRING	NO	
recvWindow	LONG	NO	
timestamp	LONG	YES	

GET CRYPTO DEPOSIT HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
coin=<coin>

api_url="https://api.binance.us"

signature=`echo -n "coin=$coin&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`
```

```
curl -X "GET" "$api_url/sapi/v1/capital/deposit/hisrec?coin=$coin&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
  {
    "amount": "8.73234",
    "coin": "BNB",
    "network": "BSC",
    "status": 1,
    "address": "0xd709f9d0bbc6b0e746a13142dfe353086edf87c2",
    "addressTag": "",
    "txId": "0xa9ebf3f4f60bc18bd6bdf4616ff8ffa14ef93a08fe79cad40519b31ea1044290",
    "insertTime": 1638342348000,
    "transferType": 0,
    "confirmTimes": "0/0"
  }
]
```


`GET /sapi/v1/capital/deposit/hisrec (HMAC SHA256)`

Use this endpoint to fetch your crypto deposit history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
coin	STRING	YES	
status	INT	NO	0: pending, 6: credited but cannot withdraw, 1: success
startTime	LONG	NO	Default: 90 days from current timestamp
endTime	LONG	NO	Default: present timestamp
offset	INT	NO	Default: 0
limit	INT	NO	Default: 1000, max: 1000
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Memory

GET FIAT DEPOSIT HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/fiatpayment/query/deposit/history?timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "assetLogRecordList": [
        {
            "orderId": "c3415d574fa747df88a75c66c037f890",
            "paymentAccount": "ab13f629-b074-470a-bc80-b830e169691f",
            "paymentChannel": "MODERNTREASURY",
            "paymentMethod": "ACH",
            "orderStatus": "Successful",
            "fiatCurrency": "USD",
            "amount": "0.99",
            "transactionFee": "0.01",
            "platformFee": "0"
        }
    ]
}
```


`GET /sapi/v1/fiatpayment/query/deposit/history (HMAC SHA256)`

Use this endpoint to fetch your fiat (USD) deposit history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
fiatCurrency	STRING	NO	
orderId	STRING	NO	
offset	INT	NO	
paymentChannel	STRING	NO	
paymentMethod	STRING	NO	
startTime	LONG	NO	Default to 90 days from current timestamp
endTime	LONG	NO	Default to current timestamp

Name	Type	Mandatory	Description
recvWindow	Long	NO	
timestamp	Long	YES	

- Please pay attention to the default value of startTime and endTime.
- If both startTime and endTime are sent, the duration between startTime and endTime must be greater than 0 day and less than 90 days.

GET SUB-ACCOUNT DEPOSIT ADDRESS

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

email="ios@mt.com"
coin="BNB"

signature=`echo -n "email=$email&coin=$coin&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/capital/sub-account/deposit/address?email=$email&coin=$coin&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "coin": "BNB", // coin
    "address": "0xf79b6274f18425441b8f05c0a711d171c0fe119f", // address
    "tag": "", //memo
    "url": "https://etherscan.io/address/0xf79b6274f18425441b8f05c0a711d171c0fe119f" //Blockchain browser address
}
```

`GET /sapi/v1/capital/sub-account/deposit/address (HMAC SHA256)`

Use this endpoint to fetch a sub-account's deposit address.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	YES	Sub-account Email
coin	STRING	YES	coin
network	STRING	NO	Network (If empty, returns the default network)

Data Source: Database

GET SUB-ACCOUNT DEPOSIT HISTORY

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

email="ios@mt.com"

signature=`echo -n "email=$email&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/capital/sub-account/deposit/history?email=$email&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
    {
        "amount": "9.9749",//deposit amount
        "coin": "BTC", //coin
        "network": "btc", //network
        "status": 4,
        "address": "bc1qxurvdd7tzn09agdvg3j8xpm3f7e978y07wg83s",
        "addressTag": "",
        "txId": "0x1b4b8c8090d15e3c1b0476b1c19118b1f00066e01de567cd7bc5b6e9c100193f",
        "insertTime": 1652942429211,
        "transferType": 0,
        "confirmTimes": "0/0"
    }
]
```

`GET /sapi/v1/capital/sub-account/deposit/history (HMAC SHA256)`

Use this endpoint to fetch sub-account deposit history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
email	STRING	YES	Sub-account Email
coin	STRING	NO	coin

Name	Type	Mandatory	Description
status	INT	NO	0 (0:pending, 6:credited but cannot withdraw, 1:success)
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	
offset	INT	NO	default: 0

Data Source: Database

Convert Dust

Convert Dust

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

fromAsset=<fromAsset>
toAsset=<toAsset>

signature=`echo -n "fromAsset=$fromAsset&toAsset=$toAsset&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X POST "$api_url/sapi/v1/asset/dust?fromAsset=$fromAsset&toAsset=$toAsset&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "totalTransferred": "0.00289855",
  "totalServiceCharge": "0.00005797",
  "transferResult": [
       {
          "tranId": 28822799,
          "fromAsset": "LTC",
          "toAsset": "BTC",
          "amount": "0.2",
          "transferredAmount": "0.00289855",
          "serviceChargeAmount": "0.00005797",
          "operateTime": 1659583487273,
          "result": "S"
       }
  ]
}
```

`POST /sapi/v1/asset/dust (HMAC SHA256)`

Use this endpoint to convert dust assets to BNB/BTC/ETH/USDT.

Weight(UID): 10

Parameters:

Name	Type	Mandatory	Description
fromAsset	ARRAY	YES	The assets being converted. For example: fromAsset=BTC&fromAsset=ETH
toAsset	STRING	YES	To asset name, e.g. BNB, BTC, ETH, USDT.

Data Source: Database

Get Convert Dust History

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

startTime=<startTime>
endTime=<endTime>

api_url="https://api.binance.us"

signature=`echo -n "startTime=$startTime&endTime=$endTime&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X GET "$api_url/sapi/v1/asset/query/dust-logs?startTime=$startTime&endTime=$endTime&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "total": 1,
  "userDustConvertHistory": [
       {
          "operateTime": 1659583487000,
          "totalServiceChargeAmount": "0.00005797",
          "totalTransferredAmount": "0.00284058",
          "userAssetDribbletDetails": [
```

```
                {
                    "fromAsset": "LTC",
                    "toAsset": "BTC",
                    "amount": "0.2",
                    "transferredAmount": "0.00284058",
                    "serviceChargeAmount": "0.00005797",
                    "operateTime": 1659583487000,
                    "tranId": 28822799
                }
            ]
        }
    ]
}
```

`GET /sapi/v1/asset/query/dust-logs (HMAC SHA256)`

Use this endpoint to get dust conversion history.

Weight(IP): 1

Parameters:

Name	Type	Mandatory	Description
startTime	LONG	YES	Start time
endTime	LONG	YES	End time

Data Source: Database

Get Assets That Can Be Converted

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

toAsset=<toAsset>

api_url="https://api.binance.us"

signature=`echo -n "toAsset=$toAsset&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X GET "$api_url/sapi/v1/asset/query/dust-assets?toAsset=$toAsset&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "convertibleAssets":[
        {
            "fromAsset": "BTC",
            "toAsset": "BNB",
            "availableBalance": "0.0001",
            "convertedAsset": "0.0028987",
            "usdValueConvertedAsset": "2.0001",
            "conversionFee": "0.00005797",
            "receivedAsset": "0.00284073"
        },
        {
            "fromAsset": "USDT",
            "toAsset": "BNB",
            "availableBalance": "14.286",
            "convertedAsset": "0.02029026",
            "usdValueConvertedAsset": "14.00028",
            "conversionFee": "0.00040581",
            "receivedAsset": "0.01988445"
        }
    ],
    "totalConvertedAsset": "0.02318896",
    "usdValueTotalConvertedAsset": "16.00038",
    "totalConversionFee": "0.00046378",
    "totalReceivedAsset": "0.02272518",
    "feeRate": "0.02",
    "withinRestrictedTime": false
}
```

`GET /sapi/v1/asset/query/dust-assets (HMAC SHA256)`

Use this endpoint to get your dust assets that can be converted.

Weight(IP): 1

Parameters:

Name	Type	Mandatory	Description
toAsset	STRING	YES	To asset name, e.g. BNB, BTC, ETH, USDT.

Data Source: Database

Referral Endpoints

Get Referral Reward History

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
userBizType=<your userBizType>
page=<page>
rows=<row>

api_url="https://api.binance.us"

signature=`echo -n "userBizType=$userBizType&page=$page&rows=$rows&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/marketing/referral/reward/history?userBizType=$userBizType&page=$page&rows=$rows&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "total": 1,
    "rows": [
        {
            "userId": 350991652,
            "rewardAmount": "8",
            "receiveDateTime": 1651131084091,
            "rewardType": "USD"
        }
    ]
}
```


`GET /sapi/v1/marketing/referral/reward/history (HMAC SHA256)`

Use this endpoint to get the user's referral reward history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
userBizType	INT	YES	user business type(0: referrer, 1: referee)
page	INT	YES	
rows	INT	YES	min: 1, max: 200
timestamp	LONG	YES	

Data Source: Database

Staking Endpoints

Get Staking Asset Information

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
stakingAsset=<stakingAsset>
api_url="https://api.binance.us"
secret_key=<your_secret_key>

signature=`echo -n "stakingAsset=$stakingAsset&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/staking/asset?stakingAsset=$stakingAsset&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/json"
```

Response


```
[
    {
        "stakingAsset": "BNB",  // Asset is being staked
        "rewardAsset": "BNB",  // Asset received as staking reward
        "apr": "0.0517",  // Annualized rate of return without rewards restaked
        "apy": "0.04",  // Annualized rate of return with rewards restaked
        "unstakingPeriod": 168,  // Amount of time to unstake asset in hours
        "minStakingLimit": "0.01", // Minimum amount allowed for staking
        "maxStakingLimit": "10000", // Maximum amount allowed for staking
        "autoRestake": true // Are rewards for this asset automatically restaked
    }
]
```


`GET sapi/v1/staking/asset (HMAC SHA256)`

Use this endpoint to get staking information for a supported asset (or assets)

Weight: 1

Parameters:

Name	Type	Mandatory	Description
stakingAsset	String	NO	Asset symbol (e.g. BNB). If empty, returns all staking assets

Data Source: Database

Stake Asset

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
stakingAsset=<your_asset>
amount=<your_amount>
autoRestake=<your_autoRestake>
api_url="https://api.binance.us"

signature=`echo -n "stakingAsset=$stakingAsset&amount=$amount&autoRestake=$autoRestake&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/staking/stake?userId=$userId&asset=$asset&amount=$amount&autoRestake=$autoRestake&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/x-www-form-urlencoded;charset=utf-8"
```

Response


```
{
  "code": "000000",
  "message": "success",
  "data":
      {
          "result": "SUCCESS",
          "purchaseRecordId": "111"
      },
  "success": true
}
```


`POST sapi/v1/staking/stake (HMAC SHA256)`

Use this endpoint to stake a supported asset.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
stakingAsset	STRING	YES	Asset symbol (e.g. BNB)
amount	DECIMAL	YES	Staking amount
autoRestake	BOOLEAN	NO	If need auto restaking - default: true

Data Source: Database

Unstake Asset

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
stakingAsset=<your_asset>
amount=<your_amount>

api_url="https://api.binance.us"

signature=`echo -n "stakingAsset=$stakingAsset&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/staking/unstake?stakingAsset=$stakingAsset&amount=$amount&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/x-www-form-urlencoded;charset=utf-8"
```

Response


```
{
   "code": "000000",
   "message": "success",
   "data":
       {
           "result": "SUCCESS"
       },
   "success": true
}
```


`POST sapi/v1/staking/unstake (HMAC SHA256)`

Use this endpoint to unstake a staked asset.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
stakingAsset	STRING	YES	Asset symbol (e.g. BNB)
amount	DECIMAL	YES	Unstaking amount

Data Source: Database

Get Staking Balance

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

asset=<asset>

signature=`echo -n "asset=$asset&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/staking/stakingBalance?asset=$asset&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "code": "000000",
    "message": "success",
    "data": [
        {
            "asset": "BNB",
            "stakingAmount": "3882.50133916",
            "rewardAsset": "BNB",
            "apr": "0.0517",
            "apy": "0.0869",
            "autoRestake": true
        }
    ],
    "status": ["holding"],
    "success": true
}
```

`GET /sapi/v1/staking/stakingBalance (HMAC SHA256)`

Use this endpoint to get the staking balance for an asset(or assets).

Weight: 1

Parameters:

Name	Type	Mandatory	Description
asset	String	NO	Staked asset. If empty, returns all assets with balances.

Data Source: Database

Get Staking History

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
asset=<asset>
startTime=<startTime>
endTime=<endTime>
Page=<page>
limit=<limit>
api_url="https://api.binance.us"

signature=`echo -n "asset=$asset&startTime=$startTime&endTime=$endTime&page=$page&limit=<limit>&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/staking/history?asset=$asset&startTime=$startTime&endTime=$endTime&page=$page&limit=<limit>&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/json"
```

Response

```
[
    {
        "asset": "ETH",
        "amount": "2500", // amount of the transaction
        "type": "staked", //transaction type
        "initiatedTime": 1663096490748, // transaction initiation time
        "status": "SUCCESS" //// status of the transaction
    },
    {
        "asset": "ETH",
        "amount": "1",
        "type": "staked",
        "initiatedTime": 1665462088011,
        "status": "SUCCESS"
    }
]
```

`GET sapi/v1/staking/history (HMAC SHA256)`

Use this endpoint to get the staking history of an asset (or assets) within a given time range.

Weight: 1

Parameters:

Name	Type	Mandatory	Description

Name	Type	Mandatory	Description
asset	STRING	NO	Asset symbol (e.g. BNB). If empty, returns all assets with history
startTime	LONG	NO	UNIX Timestamp
endTime	LONG	NO	UNIX Timestamp
page	INT	NO	Page number - default: 1
limit	INT	NO	Default value: 500 (each page contains at most 500 records)

Data Source: Database

Get Staking Rewards History

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/staking/stakingRewardsHistory?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "code": "000000",
  "message": "success",
  "data": [
      {
          "asset": "BNB",
          "amount": "0.03371769",
          "usdValue": "20.23",
          "time": 1658475133009,
          "tranId": 3123201,
          "autoRestaked": false
      }
  ],
  "total": 1,
  "success": true
}
```


`GET /sapi/v1/staking/stakingRewardsHistory (HMAC SHA256)`

Use this endpoint to get the staking rewards history for an asset(or assets) within a given time range.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
asset	String	NO	Staked asset. If empty, returns all assets with balances.
startTime	LONG	NO	Start time
endTime	LONG	NO	End time
page	INTEGER	NO	The transfer history batch number(each batch contains at most 500 transfer history records)
limit	INTEGER	NO	Default value: 500

Data Source: Database

Custodial Solution Endpoints

User Account Data (Custodial)

GET ACCOUNT BALANCE

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "GET" "$api_url/sapi/v1/custodian/balance?$parameter_concat&signature=$signature" \
  -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
      "exchangeWalletBalance": [
          {
              "asset": "BNB",
              "free": "0.0083686",
              "locked": "0",
              "lastUpdatedTime": 1662535206000
          }
     ],
      "custodialAcctBalance": [
          {
              "asset": "BTC",
              "free": "9.02933865",
              "locked": "0",
              "lastUpdatedTime": 1658478839000
          },
          {
              "asset": "ETHT",
              "free": "61.00129998",
              "locked": "0",
              "inSettlement": "25.17",
              "lastUpdatedTime": 1663752210000
          }
     ]
  }
```


`GET /sapi/v1/custodian/balance (HMAC SHA256)`

Use this endpoint to get balance information for Binance.US exchange wallet and Binance.US custodial sub-account.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
timestamp	LONG	YES	

Data Source: Database

GET SUPPORTED ASSET LIST

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>

api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/supportedAssetList?rail=$rail&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
    "transferEligible":[
        {
            "asset": "BTC",
            "precision": 8,
            "network": [
                "BTC"
            ]
        },
        {
            "asset": "ETH",
            "precision": 8,
            "network": [
                "ETH"
            ]
        }
    ],
    "settlementEligible":[
        {
            "asset": "BTC",
            "precision": 8,
            "network": [
                "BTC"
            ]
        },
        {
            "asset": "ETH",
            "precision": 8,
            "network": [
                "ETH"
            ]
        }
    ]
}
```


`GET /sapi/v1/custodian/supportedAssetList (HMAC SHA256)`

Use this endpoint to get a list of assets supported with custodial solutions including eligibility for transfer (from custodial partner) and settlement (to custodial partner).

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
timestamp	LONG	YES	Current timestamp.

Data Source: Database

Transfer (Custodial)

TRANSFER FROM EXCHANGE WALLET

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
asset=<asset>
amount=<amount>

api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&asset=$asset&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

payload="rail=$rail&asset=$asset&amount=$amount&timestamp=$timestamp&signature=$signature"

curl -X "POST" "$api_url/sapi/v1/custodian/walletTransfer" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    -d $payload
```

Response

```
{
    "asset": "BTC",
    "amount": 1,
    "clientOrderId": "1663752208086",
    "transferId": "2022092709232937542366",
    "status": "SUCCESS",
    "createTime": 1664276400000
}
```

`POST /sapi/v1/custodian/walletTransfer (HMAC SHA256)`

Use this endpoint to request an asset transfer from your Binance.US exchange wallet to your Binance.US custodial sub-account.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
asset	STRING	YES	
amount	DECIMAL	YES	
clientOrderId	STRING	NO	Your reference ID for the order, must be unique. Automatically generated if not sent.
timestamp	LONG	YES	Current timestamp.

Data Source: Database

TRANSFER FROM CUSTODIAN

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
asset=<asset>
amount=<amount>

api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&asset=$asset&amount=$amount&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

payload="rail=$rail&asset=$asset&amount=$amount&timestamp=$timestamp&signature=$signature"

curl -X "POST" "$api_url/sapi/v1/custodian/custodianTransfer" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    -d $payload
```

Response

```
{
    "asset": "BTC",
    "amount": 1,
    "clientOrderId": "1663752208086",
    "transferId": "2022092709232937542366",
    "custodyAccountId": "custodyAccountId",
    "custodyAccountName": "custodyAccountName",
    "status": "Transfer request received ",
    "createTime": 1664276400000
}
```

`POST /sapi/v1/custodian/custodianTransfer (HMAC SHA256)`

Use this endpoint to request an asset transfer from a custodial partner account to the Binance.US custodial sub-account.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
asset	STRING	YES	
amount	DECIMAL	YES	
clientOrderId	STRING	NO	Your reference ID for the order, must be unique. Automatically generated if not sent.
timestamp	LONG	YES	Current timestamp.

Data Source: Database

UNDO TRANSFER

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
originTransferId=<originTransferId>


api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&originTransferId=$originTransferId&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

payload="rail=$rail&originTransferId=$originTransferId&timestamp=$timestamp&signature=$signature"

curl -X "POST" "$api_url/sapi/v1/custodian/undoTransfer" \
    -H "X-MBX-APIKEY: $api_key" \
    -H 'Content-Type: application/x-www-form-urlencoded; charset=utf-8' \
    -d $payload
```

Response


```
{
    "transferId": "2022092709232937542366",
    "asset": "BTC",
    "amount": 1
}
```


`POST /sapi/v1/custodian/undoTransfer (HMAC SHA256)`

Use this endpoint to undo a previous transfer from your custodial partner.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
originTransferId	STRING	YES	Previous transfer ID.
timestamp	LONG	YES	Current timestamp.

Data Source: Database

GET EXCHANGE WALLET TRANSFER

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>

api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/walletTransferHistory?rail=$rail&timestamp=$timestamp&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "data": [
      {
          "transferId": "2022092709232937542366",
          "clientOrderId": "1663752208086",
          "asset": "BTC",
          "amount": 1,
          "status": "SUCCESS",
          "createTime": 1664276400000,
          "updateTime": 1664276400000
      }
  ],
  "total": 1
}
```


`GET /sapi/v1/custodian/walletTransferHistory (HMAC SHA256)`

Use this endpoint to check a Binance.US exchange wallet transfer status.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
transferId	STRING	NO	
clientOrderId	STRING	NO	
asset	STRING	NO	BTC,etc
startTime	LONG	NO	Default: 90 days from current timestamp
endTime	LONG	NO	Default: current timestamp
page	INT	NO	defaultValue:1
limit	INT	NO	defaultValue:20, max:100
timestamp	LONG	YES	Current timestamp.

Data Source: Database

GET CUSTODIAN TRANSFER

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>

api_url="https://api.binance.us"

signature=`echo -n "rail=$rail&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/custodianTransferHistory?rail=$rail&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
  "data": [
      {
          "transferId": "20220927092329375423366",
          "clientOrderId": "1663752208086",
          "asset": "BTC",
          "amount": 1,
          "status": "SUCCESS",
          "expressTrade": FALSE,
          "createTime": 1664276400000,
          "updateTime": 1664276400000
      }
  ],
  "total": 1
}
```

`GET /sapi/v1/custodian/custodianTransferHistory (HMAC SHA256)`

Use this endpoint to check the status of a transfer from a custodial partner account, including ExpressTrade transfer, Custodian transfer and Undo Transfer.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
transferId	STRING	NO	
clientOrderId	STRING	NO	
expressTradeTransfer	BOOLEAN	NO	Default FALSE
asset	STRING	NO	BTC,etc
startTime	LONG	NO	Default: 90 days from current timestamp
endTime	LONG	NO	Default: current timestamp
page	INT	NO	defaultValue:1
limit	INT	NO	defaultValue:20, max:100
timestamp	LONG	YES	Current timestamp.

Data Source: Database

Trade Order (Custodial)

CREATE NEW ORDER (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature
timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
symbol=<symbol>
side=<side>
type=<type>
quantity=<quantity>
price=<price>
timeInForce=<timeInForce>
asset=<asset>
allowExpressTrade=<allowExpressTrade>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&symbol=$symbol&side=$side&type=$type&quantity=$quantity&price=$price&timeInForce=$timeInForce&asset=$asset&allowExpressTrade=$allowExpressTrade&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "POST" "$api_url/sapi/v1/custodian/order?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "symbol": "ETHTUSD",
  "orderId": 70,
  "orderListId": -1,
  "clientOrderId": "7c713b92cce34358b23fcf1fcd9953bc",
  "price": "18",
  "origQty": "1",
  "executedQty": "0",
  "cummulativeQuoteQty": "0",
  "status": "NEW",
  "timeInForce": "GTC",
  "type": "LIMIT",
  "side": "BUY",
  "expressTradeFlag": false
}
```


`POST /sapi/v1/custodian/order (HMAC SHA256)`

Use this endpoint to place a new trade order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD)
side	ENUM	YES	Order side (e.g., BUY, SELL)
type	ENUM	YES	(Order type (e.g., LIMIT, MARKET, STOP_LOSS, STOP_LOSS_LIMIT, TAKE_PROFIT, TAKE_PROFIT_LIMIT, LIMIT_MAKER))
timeInForce	ENUM	NO	
quantity	DECIMAL	NO	
quoteOrderQty	DECIMAL	NO	
price	DECIMAL	NO	Order price
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
asset	STRING	NO	Optional. When allowExpressTrade=true, enter the asset you are selling. E.g. If symbol = BTCUSD, and side = BUY, enter “USD”.
allowExpressTrade	BOOLEAN	NO	Default false; if true, when Binance.US Custodial sub-account Balance is smaller than the order amount, full amount will be transferred from custodial partner account.
timestamp	LONG	YES	Current timestamp.

Some additional mandatory parameters based on order type:

Type	Additional Mandatory Parameters	Additional Information
LIMIT	timeInForce, quantity, price	
MARKET	quantity or quoteOrderQty	MARKET orders using the quantity field specifies the amount of the base asset the user wants to buy or sell at the market price. E.g., a MARKET order on BTCUSDT will specify how much BTC the user is buying or selling MARKET orders using quoteOrderQty specify the amount the user wants to spend (when buying) or receive (when selling) the quote asset; the correct quantity will be determined based on the market liquidity and quoteOrderQty. E.g., Using the symbol BTCUSDT:BUY side, the order will buy as many BTC as quoteOrderQty USDT can. SELL side, the order will sell as much BTC needed to receive quoteOrderQty USDT
STOP_LOSS	quantity, stopPrice	This will execute a MARKET order when the stopPrice is reached
STOP_LOSS_LIMIT	timeInForce, quantity, price, stopPrice	This will execute a LIMIT order when the stopPrice is reached
TAKE_PROFIT	quantity, stopPrice	This will execute a MARKET order when the stopPrice is reached
TAKE_PROFIT_LIMIT	timeInForce, quantity, price, stopPrice	This will execute a LIMIT order when the stopPrice is reached
LIMIT_MAKER	quantity, price	This is a LIMIT order that will be rejected if the order immediately matches and trades as a taker. This is also known as a POST-ONLY order

Data Source: Database

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
symbol=<symbol>
side=<side>
quantity=<quantity>
price=<price>
stopPrice=<stopPrice>
stopLimitPrice=<stopLimitPrice>
stopLimitTimeInForce=<stopLimitTimeInForce>
asset=<asset>
allowExpressTrade=<allowExpressTrade>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&symbol=$symbol&side=$side&quantity=$quantity&price=$price&stopPrice=$stopPrice&stopLimitPrice=$stopLimitPrice&stopLimitTimeInForce=$stopLimitTimeInForce&asset=$asset&allowExpressTrade=$allowExpressTrade&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "POST" "$api_url/sapi/v1/custodian/ocoOrder?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "orderListId": 0,
  "contingencyType": "OCO",
  "listStatusType": "EXEC_STARTED",
  "listOrderStatus": "EXECUTING",
  "listClientOrderId": "JYVpp3F0f5CAG15DhtrqLp",
  "transactionTime": 1563417480525,
  "symbol": "LTCBTC",
  "expressTradeFlag": false,
  "orders": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "clientOrderId": "xTXKaGYd4bluPVp78IVRvl"
    }
  ],
  "orderReports": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "orderListId": 0,
      "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos",
      "transactTime": 1563417480525,
      "price": "0.000000",
      "origQty": "0.624363",
      "executedQty": "0.000000",
      "cummulativeQuoteQty": "0.000000",
      "status": "NEW",
      "timeInForce": "GTC",
      "type": "STOP_LOSS",
      "side": "BUY",
      "stopPrice": "0.960664"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "orderListId": 0,
      "clientOrderId": "xTXKaGYd4bluPVp78IVRvl",
      "transactTime": 1563417480525,
      "price": "0.036435",
      "origQty": "0.624363",
      "executedQty": "0.000000",
      "cummulativeQuoteQty": "0.000000",
      "status": "NEW",
      "timeInForce": "GTC",
      "type": "LIMIT_MAKER",
      "side": "BUY"
    }
  ]
}
```


`POST /sapi/v1/custodian/ocoOrder (HMAC SHA256)`

Use this endpoint to place a new OCO(one-cancels-the-other) order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).e.g.,ANCHORAGE
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD)
side	ENUM	YES	Order side (e.g., BUY, SELL)
quantity	DECIMAL	YES	
limitClientOrderId	STRING	NO	A unique ID for the limit order
price	DECIMAL	YES	
limitIcebergQty	DECIMAL	NO	
stopClientOrderId	STRING	NO	A unique ID for the stop loss/stop loss limit leg

Name	Type	Mandatory	Description
stopPrice	DECIMAL	YES	
stopLimitPrice	DECIMAL	NO	If provided, stopLimitTimeInForce is required
stopIcebergQty	DECIMAL	NO	
stopLimitTimeInForce	ENUM	NO	Valid values are GTC/FOK/IOC
asset	STRING	NO	Optional. When allowExpressTrade=true, enter the asset you are selling. E.g. If symbol = BTCUSD, and side = BUY, enter "USD".
allowExpressTrade	BOOLEAN	NO	Default false; if true, when Binance.US Custodial sub-account Balance is smaller than the order amount, full amount will be transferred from custodial partner account.
timestamp	LONG	YES	

Other Info: Price Restrictions: SELL: Limit Price > Last Price > Stop Price BUY: Limit Price < Last Price < Stop Price Quantity Restrictions: Both legs must have the same quantity. ICEBERG quantities however do not have to be the same Order Rate Limit OCO counts as 2 orders against the order rate limit.

Data Source: Matching Engine

GET ALL OPEN ORDERS (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "GET" "$api_url/sapi/v1/custodian/openOrders?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
        "symbol": "LTCBTC",
        "orderId": 1,
        "orderListId": -1, //Unless OCO, the value will always be -1
        "clientOrderId": "myOrder1",
        "price": "0.1",
        "origQty": "1.0",
        "executedQty": "0.0",
        "cummulativeQuoteQty": "0.0",
        "status": "NEW",
        "timeInForce": "GTC",
        "type": "LIMIT",
        "side": "BUY",
        "stopPrice": "0.0",
        "icebergQty": "0.0",
        "time": 1499827319559,
        "updateTime": 1499827319559,
        "isWorking": true,
        "origQuoteOrderQty": "0.000000",
        "expressTradeFlag": true
  }
]
```

`GET /sapi/v1/custodian/openOrders (HMAC SHA256)`

Use this endpoint to get all open trade orders for a token symbol. Do not access this without a token symbol as this would return all pair data.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	NO	Order trading pair (e.g., BTCUSD, ETHUSD).
timestamp	LONG	YES	Current timestamp.

Data Source: Matching Engine

GET ORDER (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
orderId=<orderId>
symbol=<symbol>

api_url="https://api.binance.us"

parameter_concat="rail=$rail&orderId=$orderId&symbol=$symbol&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/order?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
        "symbol": "LTCBTC",
        "orderId": 1,
        "orderListId": -1, //Unless OCO, the value will always be -1
        "clientOrderId": "myOrder1",
        "price": "0.1",
        "origQty": "1.0",
        "executedQty": "0.0",
        "cummulativeQuoteQty": "0.0",
        "status": "NEW",
        "timeInForce": "GTC",
        "type": "LIMIT",
        "side": "BUY",
        "stopPrice": "0.0",
        "icebergQty": "0.0",
        "time": 1499827319559,
        "updateTime": 1499827319559,
        "isWorking": true,
        "origQuoteOrderQty": "0.000000",
        "expressTradeFlag": true
 }
```

`GET /sapi/v1/custodian/order (HMAC SHA256)`

Use this endpoint to check a trade order's status.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD).
orderId	LONG	YES	
timestamp	LONG	YES	Current timestamp.

Data Source: Database

GET ORDER HISTORY (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "GET" "$api_url/sapi/v1/custodian/orderHistory?$parameter_concat&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
  {
    "symbol": "1565776717369",
    "orderId": 11,
    "orderListId": -1,
    "clientOrderId": "mhUTILjXKS0BHTaBG0is1p",
    "price": "0.0000",
    "origQty": "5.000000",
    "executedQty": "5.000000",
    "cummulativeQuoteQty": "9.5000",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "TAKE_PROFIT",
    "side": "BUY",
    "stopPrice": "2.0000",
    "icebergQty": "0.000000",
    "time": 1565776718390,
    "updateTime": 1565776718779,
    "isWorking": true,
    "origQuoteOrderQty": "0.000000",
    "expressTradeFlag": false
  }
]
```

`GET /sapi/v1/custodian/orderHistory (HMAC SHA256)`

Use this endpoint to check an order's status as well as past orders.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	NO	Order trading pair (e.g., BTCUSD, ETHUSD).
startTime	LONG	NO	
endTime	LONG	NO	

Name	Type	Mandatory	Description
fromId	LONG	NO	defaultValue:1
limit	INT	NO	defaultValue:200
timestamp	LONG	YES	

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

Data Source: Matching Engine

GET TRADE HISTORY (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "GET" "$api_url/sapi/v1/custodian/tradeHistory?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
    {
        "symbol": "BTCUSD",
        "price": "10000",
        "qty": "1",
        "quoteQty": "10000",
        "time": 1662027374093,
        "isBuyer": true,
        "isMaker": false,
        "isBestMatch": true,
        "orderId": 115,
        "orderListId": -1,
        "commission": "0",
        "commissionAsset": "BTC"
    },
    {
        "symbol": "BTCUSD",
        "price": "5000",
        "qty": "0.0001",
        "quoteQty": "0.5",
        "time": 1662029127602,
        "isBuyer": false,
        "isMaker": false,
        "isBestMatch": true,
        "orderId": 111,
        "orderListId": -1,
        "commission": "0",
        "commissionAsset": "USD"
    }
]
```


`GET /sapi/v1/custodian/tradeHistory (HMAC SHA256)`

Use this endpoint to get past trade data.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	NO	Order trading pair (e.g., BTCUSD, ETHUSD).
orderId	LONG	NO	
startTime	LONG	NO	
endTime	LONG	NO	
fromId	LONG	NO	
limit	INT	NO	defaultValue:200
timestamp	LONG	YES	

Notes: If fromId is set, it will get orders >= than fromId. Otherwise most recent orders are returned.

Data Source: Database

CANCEL ORDER (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature
timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
symbol=<symbol>
orderId=<orderId>
api_url="https://api.binance.us"
parameter_concat="rail=$rail&symbol=$symbol&orderId=$orderId&timestamp=$timestamp"
```

```
signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "DELETE" "$api_url/sapi/v1/custodian/cancelOrder?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "symbol": "LTCBTC",
  "origClientOrderId": "myOrder1",
  "orderId": 4,
  "orderListId": -1, //Unless part of an OCO, the value will always be -1.
  "clientOrderId": "cancelMyOrder1",
  "price": "2.00000000",
  "origQty": "1.00000000",
  "executedQty": "0.00000000",
  "cummulativeQuoteQty": "0.00000000",
  "status": "CANCELED",
  "timeInForce": "GTC",
  "type": "LIMIT",
  "side": "BUY"
}
```


`DELETE /sapi/v1/custodian/cancelOrder (HMAC SHA256)`

Use this endpoint to cancel an active trade order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD).
orderId	LONG	NO	Either orderId or origClientOrderId must be sent.
origClientOrderId	STRING	NO	Either orderId or origClientOrderId must be sent.
newClientOrderId	STRING	NO	Used to uniquely identify this cancel. Automatically generated by default
timestamp	LONG	YES	

Data Source: Matching Engine

CANCEL OPEN ORDERS FOR SYMBOL (CUST.)

Example

```
#! /bin/bash
# Get HMAC SHA256 signature
timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
symbol=<symbol>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&symbol=$symbol&timestamp=$timestamp"
signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "DELETE" "$api_url/sapi/v1/custodian/cancelOrdersBySymbol?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
[
  {
    "symbol": "BTCUSDT",
    "origClientOrderId": "KZJijIsAFf5BU5oxkWTAU3",
    "orderId": 0,
    "orderListId": -1,
    "clientOrderId": "8epSIUMvNCknntXcSzWs7H",
    "price": "0.10000000",
    "origQty": "1.00000000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY"
  }
]
```


`DELETE /sapi/v1/custodian/cancelOrdersBySymbol (HMAC SHA256)`

Use this endpoint to cancel all active trade orders on a token symbol (this includes OCO orders).

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD).
timestamp	LONG	YES	

Data Source: Matching Engine

CANCEL OCO ORDER (CUST.)

Example

```
#! /bin/bash
```

```
# Get HMAC SHA256 signature
timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
symbol=<symbol>
orderListId=<orderListId>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&symbol=$symbol&orderListId=$orderListId&timestamp=$timestamp"
signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`
curl -X "DELETE" "$api_url/sapi/v1/custodian/cancelOcoOrder?$parameter_concat&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response


```
{
  "orderListId": 0,
  "contingencyType": "OCO",
  "listStatusType": "ALL_DONE",
  "listOrderStatus": "ALL_DONE",
  "listClientOrderId": "C3wyj4WVEktd7u9aVBRXcN",
  "transactionTime": 1574040868128,
  "symbol": "LTCBTC",
  "orders": [
    {
      "symbol": "LTCBTC",
      "orderId": 2,
      "clientOrderId": "pO9ufTiFGg3nw2fOdgeOXa"
    },
    {
      "symbol": "LTCBTC",
      "orderId": 3,
      "clientOrderId": "TXOvglzXuaubXAaENpaRCB"
    }
  ],
  "orderReports": [
    {
      "symbol": "LTCBTC",
      "origClientOrderId": "pO9ufTiFGg3nw2fOdgeOXa",
      "orderId": 2,
      "orderListId": 0,
      "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
      "price": "1.00000000",
      "origQty": "10.00000000",
      "executedQty": "0.00000000",
      "cummulativeQuoteQty": "0.00000000",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "STOP_LOSS_LIMIT",
      "side": "SELL",
      "stopPrice": "1.00000000"
    },
    {
      "symbol": "LTCBTC",
      "origClientOrderId": "TXOvglzXuaubXAaENpaRCB",
      "orderId": 3,
      "orderListId": 0,
      "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
      "price": "3.00000000",
      "origQty": "10.00000000",
      "executedQty": "0.00000000",
      "cummulativeQuoteQty": "0.00000000",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "LIMIT_MAKER",
      "side": "SELL"
    }
  ]
}
```


`DELETE /sapi/v1/custodian/cancelOcoOrder (HMAC SHA256)`

Use this endpoint to cancel an entire order list.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
symbol	STRING	YES	Order trading pair (e.g., BTCUSD, ETHUSD).
orderListId	LONG	YES	
listClientOrderId	STRING	NO	
newClientOrderId	STRING	NO	
timestamp	LONG	YES	

Data Source: Matching Engine

Settlement (Custodial)

GET SETTLEMENT SETTINGS

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
```

```
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/settlementSetting?$parameter_concat&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "settlementActive": true,
    "frequencyInHours": 24,
    "nextTriggerTime": 1646524800000
}
```

`GET /sapi/v1/custodian/settlementSetting (HMAC SHA256)`

Use this endpoint to get current settlement settings (status, schedule and next trigger time).

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).
timestamp	LONG	YES	Current timestamp.

Data Source: Database

GET SETTLEMENT HISTORY

Example

```
#! /bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`
api_key=<your_api_key>
secret_key=<your_secret_key>
rail=<rail>
api_url="https://api.binance.us"

parameter_concat="rail=$rail&timestamp=$timestamp"

signature=`echo -n $parameter_concat | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/custodian/settlementHistory?$parameter_concat&signature=$signature" \
     -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "records": [
        {
            "status": "PROCESS",
            "triggerTime": 1660074030009,
            "settlementId": "1a22baf2329845b997eff2bc9a700504",
            "settlementAssets": [
                {
                    "asset": "USDC_T",
                    "settleTo": "0xc80e62a4Bb86c42D1a0Cda3dCBcCb26830D92F2B",
                    "network": "ETH",
                    "amount": "10"
                }
            ]
        },
        {
            "status": "FAILURE",
            "triggerTime": 1659081889724,
            "settlementId": "d9e69bb7386f4ebaa1689559b549ef25",
            "settlementAssets": []
        },
        {
            "status": "FAILURE",
            "triggerTime": 1659079319905,
            "settlementId": "58ca799c596f43a792e8add27dc42d56",
            "settlementAssets": []
        },
        {
            "status": "FAILURE",
            "triggerTime": 1658981047109,
            "settlementId": "3ab30503bf6d43bc8ea8112f6f6b2c91",
            "settlementAssets": []
        }
    ],
    "total": 4
}
```

`GET /sapi/v1/custodian/settlementHistory (HMAC SHA256)`

Use this endpoint to check your settlement history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
rail	STRING	YES	Custodial partner (all uppercase).

Name	Type	Mandatory	Description
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	defaultValue:5, max:100
page	INT	NO	defaultValue:1

Data Source: Database

Credit Line Endpoints

Get Credit Line Account Information (C.L.)

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
api_url="https://api.binance.us"
secret_key=<your_secret_key>

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v2/cl/account?timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/json"
```

Response


```
{
    "clAccountId": 351010000,
    "masterAccountId": 351010001,
    "notificationMailAddr": "email@binance.us",
    "currentLTV": "0.415",
    "initialLTV": "0.62",
    "marginCallLTV": "0.8",
    "liquidationLTV": "0.54",
    "interestRate": "0.01",
    "liquidationFeeRate": "0.02",
    "contractStartTime": 1676332800000,
    "contractEndTime": 1680220802000,
    "isActive": true,
    "canTrade": true,
    "canTransferIn": true,
    "canTransferOut": true,
    "requiredDepositAmount": "0",
    "availableAmountToTransferOut": "500",
    "loanAssets": [
        {
            "loanAsset": "BNB",
            "loanQuantity": "900000",
            "currentLoanQuantity": "5885.70162259",
            "reclaimSequence": 1,
        }
    ],
    "balances": [
        {
            "asset": "BNB",
            "free": "14227.31994203",
            "locked": "0"
        },
        {
            "asset": "USD",
            "free": "948.46",
            "locked": "0"
        }
    ]
}
```


`GET /sapi/v2/cl/account`

Use this endpoint to get current credit line account information.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

Get Alert History (C.L.)

Example

```
timestamp=`date +%s000`

api_key=<your_api_key>
alertType=<alertType>
startTime=<startTime>
endTime=<endTime>
```

```
limit=<limit>
api_url="https://api.binance.us"

signature=`echo -n "alertType=$alertType&startTime=$startTime&endTime=$endTime&page=$page&limit=<limit>&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v2/cl/alertHistory?alertType=$alertType&startTime=$startTime&endTime=$endTime&page=$page&limit=<limit>&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
-H "Content-Type:application/json"
```

Response

```
[
    {
        "alertTime": 1672314507694,
        "alertType": "LIQUIDATION_CALL",
        "currentLTV": "9",
        "initialLTV": "0.62",
        "marginCallLTV": "0.8",
        "liquidationLTV": "0.54",
        "totalBalance": "3710355.24038655",
        "loanAssets": [
            {
                "loanAsset": "BNB",
                "loanQuantity": "900000",
                "currentLoanQuantity": "5885.70162259",
                "reclaimSequence": 1,
            }
        ],
        "balances": [
            {
                "asset": "BNB",
                "free": "14257.81994203",
                "locked": "0",
                "freeze": "0"
            },
            {
                "asset": "USD",
                "free": "48.46",
                "locked": "0",
                "freeze": "0"
            }
        ]
    },
    {
        "alertTime": 1671158880863,
        "alertType": "MARGIN_CALL",
        "currentLTV": "0.8124",
        "initialLTV": "0.62",
        "marginCallLTV": "0.8",
        "liquidationLTV": "0.54",
        "totalBalance": "5839625.1599878",
        "loanAssets": [
            {
                "loanAsset": "BNB",
                "loanQuantity": "900000",
                "currentLoanQuantity": "5885.70162259",
                "reclaimSequence": 1,
            }
        ],
        "balances": [
            {
                "asset": "BNB",
                "free": "14257.81994203",
                "locked": "0",
                "freeze": "0"
            },
            {
                "asset": "USD",
                "free": "48.46",
                "locked": "0",
                "freeze": "0"
            }
        ]
    }
]
```

`GET /sapi/v2/cl/alertHistory`

Use this endpoint to get your margin call and liquidation alert history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	defaultValue:200
alertType	ENUM	NO	AlertType(e.g., LIQUIDATION_CALL, MARGIN_CALL)
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

Get Transfer History (C.L.)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`
```

```
api_key=<your_api_key>
secret_key=<your_secret_key>

api_url="https://api.binance.us"

signature=`echo -n "timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/cl/transferHistory?timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
[
    {
        "transferId": "t11119",
        "transferType": "TRANSFER_IN",
        "asset": "USD",
        "amount": "10",
        "status": "SUCCESS",
        "transferTime": 1676958403384
    },
    {
        "transferId": "t11117",
        "transferType": "TRANSFER_IN",
        "asset": "USD",
        "amount": "10",
        "status": "SUCCESS",
        "transferTime": 1676676502389
    }
]
```

`GET /sapi/v1/cl/transferHistory`

Use this endpoint to get your transfer history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
startTime	LONG	NO	
endTime	LONG	NO	
limit	INT	NO	defaultValue:20, max:100
transferType	ENUM	NO	Transfer type (e.g., TRANSFER_IN, TRANSFER_OUT)
asset	STRING	NO	BTC,etc
recvWindow	LONG	NO	The value cannot be greater than `60000`
timestamp	LONG	YES	

Data Source: Database

Execute Transfer (C.L.)

Example

```
# Get HMAC SHA256 signature

timestamp=`date +%s000`

api_key=<your_api_key>
secret_key=<your_secret_key>

transferType=<transferType>
transferAssetType=<transferAssetType>
quantity=<quantity>

api_url="https://api.binance.us"

signature=`echo -n "transferType=$transferType&transferAssetType=$transferAssetType&quantity=$quantity&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "POST" "$api_url/sapi/v1/cl/transfer?transferType=$transferType&transferAssetType=$transferAssetType&quantity=$quantity&timestamp=$timestamp&signature=$signature" \
    -H "X-MBX-APIKEY: $api_key"
```

Response

```
{
    "transferId": 1331391,
    "status": "SUCCESS"
}
```

`POST /sapi/v1/cl/transfer`

Use this endpoint to transfer assets in or out of credit line account.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
transferType	ENUM	YES	Transfer type (e.g., TRANSFER_IN, TRANSFER_OUT)
transferAssetType	STRING	YES	Asset (e.g., BTC, USD)
quantity	DECIMAL	YES	amount of the asset to be transfered
recvWindow	LONG	NO	The value cannot be greater than `60000`

Name	Type	Mandatory	Description
timestamp	LONG	YES	

Data Source: Database

API Partner Endpoints

Check User Eligibility

Example

```
#!/bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

partner_id=<partner_id>
api_key=<your_api_key>
api_url="https://api.binance.us"
secret_key=<your_secret_key>

signature=`echo -n "partnerId=$partner_id&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/apipartner/checkEligibility?partnerId=$partner_id&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
```

Response

```
{
    "isEligible": false, // If the user is eligible for rebate (true or false)
    "ifNewUser": true, //true: new, false: old,
    "vipLevel": 1,
    "referrerId": 39472261
}
```

`GET /sapi/v1/apipartner/checkEligibility (HMAC SHA256)`

Use this endpoint to check if the user is eligible for rebate or not.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
partnerId	STRING	YES	8 character-long ID generated for API partner, e.g. "ABCD1234"
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Database

Get Rebate History

Example

```
#!/bin/bash
# Get HMAC SHA256 signature

timestamp=`date +%s000`

partner_id=<partner_id>
api_key=<your_api_key>
api_url="https://api.binance.us"
secret_key=<your_secret_key>

signature=`echo -n "partnerId=$partner_id&timestamp=$timestamp" | openssl dgst -sha256 -hmac $secret_key`

curl -X "GET" "$api_url/sapi/v1/rebateHistory/checkEligibility?partnerId=$partner_id&timestamp=$timestamp&signature=$signature" \
-H "X-MBX-APIKEY: $api_key" \
```

Response

```
[
    {
      "qty": "0.02063898",
      "asset": "BTC",
      "date": 1676585861493
    },
    {
      "qty": "1.2063898",
      "asset": "USDT",
      "date": 1676585861493
    }
]
```

`GET /sapi/v1/apipartner/rebateHistory (HMAC SHA256)`

Use this endpoint to query the user's rebate history.

Weight: 1

Parameters:

Name	Type	Mandatory	Description

Name	Type	Mandatory	Description
partnerId	STRING	YES	8 character-long ID generated for API partner, e.g. "ABCD1234"
startTime	LONG	NO	Default: 7 days before current time
endTime	LONG	NO	Default: present time
limit	INT	NO	Default: 100, max: 1000
page	INT	NO	
recvWindow	LONG	NO	
timestamp	LONG	YES	

Data Source: Database

Filters

Filters define trading rules for a symbol or an exchange.

Filters come in two forms: `symbol filters` and `exchange filters`.

Symbol Filters

PRICE FILTER

ExchangeInfo format:

```
{
  "filterType": "PRICE_FILTER",
  "minPrice": "0.00000100",
  "maxPrice": "100000.00000000",
  "tickSize": "0.00000100"
}
```

The `PRICE_FILTER` defines the `price` rules for a symbol. There are three parts:

- `minPrice` defines the minimum `price`/`stopPrice` allowed; disabled on `minPrice` == 0.
- `maxPrice` defines the maximum `price`/`stopPrice` allowed; disabled on `maxPrice` == 0.
- `tickSize` defines the intervals that a `price`/`stopPrice` can be increased/decreased by; disabled on `tickSize` == 0.

Any of the above variables can be set to 0, which disables that rule in the `price filter`. In order to pass the `price filter`, the following must be true for `price`/`stopPrice` of the enabled rules:

- `price` >= `minPrice`
- `price` <= `maxPrice`
- `price` % `tickSize` == 0

PERCENT PRICE FILTER

ExchangeInfo format:

```
{
  "filterType": "PERCENT_PRICE",
  "multiplierUp": "1.3000",
  "multiplierDown": "0.7000",
  "avgPriceMins": 5
}
```

The `PERCENT_PRICE` filter defines valid range for a price based on the average of the previous trades. `avgPriceMins` is the number of minutes the average price is calculated over. 0 means the last price is used.

In order to pass the `percent price`, the following must be true for `price`:

- `price` <= `weightedAveragePrice` * `multiplierUp`
- `price` >= `weightedAveragePrice` * `multiplierDown`

PERCENT PRICE BY SIDE FILTER

The `PERCENT_PRICE_BY_SIDE` filter defines the valid range for the price based on the average of the previous trades. `avgPriceMins` is the number of minutes the average price is calculated over. 0 means the last price is used.
There is a different range depending on whether the order is placed on the BUY side or the SELL side.

Buy orders will succeed on this filter if:

- `Order Price` <= `bidMultiplierUp` * `weightedAveragePrice`
- `Order Price` >= `bidMultiplierDown` * `weightedAveragePrice`

Sell orders will succeed on this filter if:

- `Order Price` <= `askMultiplierUp` * `weightedAveragePrice`
- `Order Price` >= `askMultiplierDown` * `weightedAveragePrice`

ExchangeInfo format:

```
{
  "filterType": "PERCENT_PRICE_BY_SIDE",
  "bidMultiplierUp": "1.2",
  "bidMultiplierDown": "0.2",
  "askMultiplierUp": "5",
  "askMultiplierDown": "0.8",
  "avgPriceMins": 1
}
```

LOT SIZE FILTER

ExchangeInfo format:

```
{
  "filterType": "LOT_SIZE",
```

```
  "minQty": "0.00100000",
  "maxQty": "100000.00000000",
  "stepSize": "0.00100000"
}
```

The `LOT_SIZE` filter defines the `quantity` (aka "lots" in auction terms) rules for a symbol. There are three 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` % `stepSize` == 0

NOTIONAL FILTER

> ExchangeInfo format:

```
{
  "filterType": "NOTIONAL",
  "minNotional": "10.00000000",
  "applyMinToMarket": false,
  "maxNotional": "10000.00000000",
  "applyMaxToMarket": false,
  "avgPriceMins": 5
}
```

The `NOTIONAL` filter defines the acceptable notional range allowed for an order on a symbol.

- `applyMinToMarket` determines whether the minNotional will be applied to `MARKET` orders.
- `applyMaxToMarket` determines whether the maxNotional will be applied to `MARKET` orders.

In order to pass this filter, the notional amount (`price` * `quantity`) has to meet the following conditions:

- `price` * `quantity` <= `maxNotional`
- `price` * `quantity` >= `minNotional`

For `MARKET` orders, the average price used over the last `avgPriceMins` minutes will be used for calculation. If the `avgPriceMins` is 0, then the last price will be used.

MIN NOTIONAL FILTER

> ExchangeInfo format:

```
{
  "filterType": "MIN_NOTIONAL",
  "minNotional": "0.00100000",
  "applyToMarket": true,
  "avgPriceMins": 5
}
```

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`.

`applyToMarket` determines whether or not the `MIN_NOTIONAL` filter will also be applied to `MARKET` orders. Since `MARKET` orders have no price, the average price is used over the last `avgPriceMins` minutes. `avgPriceMins` is the number of minutes the average price is calculated over. 0 means the last price is used.

ICEBERG PARTS FILTER

The `ICEBERG_PARTS` filter defines the maximum parts an iceberg order can have. The number of `ICEBERG_PARTS` is defined as `CEIL(qty / icebergQty)`.

> ExchangeInfo format:

```
{
  "filterType": "ICEBERG_PARTS",
  "limit": 10
}
```

MARKET ORDER LOT SIZE FILTER

> ExchangeInfo format:

```
{
  "filterType": "MARKET_LOT_SIZE",
  "minQty": "0.00100000",
  "maxQty": "100000.00000000",
  "stepSize": "0.00100000"
}
```

The `MARKET_LOT_SIZE` filter defines the `quantity` (aka "lots" in auction terms) rules for `MARKET` orders on a symbol. There are three parts:

- `minQty` defines the minimum `quantity` allowed;
- `maxQty` defines the maximum `quantity` allowed;
- `stepSize` defines the intervals that a `quantity` can be increased/decreased by.

In order to pass the `market lot size`, the following must be true for `quantity`:

- `quantity` >= `minQty`
- `quantity` <= `maxQty`
- `quantity` % `stepSize` == 0

SYMBOL MAX ORDERS FILTER

> ExchangeInfo format:

```
{
  "filterType": "MAX_NUM_ORDERS",
  "limit": 25
}
```

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.

SYMBOL MAX ALGO ORDERS FILTER

> ExchangeInfo format:

```
{
  "filterType": "MAX_NUM_ALGO_ORDERS",
```

```
  "maxNumAlgoOrders": 5
}
```

The `MAX_NUM_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.

MAX ICEBERG ORDERS FILTER

> ExchangeInfo format:

```
{
  "filterType": "MAX_NUM_ICEBERG_ORDERS",
  "maxNumIcebergOrders": 5
}
```

The `MAX_NUM_ICEBERG_ORDERS` filter defines the maximum number of `ICEBERG` orders an account is allowed to have open on a symbol. An `ICEBERG` order is any order where the `icebergQty` is > 0.

MAX POSITION FILTER

> ExchangeInfo format:

```
{
  "filterType":"MAX_POSITION",
  "maxPosition":"10.00000000"
}
```

The `MAX_POSITION` filter defines the allowed maximum position an account can have on the base asset of a symbol. An account's position is defined as the sum of the account's:

1. Free balance of the base asset
2. Locked balance of the base asset
3. Sum of the qty of all open BUY orders

`BUY` orders will be rejected if the account's position is greater than the maximum position allowed.

TRAILING DELTA FILTER

> ExchangeInfo format:

```
{
  "filterType": "TRAILING_DELTA",
  "minTrailingAboveDelta": 10,
  "maxTrailingAboveDelta": 2000,
  "minTrailingBelowDelta": 10,
  "maxTrailingBelowDelta": 2000
}
```

The `TRAILING_DELTA` filter defines the minimum and maximum value for the parameter `trailingDelta`.

In order for a `trailing stop order` to pass this filter, the following must be true:

For `STOP_LOSS_LIMIT BUY`,and `TAKE_PROFIT_LIMIT SELL` orders:

- `trailingDelta` >= `minTrailingAboveDelta`
- `trailingDelta` <= `maxTrailingAboveDelta`

For `STOP_LOSS_LIMIT SELL`,and `TAKE_PROFIT_LIMIT BUY` orders:

- `trailingDelta` >= `minTrailingBelowDelta`
- `trailingDelta` <= `maxTrailingBelowDelta`

Exchange Filters

EXCHANGE MAX ORDERS FILTER

> ExchangeInfo format:

```
{
  "filterType": "EXCHANGE_MAX_NUM_ORDERS",
  "maxNumOrders": 1000
}
```

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.

EXCHANGE MAX ALGO ORDERS FILTER

> ExchangeInfo format:

```
{
  "filterType": "EXCHANGE_MAX_ALGO_ORDERS",
  "maxNumAlgoOrders": 200
}
```

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.

EXCHANGE MAX NUM ICEBERG ORDERS

The `EXCHANGE_MAX_NUM_ICEBERG_ORDERS` filter defines the maximum number of iceberg orders an account is allowed to have open on the exchange.

> ExchangeInfo format:

```
{
  "filterType": "EXCHANGE_MAX_NUM_ICEBERG_ORDERS",
  "maxNumIcebergOrders": 10000
}
```

WebSocket API

General WebSocket API Information

- The base endpoint is: `wss://ws-api.binance.us:443/ws-api/v3`
    - If you experience issues with the standard 443 port, alternative port 9443 is also available.
- A single connection to the API is only valid for 24 hours; expect to be disconnected after the 24-hour mark.
- WebSocket server will send a **ping frame** every 3 minutes.
    - If the server does not receive a **pong frame** response within 10 minutes, you will be disconnected.
    - Unsolicited pong frames are allowed and will prevent disconnection.
- Lists are returned in **chronological order**, unless noted otherwise.
- All timestamps are in **milliseconds** in UTC, unless noted otherwise.
- All field names and values are **case-sensitive**, unless noted otherwise.

Request format

Example of request:

```
{
  "id": "e2a85d9f-07a5-4f94-8d5f-789dc3deb097",
  "method": "order.place",
  "params": {
    "symbol": "BTCUSDT",
    "side": "BUY",
    "type": "LIMIT",
    "price": "0.1",
    "quantity": "10",
    "timeInForce": "GTC",
    "timestamp": 1655716096498,
    "apiKey": "T59MTDLWlpRW16JVeZ2Nju5A5C98WkMm8CSzWC4oqynUlTm1zXOxyauT8LmwXEv9",
    "signature": "5942ad337e6779f2f4c62cd1c26dba71c91514400a24990a3e7f5edec9323f90"
  }
}
```

Requests must be sent as JSON in **text frames**, one request per frame.

Request fields:

Name	Type	Mandatory	Description
`id`	INT / STRING / `null`	YES	Arbitrary ID used to match responses to requests
`method`	STRING	YES	Request method name
`params`	OBJECT	NO	Request parameters. May be omitted if there are no parameters

- Request `id` is truly arbitrary. You can use UUIDs, sequential IDs, current timestamp, etc. The server does not interpret `id` in any way, simply echoing it back in the response.

You can freely reuse IDs within a session. However, be careful to not send more than one request at a time with the same ID, since otherwise it might be impossible to tell the responses apart.

- Request method names may be prefixed with explicit version: e.g., `"v3/order.place"`.
- The order of `params` is not significant.

Response format

Example of successful response:

```
{
  "id": "e2a85d9f-07a5-4f94-8d5f-789dc3deb097",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "orderId": 12510053279,
    "orderListId": -1,
    "clientOrderId": "a097fe6304b20a7e4fc436",
    "transactTime": 1655716096505,
    "price": "0.10000000",
    "origQty": "10.00000000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "workingTime": 1655716096505,
    "selfTradePreventionMode": "NONE"
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 12
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 4043
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 321
    }
  ]
}
```

Example of failed response:

```
{
  "id": "e2a85d9f-07a5-4f94-8d5f-789dc3deb097",
  "status": 400,
  "error": {
    "code": -2010,
    "msg": "Account has insufficient balance for requested action."
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 13
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 4044
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 322
    }
  ]
}
```

Responses are returned as JSON in **text frames**, one response per frame.

Response fields:

Name	Type	Mandatory	Description
`id`	INT / STRING / `null`	YES	Same as in the original request
`status`	INT	YES	Response status. See Status codes
`result`	OBJECT / ARRAY	YES	Response content. Present if request succeeded
`error`	OBJECT		Error description. Present if request failed
`rateLimits`	ARRAY	NO	Rate limiting status. See Rate limits(WebSocket)

STATUS CODES

Status codes in the `status` field are the same as in HTTP.

Here are some common status codes that you might encounter:

- `200` indicates a successful response.
- `4XX` status codes indicate invalid requests; the issue is on your side.
  - `400` – your request failed, see `error` for the reason.
  - `403` – you have been blocked by the Web Application Firewall.
  - `409` – your request partially failed but also partially succeeded, see `error` for details.
  - `418` – you have been auto-banned for repeated violation of rate limits.
  - `429` – you have exceeded the API request rate limit, please slow down.
- `5XX` status codes indicate internal errors; the issue is on Binance US' side.
  - **Important:** If a response contains 5xx status code, it **does not** necessarily mean that your request has failed. Execution status is *unknown* and the request might have actually succeeded. Please use query methods to confirm the status. You might also want to establish a new WebSocket connection for that.

See Error codes for a list of error codes and messages.

Request security

- Every method has a security type which determines how to call it.
  - Security type is stated below Data Source. For example, Place new order (WebSocket).
  - If no security type is stated, the security type is NONE.

Security type	API key	Signature	Description
`NONE`			Public market data
`TRADE`	required	required	Trading on the exchange, placing and canceling orders
`USER_DATA`	required	required	Private account information, such as order status and your trading history
`USER_STREAM`	required		Managing User Data Stream subscriptions
`MARKET_DATA`	required		Historical market data access

- Secure methods require a valid API key to be specified and authenticated.
  - API keys can be created on the **API Management** page of your Binance.US account.
  - **Both API key and secret key are sensitive.** Never share them with anyone. If you notice unusual activity in your account, immediately revoke all the keys and contact Binance.US support.
- API keys can be configured to allow access only to certain types of secure methods.
  - For example, you can have an API key with `TRADE` permission for trading, while using a separate API key with `USER_DATA` permission to monitor your order status.
  - By default, an API key cannot `TRADE`. You need to enable trading in API Management first.
- `TRADE` and `USER_DATA` requests are also known as `SIGNED` requests.

SIGNED (TRADE AND USER_DATA) REQUEST SECURITY

- `SIGNED` requests require an additional parameter: `signature`, authorizing the request.
- The signature is computed using HMAC-SHA-256 algorithm. See computation example below.

TIMING SECURITY

Request processing logic is as follows:

```
if (timestamp < (serverTime + 1000) && (serverTime - timestamp) <= recvWindow) {
  // process request
} else {
```

```
  // reject request
}
```

- `SIGNED` requests also require a `timestamp` parameter which should be the current millisecond timestamp.
- An additional optional parameter, `recvWindow`, specifies for how long the request stays valid.
  - If `recvWindow` is not sent, **it defaults to 5000 milliseconds**.
  - Maximum `recvWindow` is 60000 milliseconds.

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.

It is recommended to use a small `recvWindow` of 5000 or less!

SIGNED REQUEST EXAMPLE

Example of request:

```
{
  "id": "4885f793-e5ad-4c3b-8f6c-55d891472b71",
  "method": "order.place",
  "params": {
    "symbol":           "BTCUSDT",
    "side":             "SELL",
    "type":             "LIMIT",
    "timeInForce":      "GTC",
    "quantity":         "0.01000000",
    "price":            "52000.00",
    "newOrderRespType": "ACK",
    "recvWindow":       100,
    "timestamp":        1645423376532,
    "apiKey":           "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature":        "------ FILL ME ------"
  }
}
```

Here is a step-by-step guide on how to sign requests.

Example API key and secret key:

Key	Value
apiKey	`vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A`
secretKey	`NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j`

WARNING: DO NOT SHARE YOUR API KEY AND SECRET KEY WITH ANYONE.

The example keys are provided here only for illustrative purposes.

As you can see, the `signature` parameter is currently missing.

Step 1. Construct the signature payload

Take all request `params` except for the `signature`, sort them by name in alphabetical order:

Parameter	Value
apiKey	vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A
newOrderRespType	ACK
price	52000.00
quantity	0.01000000
recvWindow	100
side	SELL
symbol	BTCUSDT
timeInForce	GTC
timestamp	1645423376532
type	LIMIT

Format parameters as `parameter=value` pairs separated by `&`.

Resulting signature payload:

`apiKey=vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A&newOrderRespType=ACK&price=52000.00&quantity=0.01000000&recvWindow=100&side=SELL&symbol=BTCUSDT&timeInForce=GTC&timestamp=1645423376532&type=LIMIT`

Example of signature computation using OpenSSL:

```
$ echo -n 'apiKey=vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A&newOrderRespType=ACK&price=52000.00&quantity=0.01000000&recvWindow=100&side=SELL&symbol=BTCUSDT&timeInForce=GTC&timestamp=1645423376532&type=LIMIT' \
  | openssl dgst -hex -sha256 -hmac 'NhqPtmdSJYdKjVHjA7PZj4Mge3R5YNiP1e3UZjInClVN65XAbvqqM6A7H5fATj0j'

cc15477742bd704c29492d96c7ead9414dfd8e0ec4a00f947bb5bb454ddbd08a
```

Step 2. Compute the signature

1. Interpret `secretKey` as ASCII data, using it as a key for HMAC-SHA-256.
2. Sign signature payload as ASCII data.
3. Encode HMAC-SHA-256 output as a hex string.

Note that `apiKey`, `secretKey`, and the payload are **case-sensitive**, while resulting signature value is case-insensitive.

You can cross-check your signature algorithm implementation with OpenSSL.

Example of signed request:

```
{
  "id": "4885f793-e5ad-4c3b-8f6c-55d891472b71",
  "method": "order.place",
  "params": {
    "symbol":           "BTCUSDT",
```

```
    "side":             "SELL",
    "type":             "LIMIT",
    "timeInForce":      "GTC",
    "quantity":         "0.01000000",
    "price":            "52000.00",
    "newOrderRespType": "ACK",
    "recvWindow":       100,
    "timestamp":        1645423376532,
    "apiKey":           "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature":        "cc15477742bd704c29492d96c7ead9414dfd8e0ec4a00f947bb5bb454ddbd08a"
  }
}
```

Step 3. Add `signature` to request `params`

Finally, complete the request by adding the `signature` parameter with the signature string.

Data sources(WebSocket)

- The API system is asynchronous. Some delay in the response is normal and expected.
- Each method has a data source indicating where the data is coming from, and thus how up-to-date it is.

Data Source	Latency	Description
Matching Engine	lowest	The matching engine produces the response directly
Memory	low	Data is fetched from API server's local or external memory cache
Database	moderate	Data is retrieved from the database

- Some methods have more than one data source (e.g., Memory => Database).

This means that the API will look for the latest data in that order: first in the cache, then in the database.

Terminology

These terms will be used throughout the documentation, so it is recommended especially for new users to read to help their understanding of the API.

- `base asset` refers to the asset that is the `quantity` of a symbol. For the symbol BTCUSDT, BTC would be the `base asset`.
- `quote asset` refers to the asset that is the `price` of a symbol. For the symbol BTCUSDT, USDT would be the `quote asset`.

ENUM definitions(WebSocket)

Symbol status (status):

- `PRE_TRADING`
- `TRADING`
- `POST_TRADING`
- `END_OF_DAY`
- `HALT`
- `AUCTION_MATCH`
- `BREAK`

Account and Symbol Permissions (permissions):

- `SPOT`

Order status (status):

Status	Description
`NEW`	The order has been accepted by the engine.
`PARTIALLY_FILLED`	A part of the order has been filled.
`FILLED`	The order has been completed.
`CANCELED`	The order has been canceled by the user.
`PENDING_CANCEL`	Currently unused
`REJECTED`	The order was not accepted by the engine and not processed.
`EXPIRED`	The order was canceled according to the order type's rules (e.g. LIMIT FOK orders with no fill, LIMIT IOC or MARKET orders that partially fill) or by the exchange, (e.g. orders canceled during liquidation, orders canceled during maintenance)
`EXPIRED_IN_MATCH`	The order was expired by the exchange due to STP. (e.g. an order with `EXPIRE_TAKER` will match with existing orders on the book with the same account or same `tradeGroupId`)

OCO Status (listStatusType):

Status	Description
`RESPONSE`	This is used when the ListStatus is responding to a failed action. (E.g. Orderlist placement or cancellation)
`EXEC_STARTED`	The order list has been placed or there is an update to the order list status.
`ALL_DONE`	The order list has finished executing and thus no longer active.

OCO Order Status (listOrderStatus):

Status	Description
`EXECUTING`	Either an order list has been placed or there is an update to the status of the list.

Status	Description
`ALL_DONE`	An order list has completed execution and thus no longer active.
`REJECT`	The List Status is responding to a failed action either during order placement or order canceled

ContingencyType * `OCO`

Rate limits(WebSocket)

General information on rate limits

- Current API rate limits can be queried using the `exchangeInfo` request.
- There are multiple rate limit types across multiple intervals.
- Responses can indicate current rate limit status in the optional `rateLimits` field.
- Requests fail with status `429` when rate limits are violated.
- REST API and WebSocket API are subject to the same Rate Limit rules.

HOW TO INTERPRET RATE LIMITS

> A response with rate limit status may look like this:

```
{
  "id": "7069b743-f477-4ae3-81db-db9b8df085d2",
  "status": 200,
  "result": {
    "serverTime": 1656400526260
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 70
    }
  ]
}
```

The `rateLimits` array describes all currently active rate limits affected by the request.

Name	Type	Mandatory	Description
`rateLimitType`	ENUM	YES	Rate limit type: `REQUEST_WEIGHT`, `ORDERS`
`interval`	ENUM	YES	Rate limit interval: `SECOND`, `MINUTE`, `HOUR`, `DAY`
`intervalNum`	INT	YES	Rate limit interval multiplier
`limit`	INT	YES	Request limit per interval
`count`	INT	YES	Current usage per interval

Rate limits are accounted by intervals.

For example, a `1 MINUTE` interval starts every minute. Request submitted at 00:01:23.456 counts towards the 00:01:00 minute's limit. Once the 00:02:00 minute starts, the count will reset to zero again.

Other intervals behave in a similar manner. For example, `1 DAY` rate limit resets at 00:00 UTC every day, and `10 SECOND` interval resets at 00, 10, 20... seconds of each minute.

APIs have multiple rate-limiting intervals. If you exhaust a shorter interval but the longer interval still allows requests, you will have to wait for the shorter interval to expire and reset. If you exhaust a longer interval, you will have to wait for that interval to reset, even if shorter rate limit count is zero.

HOW TO SHOW/HIDE RATE LIMIT INFORMATION

> Default request and response:

```
{"id":1,"method":"time"}
```

```
{"id":1,"status":200,"result":{"serverTime":1656400526260},"rateLimits":[{"rateLimitType":"REQUEST_WEIGHT","interval":"MINUTE","intervalNum":1,"limit":1200,"count":70}]}
```

> Request and response without rate limit status:

```
{"id":2,"method":"time","params":{"returnRateLimits":false}}
```

```
{"id":2,"status":200,"result":{"serverTime":1656400527891}}
```

`rateLimits` field is included with every response by default.

However, rate limit information can be quite bulky. If you are not interested in detailed rate limit status of every request, the `rateLimits` field can be omitted from responses to reduce their size.

- Optional `returnRateLimits` boolean parameter in request.

Use `returnRateLimits` parameter to control whether to include `rateLimits` fields in response to individual requests.

- Optional `returnRateLimits` boolean parameter in connection URL.

If you wish to omit `rateLimits` from all responses by default, use `returnRateLimits` parameter in the query string instead: `wss://ws-api.binance.us/ws-api/v3?returnRateLimits=false`

This will make all requests made through this connection behave as if you have passed `"returnRateLimits": false`.

If you *want* to see rate limits for a particular request, you need to explicitly pass the `"returnRateLimits": true` parameter.

Note: Your requests are still rate limited if you hide the `rateLimits` field in responses.

IP limits(WebSocket)

- Every request has a certain **weight**, added to your limit as you perform requests.
  - Most requests cost 1 unit of weight, heavier requests acting on multiple symbols cost more.
  - Connecting to WebSocket API costs 1 weight.
- Current weight usage is indicated by the `REQUEST_WEIGHT` rate limit type.
- Use the `exchangeInfo` request to keep track of the current weight limits.
- Weight is accumulated **per IP address** and is shared by all connections from that address.
- If you go over the weight limit, requests fail with status `429`.
  - This status code indicates you should back off and stop spamming the API.
  - Rate-limited responses include a `retryAfter` field, indicating when you can retry the request.
- **Repeatedly violating rate limits and/or failing to back off after receiving 429s will result in an automated IP ban and you will be disconnected.**
  - Requests from a banned IP address fail with status `418`.
  - `retryAfter` field indicates the timestamp when the ban will be lifted.
- IP bans are tracked and **scale in duration** for repeat offenders, **from 2 minutes to 3 days**.

Successful response:

```
{
  "id": "7069b743-f477-4ae3-81db-db9b8df085d2",
  "status": 200,
  "result": [],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 70
    }
  ]
}
```

Successful response indicating that in 1 minute you have used 70 weight out of your 1200 limit.

Failed response:

```
{
  "id": "fc93a61a-a192-4cf4-bb2a-a8f0f0c51e06",
  "status": 418,
  "error": {
    "code": -1003,
    "msg": "Way too much request weight used; IP banned until 1659146400000. Please use WebSocket Streams for live updates to avoid bans.",
    "data": {
      "serverTime": 1659142907531,
      "retryAfter": 1659146400000
    }
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2411
    }
  ]
}
```

Failed response indicating that you are banned and the ban will last until epoch `1659146400000`.

Order rate limits(WebSocket)

- Every request to place an order counts towards your **order limit**.
  - Successfully placed orders update the `ORDERS` rate limit type.
  - Rejected or unsuccessful orders might or might not update the `ORDERS` count.
- Use the `account.rateLimits.orders` request to keep track of the current order rate limits.
- Order rate limit is maintained **per account** and is shared by all API keys of the account.
- If you go over the order rate limit, requests fail with status `429`.
  - This status code indicates you should back off and stop spamming the API.
  - Rate-limited responses include a `retryAfter` field, indicating when you can retry the request.

Successful response indicating that you have placed 12 orders in 10 seconds, and 4043 orders in the past 24 hours.

Successful response:

```
{
  "id": "e2a85d9f-07a5-4f94-8d5f-789dc3deb097",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "orderId": 12510053279,
    "orderListId": -1,
    "clientOrderId": "a097fe6304b20a7e4fc436",
    "transactTime": 1655716096505,
    "price": "0.10000000",
    "origQty": "10.00000000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "workingTime": 1655716096505,
    "selfTradePreventionMode": "NONE"
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 12
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 4043
    },
    {
```

```
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 321
    }
  ]
}
```

Public API requests

General requests

TEST CONNECTIVITY (WEBSOCKET)

Example

```
{
  "id": "922bcc6e-9de8-440d-9e84-7c80933a8d0d",
  "method": "ping"
}
```

Test connectivity to the WebSocket API.

Note: You can use regular WebSocket ping frames to test connectivity as well, WebSocket API will respond with pong frames as soon as possible. `ping` request along with `time` is a safe way to test request-response handling in your application.

Weight: 1

Parameters: NONE

Data Source: Memory

Response

```
{
  "id": "922bcc6e-9de8-440d-9e84-7c80933a8d0d",
  "status": 200,
  "result": {},
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

CHECK SERVER TIME (WEBSOCKET)

Example

```
{
  "id": "187d3cb2-942d-484c-8271-4e2141bbadb1",
  "method": "time"
}
```

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

Weight: 1

Parameters: NONE

Data Source: Memory

Response

```
{
  "id": "187d3cb2-942d-484c-8271-4e2141bbadb1",
  "status": 200,
  "result": {
    "serverTime": 1656400526260
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

EXCHANGE INFORMATION (WEBSOCKET)

Example

```
{
  "id": "5494febb-d167-46a2-996d-70533eb4d976",
  "method": "exchangeInfo",
  "params": {
    "symbols": ["BNBBTC"]
  }
}
```

Query current exchange trading rules, rate limits, and symbol information.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	NO	Describe a single symbol
`symbols`	ARRAY of STRING		Describe multiple symbols
`permissions`	ARRAY of STRING		Filter symbols by permissions

Notes:

- Only one of `symbol`, `symbols`, `permissions` parameters can be specified.
- `permissions` accepts either a list of permissions, or a single permission name: `"SPOT"`.
- Available Permissions

Data Source: Memory

Response


```
{
  "id": "5494febb-d167-46a2-996d-70533eb4d976",
  "status": 200,
  "result": {
    "timezone": "UTC",
    "serverTime": 1655969291181,
    // Global rate limits. See "Rate limits" section.
    "rateLimits": [
      {
        "rateLimitType": "REQUEST_WEIGHT",    // Rate limit type: REQUEST_WEIGHT, ORDERS, RAW_REQUESTS
        "interval": "MINUTE",                 // Rate limit interval: SECOND, MINUTE, DAY
        "intervalNum": 1,                     // Rate limit interval multiplier (i.e., "1 minute")
        "limit": 1200                         // Rate limit per interval
      },
      {
        "rateLimitType": "ORDERS",
        "interval": "SECOND",
        "intervalNum": 10,
        "limit": 50
      },
      {
        "rateLimitType": "ORDERS",
        "interval": "DAY",
        "intervalNum": 1,
        "limit": 160000
      },
      {
        "rateLimitType": "RAW_REQUESTS",
        "interval": "MINUTE",
        "intervalNum": 5,
        "limit": 6100
      }
    ],
    // Exchange filters are explained on the "Filters" page:
    // https://github.com/binance-us/binance-us-api-docs/blob/master/filters.md
    // All exchange filters are optional.
    "exchangeFilters": [],
    "symbols": [
      {
        "symbol": "BNBBTC",
        "status": "TRADING",
        "baseAsset": "BNB",
        "baseAssetPrecision": 8,
        "quoteAsset": "BTC",
        "quotePrecision": 8,
        "quoteAssetPrecision": 8,
        "baseCommissionPrecision": 8,
        "quoteCommissionPrecision": 8,
        "orderTypes": [
          "LIMIT",
          "LIMIT_MAKER",
          "MARKET",
          "STOP_LOSS_LIMIT",
          "TAKE_PROFIT_LIMIT"
        ],
        "icebergAllowed": true,
        "ocoAllowed": true,
        "quoteOrderQtyMarketAllowed": true,
        "allowTrailingStop": true,
        "cancelReplaceAllowed": true,
        "isSpotTradingAllowed": true,
        "isMarginTradingAllowed": true,
        // Symbol filters are explained on the "Filters" page:
        // https://github.com/binance-us/binance-us-api-docs/blob/master/filters.md
        // All symbol filters are optional.
        "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"
          }
        ],
        "permissions": [
          "SPOT"
        ]
      }
    ],
        "defaultSelfTradePreventionMode": "NONE",
        "allowedSelfTradePreventionModes": [
            "NONE"
        ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
```

```
}
```

Market data requests

ORDER BOOK (WEBSOCKET)

Example

```
{
  "id": "51e2affb-0aba-4821-ba75-f2625006eb43",
  "method": "depth",
  "params": {
    "symbol": "BNBBTC",
    "limit": 5
  }
}
```

Get current order book.

Note that this request returns limited market depth.

If you need to continuously monitor order book updates, please consider using WebSocket Streams:

- `<symbol>@depth<levels>`
- `<symbol>@depth`

You can use `depth` request together with `<symbol>@depth` streams to maintain a local order book.

Weight: Adjusted based on the limit:

Limit	Weight
1–100	1
101–500	5
501–1000	10
1001–5000	50

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`limit`	INT	NO	Default 100; max 5000

Data Source: Memory

Response

```
{
  "id": "51e2affb-0aba-4821-ba75-f2625006eb43",
  "status": 200,
  "result": {
    "lastUpdateId": 2731179239,
    // Bid levels are sorted from highest to lowest price.
    "bids": [
      [
        "0.01379900",   // Price
        "3.43200000"    // Quantity
      ],
      [
        "0.01379800",
        "3.24300000"
      ],
      [
        "0.01379700",
        "10.45500000"
      ],
      [
        "0.01379600",
        "3.82100000"
      ],
      [
        "0.01379500",
        "10.26200000"
      ]
    ],
    // Ask levels are sorted from lowest to highest price.
    "asks": [
      [
        "0.01380000",
        "5.91700000"
      ],
      [
        "0.01380100",
        "6.01400000"
      ],
      [
        "0.01380200",
        "0.26800000"
      ],
      [
        "0.01380300",
        "0.33800000"
      ],
      [
        "0.01380400",
        "0.26800000"
      ]
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
```

```
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

RECENT TRADES (WEBSOCKET)

Example

```
{
  "id": "409a20bd-253d-41db-a6dd-687862a5882f",
  "method": "trades.recent",
  "params": {
    "symbol": "BNBBTC",
    "limit": 1
  }
}
```

Get recent trades.

If you need access to real-time trading activity, please consider using WebSocket Streams:

- `<symbol>@trade`

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`limit`	INT	NO	Default 500; max 1000

Data Source: Memory

Response

```
{
  "id": "409a20bd-253d-41db-a6dd-687862a5882f",
  "status": 200,
  "result": [
    {
      "id": 194686783,
      "price": "0.01361000",
      "qty": "0.01400000",
      "quoteQty": "0.00019054",
      "time": 1660009530807,
      "isBuyerMaker": true,
      "isBestMatch": true
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

HISTORICAL TRADES (WEBSOCKET)

Example

```
{
  "id": "cffc9c7d-4efc-4ce0-b587-6b87448f052a",
  "method": "trades.historical",
  "params": {
    "symbol": "BNBBTC",
    "fromId": 0,
    "limit": 1,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A"
  }
}
```

Get historical trades.

Weight: 5

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`fromId`	INT	NO	Trade ID to begin at
`limit`	INT	NO	Default 500; max 1000
`apiKey`	STRING	YES	

Notes:

- If `fromId` is not specified, the most recent trades are returned.

Data Source: Database

Security Type: MARKET_DATA

Response

```
{
  "id": "cffc9c7d-4efc-4ce0-b587-6b87448f052a",
  "status": 200,
```

```
  "result": [
    {
      "id": 0,
      "price": "0.00005000",
      "qty": "40.00000000",
      "quoteQty": "0.00200000",
      "time": 1500004800376,
      "isBuyerMaker": true,
      "isBestMatch": true
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 5
    }
  ]
}
```


AGGREGATE TRADES (WEBSOCKET)

Example

```
{
  "id": "189da436-d4bd-48ca-9f95-9f613d621717",
  "method": "trades.aggregate",
  "params": {
    "symbol": "BNBBTC",
    "fromId": 50000000,
    "limit": 1
  }
}
```

Get aggregate trades.

An *aggregate trade* (aggtrade) represents one or more individual trades. Trades that fill at the same time, from the same taker order, with the same price – those trades are collected into an aggregate trade with total quantity of the individual trades.

If you need access to real-time trading activity, please consider using WebSocket Streams:

- `<symbol>@aggTrade`

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`fromId`	INT	NO	Aggregate trade ID to begin at
`startTime`	INT	NO	
`endTime`	INT	NO	
`limit`	INT	NO	Default 500; max 1000

Notes:

- If `fromId` is specified, return aggtrades with aggregate trade ID >= `fromId`.

Use `fromId` and `limit` to page through all aggtrades.

- If `startTime` and/or `endTime` are specified, aggtrades are filtered by execution time (`T`).

`fromId` cannot be used together with `startTime` and `endTime`.

- If no condition is specified, the most recent aggregate trades are returned.

Data Source: Database

Response


```
{
  "id": "189da436-d4bd-48ca-9f95-9f613d621717",
  "status": 200,
  "result": [
    {
      "a": 50000000,        // Aggregate trade ID
      "p": "0.00274100",    // Price
      "q": "57.19000000",   // Quantity
      "f": 59120167,        // First trade ID
      "l": 59120170,        // Last trade ID
      "T": 1565877971222,   // Timestamp
      "m": true,            // Was the buyer the maker?
      "M": true             // Was the trade the best price match?
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```


KLINES (WEBSOCKET)

Example

```
{
  "id": "1dbbeb56-8eea-466a-8f6e-86bdcfa2fc0b",
  "method": "klines",
  "params": {
    "symbol": "BNBBTC",
```

```
    "interval": "1h",
    "startTime": 1655969280000,
    "limit": 1
  }
}
```

Get klines (candlestick bars).

Klines are uniquely identified by their open & close time.

If you need access to real-time kline updates, please consider using WebSocket Streams:

- `<symbol>@kline_<interval>`

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`interval`	ENUM	YES	
`startTime`	INT	NO	
`endTime`	INT	NO	
`limit`	INT	NO	Default 500; max 1000

Supported kline intervals (case-sensitive):

Interval	`interval` value
minutes	`1m`, `3m`, `5m`, `15m`, `30m`
hours	`1h`, `2h`, `4h`, `6h`, `8h`, `12h`
days	`1d`, `3d`
weeks	`1w`
months	`1M`

Notes:

- If `startTime`, `endTime` are not specified, the most recent klines are returned.

Data Source: Database

Response

```
{
  "id": "1dbbeb56-8eea-466a-8f6e-86bdcfa2fc0b",
  "status": 200,
  "result": [
    [
      1655971200000,      // Kline open time
      "0.01086000",       // Open price
      "0.01086600",       // High price
      "0.01083600",       // Low price
      "0.01083800",       // Close price
      "2290.53800000",    // Volume
      1655974799999,      // Kline close time
      "24.85074442",      // Quote asset volume
      2283,               // Number of trades
      "1171.64000000",    // Taker buy base asset volume
      "12.71225884",      // Taker buy quote asset volume
      "0"                 // Unused field, ignore
    ]
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

CURRENT AVERAGE PRICE (WEBSOCKET)

Example

```
{
  "id": "ddbfb65f-9ebf-42ec-8240-8f0f91de0867",
  "method": "avgPrice",
  "params": {
    "symbol": "BNBBTC"
  }
}
```

Get current average price for a symbol.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	

Data Source: Memory

Response

```json
{
  "id": "ddbfb65f-9ebf-42ec-8240-8f0f91de0867",
  "status": 200,
  "result": {
    "mins": 5,              // Price averaging interval in minutes
    "price": "0.01378135"
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

24HR TICKER PRICE CHANGE STATISTICS (WEBSOCKET)

Example

```json
{
  "id": "93fb61ef-89f8-4d6e-b022-4f035a3fadad",
  "method": "ticker.24hr",
  "params": {
    "symbol": "BNBBTC"
  }
}
```

Get 24-hour rolling window price change statistics.

If you need to continuously monitor trading statistics, please consider using WebSocket Streams:

- `<symbol>@ticker` or `!ticker@arr`
- `<symbol>@miniTicker` or `!miniTicker@arr`

If you need different window sizes, use the `ticker` request.

Weight: Adjusted based on the number of requested symbols:

Symbols	Weight
1–20	1
21–100	20
101 or more	40
all symbols	40

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	NO	Query ticker for a single symbol
`symbols`	ARRAY of STRING		Query ticker for multiple symbols
`type`	ENUM	NO	Ticker type: `FULL` (default) or `MINI`

Notes:

- `symbol` and `symbols` cannot be used together.
- If no symbol is specified, returns information about all symbols currently trading on the exchange.

Data Source: Memory

Response

`FULL` type, for a single symbol:

```json
{
  "id": "93fb61ef-89f8-4d6e-b022-4f035a3fadad",
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "priceChange": "0.00013900",
    "priceChangePercent": "1.020",
    "weightedAvgPrice": "0.01382453",
    "prevClosePrice": "0.01362800",
    "lastPrice": "0.01376700",
    "lastQty": "1.78800000",
    "bidPrice": "0.01376700",
    "bidQty": "4.64600000",
    "askPrice": "0.01376800",
    "askQty": "14.31400000",
    "openPrice": "0.01362800",
    "highPrice": "0.01414900",
    "lowPrice": "0.01346600",
    "volume": "69412.40500000",
    "quoteVolume": "959.59411487",
    "openTime": 1660014164909,
    "closeTime": 1660100564909,
    "firstId": 194696115,       // First trade ID
    "lastId": 194968287,        // Last trade ID
    "count": 272173             // Number of trades
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

`MINI` type, for a single symbol:

```
{
  "id": "9fa2a91b-3fca-4ed7-a9ad-58e3b67483de",
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "openPrice": "0.01362800",
    "highPrice": "0.01414900",
    "lowPrice": "0.01346600",
    "lastPrice": "0.01376700",
    "volume": "69412.40500000",
    "quoteVolume": "959.59411487",
    "openTime": 1660014164909,
    "closeTime": 1660100564909,
    "firstId": 194696115,       // First trade ID
    "lastId": 194968287,        // Last trade ID
    "count": 272173             // Number of trades
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

If more than one symbol is requested, response returns an array:

```
{
  "id": "901be0d9-fd3b-45e4-acd6-10c580d03430",
  "status": 200,
  "result": [
    {
      "symbol": "BNBBTC",
      "priceChange": "0.00016500",
      "priceChangePercent": "1.213",
      "weightedAvgPrice": "0.01382508",
      "prevClosePrice": "0.01360800",
      "lastPrice": "0.01377200",
      "lastQty": "1.01400000",
      "bidPrice": "0.01377100",
      "bidQty": "7.55700000",
      "askPrice": "0.01377200",
      "askQty": "4.37900000",
      "openPrice": "0.01360700",
      "highPrice": "0.01414900",
      "lowPrice": "0.01346600",
      "volume": "69376.27900000",
      "quoteVolume": "959.13277091",
      "openTime": 1660014615517,
      "closeTime": 1660101015517,
      "firstId": 194697254,
      "lastId": 194969483,
      "count": 272230
    },
    {
      "symbol": "BTCUSDT",
      "priceChange": "-938.06000000",
      "priceChangePercent": "-3.938",
      "weightedAvgPrice": "23265.34432003",
      "prevClosePrice": "23819.17000000",
      "lastPrice": "22880.91000000",
      "lastQty": "0.00536000",
      "bidPrice": "22880.40000000",
      "bidQty": "0.00424000",
      "askPrice": "22880.91000000",
      "askQty": "0.04276000",
      "openPrice": "23818.97000000",
      "highPrice": "23933.25000000",
      "lowPrice": "22664.69000000",
      "volume": "153508.37606000",
      "quoteVolume": "3571425225.04441220",
      "openTime": 1660014615977,
      "closeTime": 1660101015977,
      "firstId": 1592019902,
      "lastId": 1597301762,
      "count": 5281861
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

ROLLING WINDOW PRICE CHANGE STATISTICS (WEBSOCKET)

Example

```
{
  "id": "f4b3b507-c8f2-442a-81a6-b2f12daa030f",
  "method": "ticker",
  "params": {
    "symbols": [
      "BNBBTC",
      "BTCUSDT"
    ],
    "windowSize": "7d"
  }
}
```

Get rolling window price change statistics with a custom window.

This request is similar to `ticker.24hr`, but statistics are computed on demand using the arbitrary window you specify.

Note: Window size precision is limited to 1 minute. While the `closeTime` is the current time of the request, `openTime` always start on a minute boundary. As such, the effective window might be up to 59999 ms wider than the requested `windowSize`.

Window computation example

For example, a request for `"windowSize": "7d"` might result in the following window:

`"openTime": 1659580020000, "closeTime": 1660184865291,`

Time of the request – `closeTime` – is 1660184865291 (August 11, 2022 02:27:45.291). Requested window size should put the `openTime` 7 days before that – August 4, 02:27:45.291 – but due to limited precision it ends up a bit earlier: 1659580020000 (August 4, 2022 02:27:00), exactly at the start of a minute.

If you need to continuously monitor trading statistics, please consider using WebSocket Streams:

- `<symbol>@ticker_<window_size>` or `!ticker_<window-size>@arr`

Weight: Adjusted based on the number of requested symbols:

Symbols	Weight
1–50	2 per symbol
51–100	100

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	Query ticker of a single symbol
`symbols`	ARRAY of STRING		Query ticker for multiple symbols
`type`	ENUM	NO	Ticker type: `FULL` (default) or `MINI`
`windowSize`	ENUM	NO	Default `1d`

Supported window sizes:

Unit	`windowSize` value
minutes	`1m`, `2m` … `59m`
hours	`1h`, `2h` … `23h`
days	`1d`, `2d` … `7d`

Notes:

- Either `symbol` or `symbols` must be specified.
- Maximum number of symbols in one request: 100.
- Window size units cannot be combined. E.g., `1d 2h` is not supported.

Data Source: Database

Response

`FULL` type, for a single symbol:

```
{
  "id": "f4b3b507-c8f2-442a-81a6-b2f12daa030f",
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "priceChange": "0.00061500",
    "priceChangePercent": "4.735",
    "weightedAvgPrice": "0.01368242",
    "openPrice": "0.01298900",
    "highPrice": "0.01418800",
    "lowPrice": "0.01296000",
    "lastPrice": "0.01360400",
    "volume": "587179.23900000",
    "quoteVolume": "8034.03382165",
    "openTime": 1659580020000,
    "closeTime": 1660184865291,
    "firstId": 192977765,       // First trade ID
    "lastId": 195365758,        // Last trade ID
    "count": 2387994            // Number of trades
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

`MINI` type, for a single symbol:

```
{
  "id": "bdb7c503-542c-495c-b797-4d2ee2e91173",
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "openPrice": "0.01298900",
    "highPrice": "0.01418800",
    "lowPrice": "0.01296000",
    "lastPrice": "0.01360400",
    "volume": "587179.23900000",
    "quoteVolume": "8034.03382165",
    "openTime": 1659580020000,
    "closeTime": 1660184865291,
    "firstId": 192977765,       // First trade ID
    "lastId": 195365758,        // Last trade ID
    "count": 2387994            // Number of trades
  },
  "rateLimits": [
```

```
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

If more than one symbol is requested, response returns an array:

```
{
  "id": "f4b3b507-c8f2-442a-81a6-b2f12daa030f",
  "status": 200,
  "result": [
    {
      "symbol": "BNBBTC",
      "priceChange": "0.00061500",
      "priceChangePercent": "4.735",
      "weightedAvgPrice": "0.01368242",
      "openPrice": "0.01298900",
      "highPrice": "0.01418800",
      "lowPrice": "0.01296000",
      "lastPrice": "0.01360400",
      "volume": "587169.48600000",
      "quoteVolume": "8033.90114517",
      "openTime": 1659580020000,
      "closeTime": 1660184820927,
      "firstId": 192977765,
      "lastId": 195365700,
      "count": 2387936
    },
    {
      "symbol": "BTCUSDT",
      "priceChange": "1182.92000000",
      "priceChangePercent": "5.113",
      "weightedAvgPrice": "23349.27074846",
      "openPrice": "23135.33000000",
      "highPrice": "24491.22000000",
      "lowPrice": "22400.00000000",
      "lastPrice": "24318.25000000",
      "volume": "1039498.10978000",
      "quoteVolume": "24271522807.76838630",
      "openTime": 1659580020000,
      "closeTime": 1660184820927,
      "firstId": 1568787779,
      "lastId": 1604337406,
      "count": 35549628
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 4
    }
  ]
}
```

SYMBOL PRICE TICKER (WEBSOCKET)

Example

```
{
  "id": "043a7cf2-bde3-4888-9604-c8ac41fcba4d",
  "method": "ticker.price",
  "params": {
    "symbol": "BNBBTC"
  }
}
```

Get the latest market price for a symbol.

If you need access to real-time price updates, please consider using WebSocket Streams:

- `<symbol>@aggTrade`
- `<symbol>@trade`

Weight: Adjusted based on the number of requested symbols:

Parameter	Weight
`symbol`	1
`symbols`	2
none	2

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	NO	Query price for a single symbol
`symbols`	ARRAY of STRING		Query price for multiple symbols

Notes:

- `symbol` and `symbols` cannot be used together.
- If no symbol is specified, returns information about all symbols currently trading on the exchange.

Data Source: Memory

Response

```
{
  "id": "043a7cf2-bde3-4888-9604-c8ac41fcba4d",
```

```
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "price": "0.01361900"
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

If more than one symbol is requested, response returns an array:

```
{
  "id": "e739e673-24c8-4adf-9cfa-b81f30330b09",
  "status": 200,
  "result": [
    {
      "symbol": "BNBBTC",
      "price": "0.01363700"
    },
    {
      "symbol": "BTCUSDT",
      "price": "24267.15000000"
    },
    {
      "symbol": "BNBBUSD",
      "price": "331.10000000"
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

SYMBOL ORDER BOOK TICKER (WEBSOCKET)

Example

```
{
  "id": "057deb3a-2990-41d1-b58b-98ea0f09e1b4",
  "method": "ticker.book",
  "params": {
    "symbols": [
      "BNBBTC",
      "BTCUSDT"
    ]
  }
}
```

Get the current best price and quantity on the order book.

If you need access to real-time order book ticker updates, please consider using WebSocket Streams:

- `<symbol>@bookTicker`

Weight: Adjusted based on the number of requested symbols:

Parameter	Weight
`symbol`	1
`symbols`	2
none	2

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	NO	Query ticker for a single symbol
`symbols`	ARRAY of STRING		Query ticker for multiple symbols

Notes:

- `symbol` and `symbols` cannot be used together.
- If no symbol is specified, returns information about all symbols currently trading on the exchange.

Data Source: Memory

Response

```
{
  "id": "9d32157c-a556-4d27-9866-66760a174b57",
  "status": 200,
  "result": {
    "symbol": "BNBBTC",
    "bidPrice": "0.01358000",
    "bidQty": "12.53400000",
    "askPrice": "0.01358100",
    "askQty": "17.83700000"
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
```

```
      "count": 1
    }
  ]
}
```

If more than one symbol is requested, response returns an array:

```
{
  "id": "057deb3a-2990-41d1-b58b-98ea0f09e1b4",
  "status": 200,
  "result": [
    {
      "symbol": "BNBBTC",
      "bidPrice": "0.01358000",
      "bidQty": "12.53400000",
      "askPrice": "0.01358100",
      "askQty": "17.83700000"
    },
    {
      "symbol": "BTCUSDT",
      "bidPrice": "23980.49000000",
      "bidQty": "0.01000000",
      "askPrice": "23981.31000000",
      "askQty": "0.01512000"
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

Trading requests

PLACE NEW ORDER (WEBSOCKET)

Example

```
{
  "id": "56374a46-3061-486b-a311-99ee972eb648",
  "method": "order.place",
  "params": {
    "symbol": "BTCUSDT",
    "side": "SELL",
    "type": "LIMIT",
    "timeInForce": "GTC",
    "price": "23416.10000000",
    "quantity": "0.00847000",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "15af09e41c36f3cc61378c2fbe2c33719a03dd5eba8d0f9206fbda44de717c88",
    "timestamp": 1660801715431
  }
}
```

Send in a new order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`side`	ENUM	YES	`BUY` or `SELL`
`type`	ENUM	YES	
`timeInForce`	ENUM	NO *	
`price`	DECIMAL	NO *	
`quantity`	DECIMAL	NO *	
`quoteOrderQty`	DECIMAL	NO *	
`newClientOrderId`	STRING	NO	Arbitrary unique ID among open orders. Automatically generated if not sent. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: “ABCD1234-…”.
`newOrderRespType`	ENUM	NO	Select response format: `ACK`, `RESULT`, `FULL`. `MARKET` and `LIMIT` orders use `FULL` by default, other order types default to `ACK`.
`stopPrice`	DECIMAL	NO *	
`trailingDelta`	INT	NO *	See Trailing Stop order FAQ
`icebergQty`	DECIMAL	NO	
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	

Name	Type	Mandatory	Description
`timestamp`	INT	YES	

Certain parameters (*) become mandatory based on the order `type`:

Order `type`	Mandatory parameters
`LIMIT`	• `timeInForce` • `price` • `quantity`
`LIMIT_MAKER`	• `price` • `quantity`
`MARKET`	• `quantity` or `quoteOrderQty`
`STOP_LOSS`	• `quantity` • `stopPrice` or `trailingDelta`
`STOP_LOSS_LIMIT`	• `timeInForce` • `price` • `quantity` • `stopPrice` or `trailingDelta`
`TAKE_PROFIT`	• `quantity` • `stopPrice` or `trailingDelta`
`TAKE_PROFIT_LIMIT`	• `timeInForce` • `price` • `quantity` • `stopPrice` or `trailingDelta`

Supported order types:

Order `type`	Description
`LIMIT`	Buy or sell `quantity` at the specified `price` or better.
`LIMIT_MAKER`	`LIMIT` order that will be rejected if it immediately matches and trades as a taker. This order type is also known as a POST-ONLY order.
`MARKET`	Buy or sell at the best available market price. • `MARKET` order with `quantity` parameter specifies the amount of the *base asset* you want to buy or sell. Actually executed quantity of the quote asset will be determined by available market liquidity. E.g., a MARKET BUY order on BTCUSDT for `"quantity": "0.1000"` specifies that you want to buy 0.1 BTC at the best available price. If there is not enough BTC at the best price, keep buying at the next best price, until either your order is filled, or you run out of USDT, or market runs out of BTC. • `MARKET` order with `quoteOrderQty` parameter specifies the amount of the *quote asset* you want to spend (when buying) or receive (when selling). Actually executed quantity of the base asset will be determined by available market liquidity. E.g., a MARKET BUY on BTCUSDT for `"quoteOrderQty": "100.00"` specifies that you want to buy as much BTC as you can for 100 USDT at the best available price. Similarly, a SELL order will sell as much available BTC as needed for you to receive 100 USDT (before commission).
`STOP_LOSS`	Execute a `MARKET` order for given `quantity` when specified conditions are met. I.e., when `stopPrice` is reached, or when `trailingDelta` is activated.
`STOP_LOSS_LIMIT`	Place a `LIMIT` order with given parameters when specified conditions are met.
`TAKE_PROFIT`	Like `STOP_LOSS` but activates when market price moves in the favorable direction.
`TAKE_PROFIT_LIMIT`	Like `STOP_LOSS_LIMIT` but activates when market price moves in the favorable direction.

Available `timeInForce` options, setting how long the order should be active before expiration:

TIF	Description
`GTC`	**Good 'til Canceled** – the order will remain on the book until you cancel it, or the order is completely filled.
`IOC`	**Immediate or Cancel** – the order will be filled for as much as possible, the unfilled quantity immediately expires.
`FOK`	**Fill or Kill** – the order will expire unless it cannot be immediately filled for the entire quantity.

Notes:

- `newClientOrderId` specifies `clientOrderId` value for the order.

A new order with the same `clientOrderId` is accepted only when the previous one is filled or expired.

- Any `LIMIT` or `LIMIT_MAKER` order can be made into an iceberg order by specifying the `icebergQty`.

An order with an `icebergQty` must have `timeInForce` set to `GTC`.

- Trigger order price rules for `STOP_LOSS`/`TAKE_PROFIT` orders:
  - `stopPrice` must be above market price: `STOP_LOSS BUY`, `TAKE_PROFIT SELL`
  - `stopPrice` must be below market price: `STOP_LOSS SELL`, `TAKE_PROFIT BUY`

- `MARKET` orders using `quoteOrderQty` follow `LOT_SIZE` filter rules.

The order will execute a quantity that has notional value as close as possible to requested `quoteOrderQty`.

Data Source: Matching Engine

Security Type: TRADE

Response

Response format is selected by using the `newOrderRespType` parameter.

`ACK` response type:

```
{
  "id": "56374a46-3061-486b-a311-99ee972eb648",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "orderId": 12569099453,
    "orderListId": -1, // always -1 for singular orders
    "clientOrderId": "4d96324ff9d44481926157ec08158a40",
    "transactTime": 1660801715639
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

`RESULT` response type:

```
{
  "id": "56374a46-3061-486b-a311-99ee972eb648",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "orderId": 12569099453,
    "orderListId": -1, // always -1 for singular orders
    "clientOrderId": "4d96324ff9d44481926157ec08158a40",
    "transactTime": 1660801715639,
    "price": "23416.10000000",
    "origQty": "0.00847000",
    "executedQty": "0.00000000",
    "cummulativeQuoteQty": "0.00000000",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "SELL",
    "stopPrice": "23500.00000000",      // present only if stopPrice set for the order
    "trailingDelta": 10,                // present only if trailingDelta set for the order
    "trailingTime": -1,                 // present only if trailingDelta set for the order
    "icebergQty": "0.00000000",         // present only if icebergQty set for the order
    "workingTime": 1660801715639,
    "selfTradePreventionMode": "NONE"
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

`FULL` response type:

```
{
  "id": "56374a46-3061-486b-a311-99ee972eb648",
  "status": 200,
  "result": {
```

```
    "symbol": "BTCUSDT",
    "orderId": 12569099453,
    "orderListId": -1,
    "clientOrderId": "4d96324ff9d44481926157ec08158a40",
    "transactTime": 1660801715793,
    "price": "23416.10000000",
    "origQty": "0.00847000",
    "executedQty": "0.00847000",
    "cummulativeQuoteQty": "198.33521500",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "SELL",
    "workingTime": 1660801715793,
    // FULL response is identical to RESULT response, with the same optional fields
    // based on the order type and parameters. FULL response additionally includes
    // the list of trades which immediately filled the order.
    "fills": [
      {
        "price": "23416.10000000",
        "qty": "0.00635000",
        "commission": "0.000000",
        "commissionAsset": "BNB",
        "tradeId": 1650422481
      },
      {
        "price": "23416.50000000",
        "qty": "0.00212000",
        "commission": "0.000000",
        "commissionAsset": "BNB",
        "tradeId": 1650422482
      }
    ],
    "selfTradePreventionMode": "NONE"
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

TEST NEW ORDER (WEBSOCKET)

Example

```
{
  "id": "6ffebe91-01d9-43ac-be99-57cf062e0e30",
  "method": "order.test",
  "params": {
    "symbol": "BTCUSDT",
    "side": "SELL",
    "type": "LIMIT",
    "timeInForce": "GTC",
    "price": "23416.10000000",
    "quantity": "0.00847000",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "15af09e41c36f3cc61378c2fbe2c33719a03dd5eba8d0f9206fbda44de717c88",
    "timestamp": 1660801715431
  }
}
```

Test order placement.

Validates new order parameters and verifies your signature but does not send the order into the matching engine.

Weight: 1

Parameters:

Same as for `order.place`.

Data Source: Memory

Security Type: TRADE

Response

```
{
  "id": "6ffebe91-01d9-43ac-be99-57cf062e0e30",
  "status": 200,
  "result": {},
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

QUERY ORDER (WEBSOCKET)

Example

```
{
  "id": "aa62318a-5a97-4f3b-bdc7-640bbe33b291",
  "method": "order.status",
  "params": {
    "symbol": "BTCUSDT",
    "orderId": 12569099453,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "2c3aab5a078ee4ea465ecd95523b77289f61476c2f238ec10c55ea6cb11a6f35",
    "timestamp": 1660801720951
  }
}
```

Check execution status of an order.

Weight: 2

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`orderId`	INT	YES	Lookup order by `orderId`
`origClientOrderId`	STRING		Lookup order by `clientOrderId`
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If both `orderId` and `origClientOrderId` parameters are specified, only `orderId` is used and `origClientOrderId` is ignored.
- For some historical orders the `cummulativeQuoteQty` response field may be negative, meaning the data is not available at this time.

Data Source: Memory => Database

Security Type: USER_DATA

Response

```
{
  "id": "aa62318a-5a97-4f3b-bdc7-640bbe33b291",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "orderId": 12569099453,
    "orderListId": -1,                  // set only for legs of an OCO
    "clientOrderId": "4d96324ff9d44481926157",
    "price": "23416.10000000",
    "origQty": "0.00847000",
    "executedQty": "0.00847000",
    "cummulativeQuoteQty": "198.33521500",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "SELL",
    "stopPrice": "0.00000000",          // always present, zero if order type does not use stopPrice
    "trailingDelta": 10,                // present only if trailingDelta set for the order
    "trailingTime": -1,                 // present only if trailingDelta set for the order
    "icebergQty": "0.00000000",         // always present, zero for non-iceberg orders
    "time": 1660801715639,              // time when the order was placed
    "updateTime": 1660801717945,        // time of the last update to the order
    "isWorking": true,
    "workingTime": 1660801715639,
    "origQuoteOrderQty": "0.00000000",  // always present, zero if order type does not use quoteOrderQty
    "selfTradePreventionMode": "NONE"
    "preventedMatchId": 0,              // present only if the order expired due to STP
    "preventedQuantity": "1.200000"     // present only if the order expired due to STP
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

CANCEL ORDER (WEBSOCKET)

Example

```
{
  "id": "5633b6a2-90a9-4192-83e7-925c90b6a2fd",
  "method": "order.cancel",
  "params": {
    "symbol": "BTCUSDT",
    "origClientOrderId": "4d96324ff9d44481926157",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "33d5b721f278ae17a52f004a82a6f68a70c68e7dd6776ed0be77a455ab855282",
    "timestamp": 1660801715830
  }
}
```

Cancel an active order.

Weight: 1

Parameters:

Name	Type	Mandatory	Description

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`orderId`	INT	YES	Cancel order by `orderId`
`origClientOrderId`	STRING		Cancel order by `clientOrderId`
`newClientOrderId`	STRING	NO	New ID for the canceled order. Automatically generated if not sent. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: “ABCD1234-…”.
`cancelRestrictions`	ENUM	NO	Supported values: `ONLY_NEW` - Cancel will succeed if the order status is `NEW`. `ONLY_PARTIALLY_FILLED` - Cancel will succeed if order status is `PARTIALLY_FILLED`.
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If both `orderId` and `origClientOrderId` parameters are specified, only `orderId` is used and `origClientOrderId` is ignored.
- `newClientOrderId` will replace `clientOrderId` of the canceled order, freeing it up for new orders.
- If you cancel an order that is a part of an OCO pair, the entire OCO is canceled.

Data Source: Matching Engine

Security Type: TRADE

Response

When an individual order is canceled:


```
{
  "id": "5633b6a2-90a9-4192-83e7-925c90b6a2fd",
  "status": 200,
  "result": {
    "symbol": "BTCUSDT",
    "origClientOrderId": "4d96324ff9d44481926157",  // clientOrderId that was canceled
    "orderId": 12569099453,
    "orderListId": -1,                              // set only for legs of an OCO
    "clientOrderId": "91fe37ce9e69c90d6358c0",      // newClientOrderId from request
    "price": "23416.10000000",
    "origQty": "0.00847000",
    "executedQty": "0.00001000",
    "cummulativeQuoteQty": "0.23416100",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "SELL",
    "stopPrice": "0.00000000",          // present only if stopPrice set for the order
    "trailingDelta": 0,                 // present only if trailingDelta set for the order
    "trailingTime": -1,                 // present only if trailingDelta set for the order
    "icebergQty": "0.00000000",         // present only if icebergQty set for the order
    "selfTradePreventionMode": "NONE"
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```


When an OCO is canceled:


```
{
  "id": "16eaf097-bbec-44b9-96ff-e97e6e875870",
  "status": 200,
  "result": {
    "orderListId": 19431,
    "contingencyType": "OCO",
    "listStatusType": "ALL_DONE",
    "listOrderStatus": "ALL_DONE",
    "listClientOrderId": "iuVNVJYYrByz6C4yGOPPK0",
    "transactionTime": 1660803702431,
    "symbol": "BTCUSDT",
    "orders": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569099453,
        "clientOrderId": "bX5wROblo6YeDwa9iTLeyY"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569099454,
        "clientOrderId": "Tnu2IP0J5Y4mxw3IATBfmW"
      }
    ],
    // OCO leg status format is the same as for individual orders.
    "orderReports": [
      {
        "symbol": "BTCUSDT",
        "origClientOrderId": "bX5wROblo6YeDwa9iTLeyY",
        "orderId": 12569099453,
        "orderListId": 19431,
        "clientOrderId": "OFFXQtxVFZ6Nbcg4PgE2DA",
        "price": "23450.50000000",
        "origQty": "0.00850000"
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "CANCELED",
```

```
        "timeInForce": "GTC",
        "type": "STOP_LOSS_LIMIT",
        "side": "BUY",
        "stopPrice": "23430.00000000",
        "selfTradePreventionMode": "NONE"
      },
      {
        "symbol": "BTCUSDT",
        "origClientOrderId": "Tnu2IP0J5Y4mxw3IATBfmW",
        "orderId": 12569099454,
        "orderListId": 19431,
        "clientOrderId": "OFFXQtxVFZ6Nbcg4PgE2DA",
        "price": "23400.00000000",
        "origQty": "0.00850000"
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "CANCELED",
        "timeInForce": "GTC",
        "type": "LIMIT_MAKER",
        "side": "BUY",
        "selfTradePreventionMode": "NONE"
      }
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

REGARDING `CANCELRESTRICTIONS`

If the `cancelRestrictions` value is not any of the supported values, the error will be:

```
{
    "code": -1145,
    "msg": "Invalid cancelRestrictions"
}
```

If the order did not pass the conditions for `cancelRestrictions`, the error will be:

```
{
    "code": -2011,
    "msg": "Order was not canceled due to cancel restrictions."
}
```

REPLACE ORDER (WEBSOCKET)

Example

```
{
  "id": "99de1036-b5e2-4e0f-9b5c-13d751c93a1a",
  "method": "order.cancelReplace",
  "params": {
    "symbol": "BTCUSDT",
    "cancelReplaceMode": "ALLOW_FAILURE",
    "cancelOrigClientOrderId": "4d96324ff9d44481926157",
    "side": "SELL",
    "type": "LIMIT",
    "timeInForce": "GTC",
    "price": "23416.10000000",
    "quantity": "0.00847000",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "7028fdc187868754d25e42c37ccfa5ba2bab1d180ad55d4c3a7e2de643943dc5",
    "timestamp": 1660813156900
  }
}
```

Cancel an existing order and immediately place a new order instead of the canceled one.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`cancelReplaceMode`	ENUM	YES	
`cancelOrderId`	INT	YES	Cancel order by `orderId`
`cancelOrigClientOrderId`	STRING		Cancel order by `clientOrderId`
`cancelNewClientOrderId`	STRING	NO	New ID for the canceled order. Automatically generated if not sent
`side`	ENUM	YES	`BUY` or `SELL`
`type`	ENUM	YES	
`timeInForce`	ENUM	NO *	
`price`	DECIMAL	NO *	
`quantity`	DECIMAL	NO *	
`quoteOrderQty`	DECIMAL	NO *	
`newClientOrderId`	STRING	NO	Arbitrary unique ID among open orders. Automatically generated if not sent. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: “ABCD1234-…”.

Name	Type	Mandatory	Description
`newOrderRespType`	ENUM	NO	Select response format: `ACK`, `RESULT`, `FULL`. `MARKET` and `LIMIT` orders produce `FULL` response by default, other order types default to `ACK`.
`stopPrice`	DECIMAL	NO *	
`trailingDelta`	DECIMAL	NO *	See [Trailing Stop order FAQ](https://github.com/binance-us/binance-us-api-docs/blob/master/faqs/trailing-stop-faq.md)
`icebergQty`	DECIMAL	NO	
`cancelRestrictions`	ENUM	NO	Supported values: `ONLY_NEW` - Cancel will succeed if the order status is `NEW`. `ONLY_PARTIALLY_FILLED` - Cancel will succeed if order status is `PARTIALLY_FILLED`.For more information please refer to Regarding `cancelRestrictions`
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Similar to the `order.place` request, additional mandatory parameters (*) are determined by the new order `type`.

Available `cancelReplaceMode` options:

- `STOP_ON_FAILURE` – if cancellation request fails, new order placement will not be attempted
- `ALLOW_FAILURE` – new order placement will be attempted even if the cancel request fails

Request	Response		
`cancelReplaceMode`	`cancelResult`	`newOrderResult`	`status`
`STOP_ON_FAILURE`	✅ `SUCCESS`	✅ `SUCCESS`	`200`
	❌ `FAILURE`	➖ `NOT_ATTEMPTED`	`400`
	✅ `SUCCESS`	❌ `FAILURE`	`409`
`ALLOW_FAILURE`	✅ `SUCCESS`	✅ `SUCCESS`	`200`
	❌ `FAILURE`	❌ `FAILURE`	`400`
	❌ `FAILURE`	✅ `SUCCESS`	`409`
	✅ `SUCCESS`	❌ `FAILURE`	`409`

Notes:

- If both `cancelOrderId` and `cancelOrigClientOrderId` parameters are specified, only `cancelOrderId` is used and `cancelOrigClientOrderId` is ignored.
- `cancelNewClientOrderId` will replace `clientOrderId` of the canceled order, freeing it up for new orders.
- `newClientOrderId` specifies `clientOrderId` value for the placed order.

A new order with the same `clientOrderId` is accepted only when the previous one is filled or expired.

The new order can reuse old `clientOrderId` of the canceled order.

- This cancel-replace operation is **not transactional**.

If one operation succeeds but the other one fails, the successful operation is still executed.

For example, in `STOP_ON_FAILURE` mode, if the new order placement fails, the old order is still canceled.

- Filters and order count limits are evaluated before cancellation and order placement occurs.
- If new order placement is not attempted, your order count is still incremented.
- Like `order.cancel`, if you cancel a leg of an OCO, the entire OCO is canceled.

Data Source: Matching Engine

Security Type: TRADE

Response

If both cancel and placement succeed, you get the following response with `"status": 200`:


```
{
  "id": "99de1036-b5e2-4e0f-9b5c-13d751c93a1a",
  "status": 200,
  "result": {
    "cancelResult": "SUCCESS",
    "newOrderResult": "SUCCESS",
    // Format is identical to "order.cancel" format.
    // Some fields are optional and are included only for orders that set them.
    "cancelResponse": {
      "symbol": "BTCUSDT",
      "origClientOrderId": "4d96324ff9d44481926157",  // cancelOrigClientOrderId from request
      "orderId": 125690984230,
      "orderListId": -1,
      "clientOrderId": "91fe37ce9e69c90d6358c0",      // cancelNewClientOrderId from request
      "price": "23450.00000000",
      "origQty": "0.00847000",
      "executedQty": "0.00001000",
      "cummulativeQuoteQty": "0.23450000",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "LIMIT",
      "side": "SELL",
      "selfTradePreventionMode": "NONE"
    },
    // Format is identical to "order.place" format, affected by "newOrderRespType".
    // Some fields are optional and are included only for orders that set them.
    "newOrderResponse": {
      "symbol": "BTCUSDT",
```

```json
      "orderId": 12569099453,
      "orderListId": -1,
      "clientOrderId": "bX5wROblo6YeDwa9iTLeyY",      // newClientOrderId from request
      "transactTime": 1660813156959,
      "price": "23416.10000000",
      "origQty": "0.00847000",
      "executedQty": "0.00000000",
      "cummulativeQuoteQty": "0.00000000",
      "status": "NEW",
      "timeInForce": "GTC",
      "type": "LIMIT",
      "side": "SELL",
      "workingTime": 1660813156959,
      "fills": [],
      "selfTradePreventionMode": "NONE"
    }
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

In `STOP_ON_FAILURE` mode, failed order cancellation prevents new order from being placed and returns the following response with `"status": 400`:

```json
{
  "id": "27e1bf9f-0539-4fb0-85c6-06183d36f66c",
  "status": 400,
  "error": {
    "code": -2022,
    "msg": "Order cancel-replace failed.",
    "data": {
      "cancelResult": "FAILURE",
      "newOrderResult": "NOT_ATTEMPTED",
      "cancelResponse": {
        "code": -2011,
        "msg": "Unknown order sent."
      },
      "newOrderResponse": null
    }
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

If cancel-replace mode allows failure and one of the operations fails, you get a response with `"status": 409`, and the `"data"` field detailing which operation succeeded, which failed, and why:

```json
{
  "id": "b220edfe-f3c4-4a3a-9d13-b35473783a25",
  "status": 409,
  "error": {
    "code": -2021,
    "msg": "Order cancel-replace partially failed.",
    "data": {
      "cancelResult": "SUCCESS",
      "newOrderResult": "FAILURE",
      "cancelResponse": {
        "symbol": "BTCUSDT",
        "origClientOrderId": "4d96324ff9d44481926157",
        "orderId": 125690984230,
        "orderListId": -1,
        "clientOrderId": "91fe37ce9e69c90d6358c0",
        "price": "23450.00000000",
        "origQty": "0.00847000",
        "executedQty": "0.00001000",
        "cummulativeQuoteQty": "0.23450000",
        "status": "CANCELED",
        "timeInForce": "GTC",
        "type": "LIMIT",
        "side": "SELL",
        "selfTradePreventionMode": "NONE"
      },
      "newOrderResponse": {
        "code": -2010,
        "msg": "Order would immediately match and take."
      }
    }
```

```json
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

```json
{
  "id": "ce641763-ff74-41ac-b9f7-db7cbe5e93b1",
  "status": 409,
  "error": {
    "code": -2021,
    "msg": "Order cancel-replace partially failed.",
    "data": {
      "cancelResult": "FAILURE",
      "newOrderResult": "SUCCESS",
      "cancelResponse": {
        "code": -2011,
        "msg": "Unknown order sent."
      },
      "newOrderResponse": {
        "symbol": "BTCUSDT",
        "orderId": 12569099453,
        "orderListId": -1,
        "clientOrderId": "bX5wROblo6YeDwa9iTLeyY",
        "transactTime": 1660813156959,
        "price": "23416.10000000",
        "origQty": "0.00847000",
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "NEW",
        "timeInForce": "GTC",
        "type": "LIMIT",
        "side": "SELL",
        "workingTime": 1669693344508,
        "fills": [],
        "selfTradePreventionMode": "NONE"
      }
    }
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

If both operations fail, response will have `"status": 400`:

```json
{
  "id": "3b3ac45c-1002-4c7d-88e8-630c408ecd87",
  "status": 400,
  "error": {
    "code": -2022,
    "msg": "Order cancel-replace failed.",
    "data": {
      "cancelResult": "FAILURE",
      "newOrderResult": "FAILURE",
      "cancelResponse": {
        "code": -2011,
        "msg": "Unknown order sent."
      },
      "newOrderResponse": {
        "code": -2010,
        "msg": "Order would immediately match and take."
      }
    }
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 1
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
```

```
      "limit": 160000,
      "count": 1
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

CURRENT OPEN ORDERS (WEBSOCKET)

Example

```
{
  "id": "55f07876-4f6f-4c47-87dc-43e5fff3f2e7",
  "method": "openOrders.status",
  "params": {
    "symbol": "BTCUSDT",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "d632b3fdb8a81dd44f82c7c901833309dd714fe508772a89b0a35b0ee0c48b89",
    "timestamp": 1660813156812
  }
}
```

Query execution status of all open orders.

If you need to continuously monitor order status updates, please consider using WebSocket Streams:

- `userDataStream.start` request
- `executionReport` user data stream event

Weight: Adjusted based on the number of requested symbols:

Parameter	Weight
`symbol`	3
none	40

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	NO	If omitted, open orders for all symbols are returned
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Data Source: Memory => Database

Security Type: USER_DATA

Status reports for open orders are identical to `order.status`.

Note that some fields are optional and included only for orders that set them.

Open orders are always returned as a flat list. If all symbols are requested, use the `symbol` field to tell which symbol the orders belong to.

Response

```
{
  "id": "55f07876-4f6f-4c47-87dc-43e5fff3f2e7",
  "status": 200,
  "result": [
    {
      "symbol": "BTCUSDT",
      "orderId": 12569099453,
      "orderListId": -1,
      "clientOrderId": "4d96324ff9d44481926157",
      "price": "23416.10000000",
      "origQty": "0.00847000",
      "executedQty": "0.00720000",
      "cummulativeQuoteQty": "172.43931000",
      "status": "PARTIALLY_FILLED",
      "timeInForce": "GTC",
      "type": "LIMIT",
      "side": "SELL",
      "stopPrice": "0.00000000",
      "icebergQty": "0.00000000",
      "time": 1660801715639,
      "updateTime": 1660801717945,
      "isWorking": true,
      "workingTime": 1660801715639,
      "origQuoteOrderQty": "0.00000000",
      "selfTradePreventionMode": "NONE"
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 3
    }
  ]
}
```

CANCEL OPEN ORDERS (WEBSOCKET)

Example

```
{
  "id": "778f938f-9041-4b88-9914-efbf64eeacc8",
  "method": "openOrders.cancelAll"
  "params": {
    "symbol": "BTCUSDT",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "773f01b6e3c2c9e0c1d217bc043ce383c1ddd6f0e25f8d6070f2b66a6ceaf3a5",
    "timestamp": 1660805557200
  }
}
```

Cancel all open orders on a symbol, including OCO orders.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Data Source: Matching Engine

Security Type: TRADE

Response

Cancellation reports for orders and OCOs have the same format as in `order.cancel`.

```
{
  "id": "778f938f-9041-4b88-9914-efbf64eeacc8",
  "status": 200,
  "result": [
    {
      "symbol": "BTCUSDT",
      "origClientOrderId": "4d96324ff9d44481926157",
      "orderId": 12569099453,
      "orderListId": -1,
      "clientOrderId": "91fe37ce9e69c90d6358c0",
      "price": "23416.10000000",
      "origQty": "0.00847000",
      "executedQty": "0.00001000",
      "cummulativeQuoteQty": "0.23416100",
      "status": "CANCELED",
      "timeInForce": "GTC",
      "type": "LIMIT",
      "side": "SELL",
      "stopPrice": "0.00000000",
      "trailingDelta": 0,                        // present only if trailingDelta set for the order
      "trailingTime": -1,                        // present only if trailingDelta set for the order
      "icebergQty": "0.00000000",
      "selfTradePreventionMode": "NONE"
    },
    {
      "orderListId": 19431,
      "contingencyType": "OCO",
      "listStatusType": "ALL_DONE",
      "listOrderStatus": "ALL_DONE",
      "listClientOrderId": "iuVNVJYYrByz6C4yGOPPK0",
      "transactionTime": 1660803702431,
      "symbol": "BTCUSDT",
      "orders": [
        {
          "symbol": "BTCUSDT",
          "orderId": 12569099453,
          "clientOrderId": "bX5wROblo6YeDwa9iTLeyY"
        },
        {
          "symbol": "BTCUSDT",
          "orderId": 12569099454,
          "clientOrderId": "Tnu2IP0J5Y4mxw3IATBfmW"
        }
      ],
      "orderReports": [
        {
          "symbol": "BTCUSDT",
          "origClientOrderId": "bX5wROblo6YeDwa9iTLeyY",
          "orderId": 12569099453,
          "orderListId": 19431,
          "clientOrderId": "OFFXQtxVFZ6Nbcg4PgE2DA",
          "price": "23450.50000000",
          "origQty": "0.00850000",
          "executedQty": "0.00000000",
          "cummulativeQuoteQty": "0.00000000",
          "status": "CANCELED",
          "timeInForce": "GTC",
          "type": "STOP_LOSS_LIMIT",
          "side": "BUY",
          "stopPrice": "23430.00000000",
          "selfTradePreventionMode": "NONE"
        },
        {
          "symbol": "BTCUSDT",
          "origClientOrderId": "Tnu2IP0J5Y4mxw3IATBfmW",
          "orderId": 12569099454,
          "orderListId": 19431,
          "clientOrderId": "OFFXQtxVFZ6Nbcg4PgE2DA",
          "price": "23400.00000000",
          "origQty": "0.00850000",
          "executedQty": "0.00000000",
          "cummulativeQuoteQty": "0.00000000",
          "status": "CANCELED",
          "timeInForce": "GTC",
          "type": "LIMIT_MAKER",
          "side": "BUY",
          "selfTradePreventionMode": "NONE"
        }
```

```
      ]
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

CREATE NEW OCO ORDER(WEBSOCKET)

Example

```
{
  "id": "56374a46-3061-486b-a311-99ee972eb648",
  "method": "orderList.place",
  "params": {
    "symbol": "BTCUSDT",
    "side": "SELL",
    "price": "23420.00000000",
    "quantity": "0.00650000",
    "stopPrice": "23410.00000000",
    "stopLimitPrice": "23405.00000000",
    "stopLimitTimeInForce": "GTC",
    "newOrderRespType": "RESULT",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "6689c2a36a639ff3915c2904871709990ab65f3c7a9ff13857558fd350315c35",
    "timestamp": 1660801713767
  }
}
```

Send in a new one-cancels-the-other (OCO) pair: `LIMIT_MAKER` + `STOP_LOSS`/`STOP_LOSS_LIMIT` orders (called *legs*), where activation of one order immediately cancels the other.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`side`	ENUM	YES	`BUY` or `SELL`
`price`	DECIMAL	YES	Price for the limit order
`quantity`	DECIMAL	YES	
`listClientOrderId`	STRING	NO	Arbitrary unique ID among open OCOs. Automatically generated if not sent
`limitClientOrderId`	STRING	NO	Arbitrary unique ID among open orders for the limit order. Automatically generated if not sent
`limitIcebergQty`	DECIMAL	NO	
`stopPrice`	DECIMAL	YES *	Either `stopPrice` or `trailingDelta`, or both must be specified
`trailingDelta`	INT	YES *	See Trailing Stop order FAQ
`stopClientOrderId`	STRING	NO	Arbitrary unique ID among open orders for the stop order. Automatically generated if not sent
`stopLimitPrice`	DECIMAL	NO *	
`stopLimitTimeInForce`	ENUM	NO *	See `order.place` for available options
`stopIcebergQty`	DECIMAL	NO *	
`newOrderRespType`	ENUM	NO	Select response format: `ACK`, `RESULT`, `FULL` (default)
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- `listClientOrderId` parameter specifies `listClientOrderId` for the OCO pair.

A new OCO with the same `listClientOrderId` is accepted only when the previous one is filled or completely expired.

`listClientOrderId` is distinct from `clientOrderId` of individual orders.

- `limitClientOrderId` and `stopClientOrderId` specify `clientOrderId` values for both legs of the OCO.

A new order with the same `clientOrderId` is accepted only when the previous one is filled or expired.

- Price restrictions on the legs:

`side`	Price relation
`BUY`	`price` < market price < `stopPrice`
`SELL`	`price` > market price > `stopPrice`

- Both legs have the same `quantity`.

However, you can set different iceberg quantity for individual legs.

If `stopIcebergQty` is used, `stopLimitTimeInForce` must be `GTC`.

- `trailingDelta` applies only to the `STOP_LOSS`/`STOP_LOSS_LIMIT` leg of the OCO.
- OCO counts as 2 orders against the order rate limit.

Data Source: Matching Engine

Security Type: TRADE

Response format for `orderReports` is selected using the `newOrderRespType` parameter. The following example is for `RESULT` response type. See `order.place` for more examples.

Response


```
{
  "id": "57833dc0-e3f2-43fb-ba20-46480973b0aa",
  "status": 200,
  "result": {
    "orderListId": 1274512,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "08985fedd9ea2cf6b28996",
    "transactionTime": 1660801713793,
    "symbol": "BTCUSDT",
    "orders": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138901,
        "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138902,
        "clientOrderId": "jLnZpj5enfMXTuhKB1d0us"
      }
    ],
    "orderReports": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138901,
        "orderListId": 1274512,
        "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU",
        "transactTime": 1660801713793,
        "price": "23410.00000000",
        "origQty": "0.00650000",
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "NEW",
        "timeInForce": "GTC",
        "type": "STOP_LOSS_LIMIT",
        "side": "SELL",
        "stopPrice": "23405.00000000",
        "workingTime": -1,
        "selfTradePreventionMode": "NONE"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138902,
        "orderListId": 1274512,
        "clientOrderId": "jLnZpj5enfMXTuhKB1d0us",
        "transactTime": 1660801713793,
        "price": "23420.00000000",
        "origQty": "0.00650000",
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "NEW",
        "timeInForce": "GTC",
        "type": "LIMIT_MAKER",
        "side": "SELL",
        "workingTime": 1660801713793,
        "selfTradePreventionMode": "NONE"
      }
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 2
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 2
    },
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```


GET OCO ORDER (WEBSOCKET)

Example


```
{
  "id": "b53fd5ff-82c7-4a04-bd64-5f9dc42c2100",
  "method": "orderList.status",
  "params": {
    "origClientOrderId": "08985fedd9ea2cf6b28996"
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "d12f4e8892d46c0ddfbd43d556ff6d818581b3be22a02810c2c20cb719aed6a4",
    "timestamp": 1660801713965
  }
}
```


Check execution status of an OCO.

For execution status of individual orders, use `order.status`.

Weight: 2

Parameters:

Name	Type	Mandatory	Description
`origClientOrderId`	STRING	YES	Query OCO by `listClientOrderId`
`orderListId`	INT		Query OCO by `orderListId`
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- `origClientOrderId` refers to `listClientOrderId` of the OCO itself.
- If both `origClientOrderId` and `orderListId` parameters are specified, only `origClientOrderId` is used and `orderListId` is ignored.

Data Source: Database

Security Type: USER_DATA

Response

```
{
  "id": "b53fd5ff-82c7-4a04-bd64-5f9dc42c2100",
  "status": 200,
  "result": {
    "orderListId": 1274512,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "08985fedd9ea2cf6b28996",
    "transactionTime": 1660801713793,
    "symbol": "BTCUSDT",
    "orders": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138901,
        "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138902,
        "clientOrderId": "jLnZpj5enfMXTuhKB1d0us"
      }
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 2
    }
  ]
}
```

CANCEL OCO ORDER (WEBSOCKET)

Example

```
{
  "id": "c5899911-d3f4-47ae-8835-97da553d27d0",
  "method": "orderList.cancel",
  "params": {
    "symbol": "BTCUSDT",
    "orderListId": 1274512,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "4973f4b2fee30bf6d45e4a973e941cc60fdd53c8dd5a25edeac96f5733c0ccee",
    "timestamp": 1660801720210
  }
}
```

Cancel an active OCO.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`orderListId`	INT	YES	Cancel OCO by `orderListId`
`listClientOrderId`	STRING		Cancel OCO by `listClientId`
`newClientOrderId`	STRING	NO	New ID for the canceled OCO. Automatically generated if not sent. For API Partner Program members: In order to receive rebates the newClientOrderId parameter must begin with your Partner ID, followed by a dash symbol, when calling order placement endpoints. For example: “ABCD1234-…”.
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If both `orderListId` and `listClientOrderId` parameters are specified, only `orderListId` is used and `listClientOrderId` is ignored.
- Canceling an individual leg with `order.cancel` will cancel the entire OCO as well.

Data Source: Matching Engine

Security Type: TRADE

Response

```json
{
  "id": "c5899911-d3f4-47ae-8835-97da553d27d0",
  "status": 200,
  "result": {
    "orderListId": 1274512,
    "contingencyType": "OCO",
    "listStatusType": "ALL_DONE",
    "listOrderStatus": "ALL_DONE",
    "listClientOrderId": "6023531d7edaad348f5aff",
    "transactionTime": 1660801720215,
    "symbol": "BTCUSDT",
    "orders": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138901,
        "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138902,
        "clientOrderId": "jLnZpj5enfMXTuhKB1d0us"
      }
    ],
    "orderReports": [
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138901,
        "orderListId": 1274512,
        "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU",
        "transactTime": 1660801720215,
        "price": "23410.00000000",
        "origQty": "0.00650000",
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "CANCELED",
        "timeInForce": "GTC",
        "type": "STOP_LOSS_LIMIT",
        "side": "SELL",
        "stopPrice": "23405.00000000",
        "selfTradePreventionMode": "NONE"
      },
      {
        "symbol": "BTCUSDT",
        "orderId": 12569138902,
        "orderListId": 1274512,
        "clientOrderId": "jLnZpj5enfMXTuhKB1d0us",
        "transactTime": 1660801720215,
        "price": "23420.00000000",
        "origQty": "0.00650000",
        "executedQty": "0.00000000",
        "cummulativeQuoteQty": "0.00000000",
        "status": "CANCELED",
        "timeInForce": "GTC",
        "type": "LIMIT_MAKER",
        "side": "SELL",
        "selfTradePreventionMode": "NONE"
      }
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

GET OPEN OCO ORDERS (WEBSOCKET)

Example

```json
{
  "id": "3a4437e2-41a3-4c19-897c-9cadc5dce8b6",
  "method": "openOrderLists.status",
  "params": {
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "1bea8b157dd78c3da30359bddcd999e4049749fe50b828e620e12f64e8b433c9",
    "timestamp": 1660801713831
  }
}
```

Query execution status of all open OCOs.

If you need to continuously monitor order status updates, please consider using WebSocket Streams:

- `userDataStream.start` request
- `executionReport` user data stream event

Weight: 3

Parameters:

Name	Type	Mandatory	Description
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Data Source: Database

Security Type: USER_DATA

Response


```
{
  "id": "3a4437e2-41a3-4c19-897c-9cadc5dce8b6",
  "status": 200,
  "result": [
    {
      "orderListId": 0,
      "contingencyType": "OCO",
      "listStatusType": "EXEC_STARTED",
      "listOrderStatus": "EXECUTING",
      "listClientOrderId": "08985fedd9ea2cf6b28996",
      "transactionTime": 1660801713793,
      "symbol": "BTCUSDT",
      "orders": [
        {
          "symbol": "BTCUSDT",
          "orderId": 4,
          "clientOrderId": "CUhLgTXnX5n2c0gWiLpV4d"
        },
        {
          "symbol": "BTCUSDT",
          "orderId": 5,
          "clientOrderId": "1ZqG7bBuYwaF4SU8CwnwHm"
        }
      ]
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 3
    }
  ]
}
```

Account requests

GET USER ACCOUNT INFORMATION (WEBSOCKET)

Example

```
{
  "id": "605a6d20-6588-4cb9-afa0-b0ab087507ba",
  "method": "account.status",
  "params": {
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "83303b4a136ac1371795f465808367242685a9e3a42b22edb4d977d0696eb45c",
    "timestamp": 1660801839480
  }
}
```

Query information about your account.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Data Source: Memory => Database

Security Type: USER_DATA

Response


```
{
  "id": "605a6d20-6588-4cb9-afa0-b0ab087507ba",
  "status": 200,
  "result": {
    "makerCommission": 15,
    "takerCommission": 15,
    "buyerCommission": 0,
    "sellerCommission": 0,
    "canTrade": true,
    "canWithdraw": true,
    "canDeposit": true,
    "commissionRates": {
      "maker": "0.00150000",
      "taker": "0.00150000",
      "buyer": "0.00000000",
      "seller":"0.00000000"
    },
    "brokered": false,
    "requireSelfTradePrevention": false,
    "updateTime": 1660801833000,
    "accountType": "SPOT",
    "balances": [
      {
        "asset": "BNB",
        "free": "0.00000000",
        "locked": "0.00000000"
      },
      {
        "asset": "BTC",
        "free": "1.3447112",
        "locked": "0.08600000"
      },
```

```
      {
        "asset": "USDT",
        "free": "1021.21000000",
        "locked": "0.00000000"
      }
    ],
    "permissions": [
      "SPOT"
    ]
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
}
```

GET ORDER RATE LIMITS (WEBSOCKET)

Example

```
{
  "id": "d3783d8d-f8d1-4d2c-b8a0-b7596af5a664",
  "method": "account.rateLimits.orders",
  "params": {
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "76289424d6e288f4dc47d167ac824e859dabf78736f4348abbbac848d719eb94",
    "timestamp": 1660801839500
  }
}
```

Query your current order rate limit.

Weight: 20

Parameters:

Name	Type	Mandatory	Description
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Data Source: Memory

Security Type: USER_DATA

Response

```
{
  "id": "d3783d8d-f8d1-4d2c-b8a0-b7596af5a664",
  "status": 200,
  "result": [
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "intervalNum": 10,
      "limit": 50,
      "count": 0
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "intervalNum": 1,
      "limit": 160000,
      "count": 0
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 20
    }
  ]
}
```

ACCOUNT ORDER HISTORY (WEBSOCKET)

Example

```
{
  "id": "734235c2-13d2-4574-be68-723e818c08f3",
  "method": "allOrders",
  "params": {
    "symbol": "BTCUSDT",
    "startTime": 1660780800000,
    "endTime": 1660867200000,
    "limit": 5,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "f50a972ba7fad92842187643f6b930802d4e20bce1ba1e788e856e811577bd42",
    "timestamp": 1661955123341
  }
}
```

Query information about all your orders – active, canceled, filled – filtered by time range.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`orderId`	INT	NO	Order ID to begin at
`startTime`	INT	NO	
`endTime`	INT	NO	
`limit`	INT	NO	Default 500; max 1000
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If `startTime` and/or `endTime` are specified, `orderId` is ignored.

Orders are filtered by `time` of the last execution status update.

- If `orderId` is specified, return orders with order ID >= `orderId`.
- If no condition is specified, the most recent orders are returned.
- For some historical orders the `cummulativeQuoteQty` response field may be negative, meaning the data is not available at this time.

Data Source: Database

Security Type: USER_DATA

Status reports for orders are identical to `order.status`.

Note that some fields are optional and included only for orders that set them.

Response

```
{
  "id": "734235c2-13d2-4574-be68-723e818c08f3",
  "status": 200,
  "result": [
    {
      "symbol": "BTCUSDT",
      "orderId": 12569099453,
      "orderListId": -1,
      "clientOrderId": "4d96324ff9d44481926157",
      "price": "23416.10000000",
      "origQty": "0.00847000",
      "executedQty": "0.00847000",
      "cummulativeQuoteQty": "198.33521500",
      "status": "FILLED",
      "timeInForce": "GTC",
      "type": "LIMIT",
      "side": "SELL",
      "stopPrice": "0.00000000",
      "icebergQty": "0.00000000",
      "time": 1660801715639,
      "updateTime": 1660801717945,
      "isWorking": true,
      "workingTime": 1660801715639,
      "origQuoteOrderQty": "0.00000000",
      "selfTradePreventionMode": "NONE",
      "preventedMatchId": 0,            // This field only appears if the order expired due to STP.
      "preventedQuantity": "1.200000"   // This field only appears if the order expired due to STP.
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
}
```

ACCOUNT OCO HISTORY (WEBSOCKET)

Example

```
{
  "id": "8617b7b3-1b3d-4dec-94cd-eefd929b8ceb",
  "method": "allOrderLists",
  "params": {
    "startTime": 1660780800000,
    "endTime": 1660867200000,
    "limit": 5,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "c8e1484db4a4a02d0e84dfa627eb9b8298f07ebf12fcc4eaf86e4a565b2712c2",
    "timestamp": 1661955123341
  }
}
```

Query information about all your OCOs, filtered by time range.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
`fromId`	INT	NO	Order list ID to begin at
`startTime`	INT	NO	

Name	Type	Mandatory	Description
`endTime`	INT	NO	
`limit`	INT	NO	Default 500; max 1000
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If `startTime` and/or `endTime` are specified, `fromId` is ignored.

OCOs are filtered by `transactionTime` of the last OCO execution status update.

- If `fromId` is specified, return OCOs with order list ID >= `fromId`.
- If no condition is specified, the most recent OCOs are returned.

Data Source: Database

Security Type: USER_DATA

Response

Status reports for OCOs are identical to `orderList.status`.


```json
{
  "id": "8617b7b3-1b3d-4dec-94cd-eefd929b8ceb",
  "status": 200,
  "result": [
    {
      "orderListId": 1274512,
      "contingencyType": "OCO",
      "listStatusType": "EXEC_STARTED",
      "listOrderStatus": "EXECUTING",
      "listClientOrderId": "08985fedd9ea2cf6b28996",
      "transactionTime": 1660801713793,
      "symbol": "BTCUSDT",
      "orders": [
        {
          "symbol": "BTCUSDT",
          "orderId": 12569138901,
          "clientOrderId": "BqtFCj5odMoWtSqGk2X9tU"
        },
        {
          "symbol": "BTCUSDT",
          "orderId": 12569138902,
          "clientOrderId": "jLnZpj5enfMXTuhKB1d0us"
        }
      ]
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
}
```


ACCOUNT TRADE HISTORY (WEBSOCKET)

Example


```json
{
  "id": "f4ce6a53-a29d-4f70-823b-4ab59391d6e8",
  "method": "myTrades",
  "params": {
    "symbol": "BTCUSDT",
    "startTime": 1660780800000,
    "endTime": 1660867200000,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "c5a5ffb79fd4f2e10a92f895d488943a57954edf5933bde3338dfb6ea6d6eefc",
    "timestamp": 1661955125250
  }
}
```


Query information about all your trades, filtered by time range.

Weight: 10

Parameters:

Name	Type	Mandatory	Description
`symbol`	STRING	YES	
`orderId`	INT	NO	
`startTime`	INT	NO	
`endTime`	INT	NO	
`fromId`	INT	NO	First trade ID to query
`limit`	INT	NO	Default 500; max 1000
`apiKey`	STRING	YES	
`recvWindow`	INT	NO	The value cannot be greater than `60000`

Name	Type	Mandatory	Description
`signature`	STRING	YES	
`timestamp`	INT	YES	

Notes:

- If `fromId` is specified, return trades with trade ID >= `fromId`.
- If `startTime` and/or `endTime` are specified, trades are filtered by execution time (`time`).

`fromId` cannot be used together with `startTime` and `endTime`.

- If `orderId` is specified, only trades related to that order are returned.

`startTime` and `endTime` cannot be used together with `orderId`.

- If no condition is specified, the most recent trades are returned.

Data Source: Memory => Database

Security Type: USER_DATA

Response


```
{
  "id": "f4ce6a53-a29d-4f70-823b-4ab59391d6e8",
  "status": 200,
  "result": [
    {
      "symbol": "BTCUSDT",
      "id": 1650422481,
      "orderId": 12569099453,
      "orderListId": -1,
      "price": "23416.10000000",
      "qty": "0.00635000",
      "quoteQty": "148.69223500",
      "commission": "0.00000000",
      "commissionAsset": "BNB",
      "time": 1660801715793,
      "isBuyer": false,
      "isMaker": true,
      "isBestMatch": true
    },
    {
      "symbol": "BTCUSDT",
      "id": 1650422482,
      "orderId": 12569099453,
      "orderListId": -1,
      "price": "23416.50000000",
      "qty": "0.00212000",
      "quoteQty": "49.64298000",
      "commission": "0.00000000",
      "commissionAsset": "BNB",
      "time": 1660801715793,
      "isBuyer": false,
      "isMaker": true,
      "isBestMatch": true
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
}
```


ACCOUNT PREVENTED MATCHES (WEBSOCKET)

Example


```
{
  "id": "g4ce6a53-a39d-4f71-823b-4ab5r391d6y8",
  "method": "myPreventedMatches",
  "params": {
    "symbol": "BTCUSDT",
    "orderId": 35,
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A",
    "signature": "c5a5ffb79fd4f2e10a92f895d488943a57954edf5933bde3338dfb6ea6d6eefc",
    "timestamp": 1673923281052
  }
}
```


Displays the list of orders that were expired due to STP.

These are the combinations supported:

- `symbol` + `preventedMatchId`
- `symbol` + `orderId`
- `symbol` + `orderId` + `fromPreventedMatchId` (`limit` will default to 500)
- `symbol` + `orderId` + `fromPreventedMatchId` + `limit`

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
preventedMatchId	LONG	NO	
orderId	LONG	NO	
fromPreventedMatchId	LONG	NO	
limit	INT	NO	Default: `500`; Max: `1000`
recvWindow	LONG	NO	The value cannot be greater than `60000`

Name	Type	Mandatory	Description
timestamp	LONG	YES	

Weight

Case	Weight
If `symbol` is invalid	1
Querying by `preventedMatchId`	1
Querying by `orderId`	10

Data Source: Database

Security Type: USER_DATA

Response


```
{
  "id": "g4ce6a53-a39d-4f71-823b-4ab5r391d6y8",
  "status": 200,
  "result": [
    {
      "symbol": "BTCUSDT",
      "preventedMatchId": 1,
      "takerOrderId": 5,
      "makerOrderId": 3,
      "tradeGroupId": 1,
      "selfTradePreventionMode": "EXPIRE_MAKER",
      "price": "1.100000",
      "makerPreventedQuantity": "1.300000",
      "transactTime": 1669101687094
    }
  ],
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 10
    }
  ]
}
```


User Data Stream requests

The following requests manage User Data Stream subscriptions.

Note: The user data can ONLY be retrieved by *a separate* Websocket connection via the **User Data Streams** url (i.e. `wss://stream.binance.us:443`).

START USER DATA STREAM (WEBSOCKET)

Example


```
{
  "id": "d3df8a61-98ea-4fe0-8f4e-0fcea5d418b0",
  "method": "userDataStream.start",
  "params": {
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A"
  }
}
```


Start a new user data stream.

Note: the stream will close in 60 minutes unless `userDataStream.ping` requests are sent regularly.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`apiKey`	STRING	YES	

Data Source: Memory

Security Type: USER_STREAM

Response

Subscribe to the received listen key on WebSocket Stream afterwards.


```
{
  "id": "d3df8a61-98ea-4fe0-8f4e-0fcea5d418b0",
  "status": 200,
  "result": {
    "listenKey": "xs0mRXdAKlIPDRFrlPcw0qI41Eh3ixNntmymGyhrhgqo7L6FuLaWArTD7RLP"
  },
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```


PING USER DATA STREAM (WEBSOCKET)

Example

```json
{
  "id": "815d5fce-0880-4287-a567-80badf004c74",
  "method": "userDataStream.ping",
  "params": {
    "listenKey": "xs0mRXdAKlIPDRFrlPcw0qI41Eh3ixNntmymGyhrhgqo7L6FuLaWArTD7RLP",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A"
  }
}
```

Ping a user data stream to keep it alive.

User data streams close automatically after 60 minutes, even if you're listening to them on WebSocket Streams. In order to keep the stream open, you have to regularly send pings using the `userDataStream.ping` request.

It is recommended to send a ping once every 30 minutes.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`listenKey`	STRING	YES	
`apiKey`	STRING	YES	

Data Source: Memory

Security Type: USER_STREAM

Response

```json
{
  "id": "815d5fce-0880-4287-a567-80badf004c74",
  "status": 200,
  "response": {},
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

STOP USER DATA STREAM (WEBSOCKET)

Example

```json
{
  "id": "819e1b1b-8c06-485b-a13e-131326c69599",
  "method": "userDataStream.stop",
  "params": {
    "listenKey": "xs0mRXdAKlIPDRFrlPcw0qI41Eh3ixNntmymGyhrhgqo7L6FuLaWArTD7RLP",
    "apiKey": "vmPUZE6mv9SD5VNHk4HlWFsOr6aKE2zvsw0MuIgwCIPy6utIco14y7Ju91duEh8A"
  }
}
```

Explicitly stop and close the user data stream.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
`listenKey`	STRING	YES	
`apiKey`	STRING	YES	

Data Source: Memory

Security Type: USER_STREAM

Response

```json
{
  "id": "819e1b1b-8c06-485b-a13e-131326c69599",
  "status": 200,
  "response": {},
  "rateLimits": [
    {
      "rateLimitType": "REQUEST_WEIGHT",
      "interval": "MINUTE",
      "intervalNum": 1,
      "limit": 1200,
      "count": 1
    }
  ]
}
```

WebSocket Streams

WebSocket Information

- The base endpoint is: **wss://stream.binance.us:9443**

- Streams can be accessed either in a single raw stream or in a combined stream
- Raw streams are accessed at **/ws/<streamName>**
- Combined streams are accessed at **/stream?streams=<streamName1>/<streamName2>/<streamName3>**
- Combined stream events are wrapped as follows: **{"stream":"<streamName>","data":<rawPayload>}**
- All symbols for streams are in **lowercase**
- A single connection to **stream.binance.us** is only valid for 24-hours; expect to be disconnected at the 24 hour mark
- The WebSocket server will send a `ping frame` every 3 minutes. If the WebSocket server does not receive a `pong frame` back from the connection within a 10-minute period, the connection will be disconnected. Unsolicited `pong frames` are allowed.

WebSocket Rate Limits

- WebSocket connections have a limit of 5 incoming messages per second. A message is considered:
  - A PING frame
  - A PONG frame
  - A JSON controlled message (e.g., subscribe, unsubscribe)
- A connection that goes beyond the limit will be disconnected; IPs that are repeatedly disconnected may be banned.
- A single connection can listen to a maximum of 1024 streams.

WebSocket Errors

Error Message	Description
{"code": 0, "msg": "Unknown property", "ID": '%s'}	Parameter used in the `SET_PROPERTY` or `GET_PROPERTY` was invalid
{"code": 1, "msg": "Invalid value type: expected Boolean"}	Value should only be `true` or `false`
{"code": 2, "msg": "Invalid request: property name must be a string"}	Property name provided was invalid
{"code": 2, "msg": "Invalid request: request ID must be an unsigned integer"}	Parameter `ID` had to be provided or the value provided in the `ID` parameter is an unsupported type
{"code": 2, "msg": "Invalid request: unknown variant %s, expected one of `SUBSCRIBE`, `UNSUBSCRIBE`, `LIST_SUBSCRIPTIONS`, `SET_PROPERTY`, `GET_PROPERTY` at line 1 column 28"}	Possible typo in the provided method or provided method was neither of the expected values
{"code": 2, "msg": "Invalid request: too many parameters"}	Unnecessary parameters provided in the data
{"code": 2, "msg": "Invalid request: property name must be a string"}	Property name was not provided
{"code": 2, "msg": "Invalid request: missing field `method` at line 1 column 73"}	`method` was not provided in the data
{"code": 3, "msg": "Invalid JSON: expected value at line %s column %s"}	JSON data sent has incorrect syntax

Subscribing/Unsubscribing

- The following data can be sent through the WebSocket instance in order to subscribe/unsubscribe from streams. Examples can be seen below.
- The `id` used in the JSON payloads is an unsigned INT used as an identifier to uniquely identify the messages going back and forth.
- In the response, if the `result` received is `null` which means the request sent was a success for non-query requests (e.g., subscribing/unsubscribing).

SUBSCRIBE TO A STREAM

Request

```
{
    "method": "SUBSCRIBE",
    "params": [
      "btcusdt@aggTrade",
      "btcusdt@depth"
    ],
    "id": 1
}
```

Response

```
{
    "result": null,
    "id": 1
}
```

UNSUBSCRIBE TO A STREAM

Request

```
  {
    "method": "UNSUBSCRIBE",
    "params": [
      "btcusdt@depth"
    ],
    "id": 312
  }
```

Response

```
  {
    "result": null,
    "id": 312
  }
```

LIST SUBSCRIPTIONS

Request

```
{
  "method": "LIST_SUBSCRIPTIONS",
  "id": 3
}
```

Response

```
{
  "result": [
    "btcusdt@aggTrade"
  ],
  "id": 3
}
```

SET PROPERTIES

Currently, the only property that can be set is whether `combined` stream payloads are enabled or not. The combined property is set to `false` when connecting using `/ws/` ("raw streams") and `true` when connecting using `/stream/`.

Request

```
{
  "method": "SET_PROPERTY",
  "params": [
    "combined",
    true
  ],
  "id": 5
}
```

Response

```
{
   "result": null,
   "id": 5
}
```

RETRIEVE PROPERTIES

Request

```
{
  "method": "GET_PROPERTY",
  "params": [
    "combined"
  ],
  "id": 2
}
```

Response

```
{
  "result": true, // Indicates that combined is set to true.
  "id": 2
}
```

Market Data Streams

- The base endpoint is: **wss://stream.binance.us:9443**
- Streams can be accessed either in a single raw stream or in a combined stream
- Raw streams are accessed at **/ws/<streamName>**
- Combined streams are accessed at **/stream?streams=<streamName1>/<streamName2>/<streamName3>**
- Combined stream events are wrapped as follows: **{"stream":"<streamName>","data":<rawPayload>}**
- All symbols for streams are **lowercase**
- A single connection to **stream.binance.us** is only valid for 24-hours; expect to be disconnected at the 24 hour mark
- The WebSocket server will send a `ping frame` every 3 minutes. If the WebSocket server does not receive a `pong frame` back from the connection within a 10-minute period, the connection will be disconnected. Unsolicited `pong frames` are allowed.

Trade Data Streams

AGGREGATE TRADE STREAM

Payload:

```
{
  "e": "aggTrade",  // Event type
  "E": 1672515782136,   // Event time
  "s": "BNBBTC",    // Symbol
  "a": 12345,       // Aggregate trade ID
  "p": "0.001",     // Price
  "q": "100",       // Quantity
  "f": 100,         // First trade ID
  "l": 105,         // Last trade ID
  "T": 1672515782136,   // Trade time
  "m": true,        // Is the buyer the market maker?
  "M": true         // Ignore
}
```

The Aggregate Trade Streams push trade information that is aggregated for a single taker order.

Stream Name: `<symbol>@aggTrade`

Update Speed: Real-time

TRADE DATA STREAM

Payload:

```
{
  "e": "trade",     // Event type
  "E": 1672515782136,   // Event time
  "s": "BNBBTC",    // Symbol
  "t": 12345,       // Trade ID
  "p": "0.001",     // Price
  "q": "100",       // Quantity
  "b": 88,          // Buyer order ID
  "a": 50,          // Seller order ID
  "T": 1672515782136,   // Trade time
  "m": true,        // Is the buyer the market maker?
  "M": true         // Ignore
}
```

The Trade Streams push raw trade information; each trade has a unique buyer and seller.

Stream Name: `<symbol>@trade`

Update Speed: Real-time

CANDLESTICK DATA STREAM

Payload:

```
{
  "e": "kline",     // Event type
  "E": 1672515782136,   // Event time
  "s": "BNBBTC",    // Symbol
  "k": {
    "t": 1672515780000, // Kline start time
    "T": 1672515839999, // Kline close time
    "s": "BNBBTC",  // Symbol
    "i": "1m",      // Interval
    "f": 100,       // First trade ID
    "L": 200,       // Last trade ID
    "o": "0.0010",  // Open price
    "c": "0.0020",  // Close price
    "h": "0.0025",  // High price
    "l": "0.0015",  // Low price
    "v": "1000",    // Base asset volume
    "n": 100,       // Number of trades
    "x": false,     // Is this kline closed?
    "q": "1.0000",  // Quote asset volume
    "V": "500",     // Taker buy base asset volume
    "Q": "0.500",   // Taker buy quote asset volume
    "B": "123456"   // Ignore
  }
}
```

The Kline/Candlestick Stream pushes updates to the current klines/candlestick every second.

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

Stream Name: `<symbol>@kline_<interval>`

Update Speed: 2000ms

Ticker Streams

INDIVIDUAL SYMBOL ROLLING WINDOW STATISTICS STREAMS

Payload:

```
{
  "e": "1hTicker",    // Event type
  "E": 1672515782136,     // Event time
  "s": "BNBBTC",      // Symbol
  "p": "0.0015",      // Price change
  "P": "250.00",      // Price change percent
  "o": "0.0010",      // Open price
  "h": "0.0025",      // High price
  "l": "0.0010",      // Low price
  "c": "0.0025",      // Last price
  "w": "0.0018",      // Weighted average price
  "v": "10000",       // Total traded base asset volume
  "q": "18",          // Total traded quote asset volume
  "O": 0,             // Statistics open time
  "C": 1675216573749,      // Statistics close time
  "F": 0,             // First trade ID
  "L": 18150,         // Last trade Id
  "n": 18151          // Total number of trades
}
```

Rolling window ticker statistics for a single symbol, computed over multiple windows.

Stream Name: `<symbol>@ticker_<window_size>`

Window Sizes: 1h,4h

Update Speed: 1000ms

Note: This stream is different from the `<symbol>@ticker` stream. The open time `O` always starts at the beginning of the minute, while the closing time `C` is the current time of the update. As such, the effective window might be up to 59999ms wider than `<window_size>`.

ALL MARKET TICKER STREAM

Payload:

```
[
  {
    // Same as <symbol>@ticker payload
  }
]
```

24hr rolling window ticker statistics for all symbols that changed in an array. These are NOT the statistics of the UTC day, but a 24hr rolling window for the previous 24hrs. Note that only tickers that have changed will be present in the array.

Stream Name: `!ticker@arr`

Update Speed: 1000ms

ALL MARKET ROLLING WINDOW STATISTICS STREAMS

Payload:

```
[
  {
    // Same as <symbol>@ticker_<window-size> payload,
    // one for each symbol updated within the interval.
  }
]
```

Rolling window ticker statistics for all market symbols, computed over multiple windows. Note that only tickers that have changed will be present in the array.

Stream Name: `!ticker_<window-size>@arr`

Window Size: 1h,4h

Update Speed: 1000ms

Price Change Streams

TICKER 24H CHANGE STREAM

Payload:

```
{
  "e": "24hrTicker",  // Event type
  "E": 1672515782136,     // Event time
  "s": "BNBBTC",      // Symbol
  "p": "0.0015",      // Price change
  "P": "250.00",      // Price change percent
  "w": "0.0018",      // Weighted average price
  "x": "0.0009",      // First trade(F)-1 price (first trade before the 24hr rolling window)
  "c": "0.0025",      // Last price
  "Q": "10",          // Last quantity
  "b": "0.0024",      // Best bid price
  "B": "10",          // Best bid quantity
  "a": "0.0026",      // Best ask price
  "A": "100",         // Best ask quantity
  "o": "0.0010",      // Open price
  "h": "0.0025",      // High price
  "l": "0.0010",      // Low price
  "v": "10000",       // Total traded base asset volume
  "q": "18",          // Total traded quote asset volume
  "O": 0,             // Statistics open time
  "C": 1675216573749,      // Statistics close time
  "F": 0,             // First trade ID
  "L": 18150,         // Last trade Id
  "n": 18151          // Total number of trades
}
```

24hr rolling window ticker statistics for a single symbol. These are NOT the statistics of the UTC day, but a 24hr rolling window for the previous 24hrs.

Stream Name: `<symbol>@ticker`

Update Speed: 1000ms

INDIVIDUAL SYMBOL 24H CHANGE STREAM

Payload:

```
  {
    "e": "24hrMiniTicker",  // Event type
    "E": 1672515782136,     // Event time
    "s": "BNBBTC",          // Symbol
    "c": "0.0025",          // Close price
    "o": "0.0010",          // Open price
    "h": "0.0025",          // High price
    "l": "0.0010",          // Low price
    "v": "10000",           // Total traded base asset volume
    "q": "18"               // Total traded quote asset volume
  }
```

24hr rolling window mini-ticker statistics. These are NOT the statistics of the UTC day, but a 24hr rolling window for the previous 24hrs.

Stream Name: <symbol>@miniTicker

Update Speed: 1000ms

ALL MARKET 24H CHANGE STREAM

Payload:

```
[
  {
    // Same as <symbol>@miniTicker payload
  }
]
```

24hr rolling window mini-ticker statistics for all symbols that changed in an array. These are NOT the statistics of the UTC day, but a 24hr rolling window for the previous 24hrs. Note that only tickers that have changed will be present in the array.

Stream Name: `!miniTicker@arr`

Update Speed: 1000ms

Order Book Streams

TICKER ORDER BOOK STREAM

Payload:

```
{
  "u":400900217,     // order book updateId
  "s":"BNBUSDT",     // symbol
  "b":"25.35190000", // best bid price
  "B":"31.21000000", // best bid qty
  "a":"25.36520000", // best ask price
  "A":"40.66000000"  // best ask qty
}
```

Pushes any update to the best bid or asks price or quantity in real-time for a specified symbol.

Stream Name: `<symbol>@bookTicker`

Update Speed: Real-time

PARTIAL ORDER BOOK DEPTH STREAM

Payload:

```
{
  "lastUpdateId": 160,  // Last update ID
  "bids": [             // Bids to be updated
    [
      "0.0024",         // Price level to be updated
      "10"              // Quantity
    ]
  ],
  "asks": [             // Asks to be updated
    [
      "0.0026",         // Price level to be updated
      "100"            // Quantity
    ]
  ]
}
```

Top **<levels>** bids and asks, pushed every second. Valid **<levels>** are 5, 10, or 20.

Stream Names: `<symbol>@depth<levels>` OR `<symbol>@depth<levels>@100ms`

Update Speed: 1000ms or 100ms

ORDER BOOK DEPTH DIFF STREAM

Payload:

```
{
  "e": "depthUpdate", // Event type
  "E": 1675216573749,     // Event time
  "s": "BNBBTC",      // Symbol
  "U": 157,           // First update ID in event
  "u": 160,           // Final update ID in event
  "b": [              // Bids to be updated
    [
      "0.0024",       // Price level to be updated
      "10"            // Quantity
    ]
  ],
  "a": [              // Asks to be updated
    [
      "0.0026",       // Price level to be updated
      "100"           // Quantity
    ]
  ]
}
```

Order book price and quantity depth updates are used to manage an order book locally.

Stream Name: `<symbol>@depth` OR `<symbol>@depth@100ms`

Update Speed: 1000ms or 100ms

MANAGING A LOCAL ORDER BOOK

1. Open a stream to **wss://stream.binance.us:9443/ws/bnbbtc@depth**
2. Buffer the events you receive from the stream
3. Get a depth snapshot from **https://www.binance.us/api/v1/depth?symbol=BNBBTC&limit=1000**
4. Drop any event where `u` is <= `lastUpdateId` in the snapshot
5. The first processed should have `U` <= `lastUpdateId`+1 **AND** `u` >= `lastUpdateId`+1
6. While listening to the stream, each new event's `U` should be equal to the previous event's `u`+1
7. The data in each event is the **absolute** quantity for a price level
8. If the quantity is 0, **remove** the price level
9. Receiving an event that removes a price level that is not in your local order book is normal

---

User Data Streams

- The base API endpoint is: **https://api.binance.us**
- A User Data Stream `listenKey` is valid for 60 minutes after creation.
- Doing a `PUT` on a `listenKey` will extend its validity for 60 minutes.
- Doing a `DELETE` on a `listenKey` will close the stream and invalidate the `listenKey`.
- Doing a `POST` on an account with an active `listenKey` will return the currently active `listenKey` and extend its validity for 60 minutes.
- The base WebSocket endpoint is: **wss://stream.binance.us:9443**
- User Data Streams are accessed at **/ws/<listenKey>** or **/stream?streams=<listenKey>**
- A single connection to **stream.binance.us** is only valid for 24 hours; expect to be disconnected at the 24-hour mark

---

User Data Stream Endpoints

CREATE USER DATA STREAM

Example

```
curl -X "POST" "https://api.binance.us/api/v3/userDataStream" </span>
   -H "X-MBX-APIKEY: <your_api_key>"
```

Response

```
{
  "listenKey": "pqia91ma19a5s61cv6a81va65sdf19v8a65a1a5s61cv6a81va65sdf19v8a65a1"
}
```

`POST /api/v3/userDataStream`

Start a new user data stream. The stream will close after 60 minutes unless a keepalive is sent. If the account has an active `listenKey`, that `listenKey` will be returned and its validity will be extended for 60 minutes.

Weight: 1

Parameters: NONE

EXTEND USER DATA STREAM

Example

```
curl -X "PUT" "https://api.binance.us/api/v3/userDataStream" \
   -H "X-MBX-APIKEY: <your_api_key>"  \
   -d 'listenKey=<listen_key>'
```

Response

```
{}
```

`PUT /api/v3/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	

CLOSE USER DATA STREAM

Example

```
curl -X "DELETE" "https://api.binance.us/api/v3/userDataStream" \
   -H "X-MBX-APIKEY: <your_api_key>" \
   -d 'listenKey=<listen_key>'
```

Response

```
{}
```

`DELETE /api/v3/userDataStream`

Close out a user data stream.

Weight: 1

Parameters:

Name	Type	Mandatory	Description
listenKey	STRING	YES	

User Data Stream Payloads

ACCOUNT UPDATE

```
{
  "e": "outboundAccountPosition", //Event type
  "E": 1564034571105,             //Event Time
  "u": 1564034571073,             //Time of last account update
  "B": [                          //Balances Array
    {
      "a": "ETH",                 //Asset
      "f": "10000.000000",        //Free
      "l": "0.000000"             //Locked
    }
  ]
}
```

The event `outboundAccountPosition` is sent any time an account balance has changed and contains the assets that were possibly changed by the event that generated the balance change.

ORDER UPDATE

Payload A

```
{
  "e": "executionReport",        // Event type
  "E": 1499405658658,            // Event time
  "s": "ETHBTC",                 // Symbol
  "c": "mUvoqJxFIILMdfAW5iGSOW", // Client order ID
  "S": "BUY",                    // Side
  "o": "LIMIT",                  // Order type
  "f": "GTC",                    // Time in force
  "q": "1.00000000",             // Order quantity
  "p": "0.10264410",             // Order price
  "P": "0.00000000",             // Stop price
  "d": 4,                        // Trailing Delta; This is only visible if the order was a trailing stop order.
  "F": "0.00000000",             // Iceberg quantity
  "g": -1,                       // OrderListId
  "C": "",                       // Original client order ID; This is the ID of the order being canceled
  "x": "NEW",                    // Current execution type
  "X": "NEW",                    // Current order status
  "r": "NONE",                   // Order reject reason; will be an error code.
  "i": 4293153,                  // Order ID
  "l": "0.00000000",             // Last executed quantity
  "z": "0.00000000",             // Cumulative filled quantity
  "L": "0.00000000",             // Last executed price
  "n": "0",                      // Commission amount
  "N": null,                     // Commission asset
  "T": 1499405658657,            // Transaction time
  "t": -1,                       // Trade ID
  "I": 8641984,                  // Ignore
  "w": true,                     // Is the order on the book?
  "m": false,                    // Is this trade the maker side?
  "M": false,                    // Ignore
  "O": 1499405658657,            // Order creation time
  "Z": "0.00000000",             // Cumulative quote asset transacted quantity
  "Y": "0.00000000",             // Last quote asset transacted quantity (i.e. lastPrice * lastQty)
  "Q": "0.00000000",             //Quote Order Quantity
  "V": "selfTradePreventionMode",
  "D": "trailing_time",          // (Appears if the trailing stop order is active)
  "W": "workingTime"             // (Appears if the order is working on the order book)
  "u":12332                      // tradeGroupId (Appear if the order has expired due to STP)
  "v":122                        // preventedMatchId (Appear if the order has expired due to STP)
  "U":2039                       // counterOrderId (Appear if the order has expired due to STP)
  "A":"1.00000000"               // preventedQuantity(Appear if the order has expired due to STP )
  "B":"2.00000000"               // lastPreventedQuantity(Appear if the order has expired due to STP)



}
```

Payload B

```
{
  "e": "listStatus",                //Event Type
  "E": 1564035303637,               //Event Time
  "s": "ETHBTC",                    //Symbol
  "g": 2,                           //OrderListId
  "c": "OCO",                       //Contingency Type
  "l": "EXEC_STARTED",              //List Status Type
  "L": "EXECUTING",                 //List Order Status
  "r": "NONE",                      //List Reject Reason
  "C": "F4QN4G8DlFATFlIUQ0cjdD",    //List Client Order ID
  "T": 1564035303625,               //Transaction Time
  "O": [                            //An array of objects
    {
      "s": "ETHBTC",                //Symbol
      "i": 17,                      // orderId
      "c": "AJYsMjErWJesZvqlJCTUgL" //ClientOrderId
    },
    {
      "s": "ETHBTC",
      "i": 18,
      "c": "bfYPSQdLoqAJeNrOr9adzq"
    }
  ]
}
```

Orders are updated with the `executionReport` event.

Check the REST API Documentation and below for relevant enum definitions.

Average price can be found by doing `Z` divided by `z`.

Execution types:

- NEW - The order has been accepted into the engine
- CANCELED - The order has been canceled by the user
- REPLACED - This is currently unused
- REJECTED - The order has been rejected and was not processed (this is never pushed into the User Data Stream)
- TRADE - Part of the order or all of the order's quantity has been filled
- EXPIRED - The order was canceled according to the order type's rules (e.g., LIMIT FOK orders with no fill, LIMIT IOC or MARKET orders that partially fill) or by the exchange, (e.g., orders canceled during liquidation, orders canceled during maintenance)
- TRADE_PREVENTION - The order has expired due to STP.

BALANCE UPDATE

Payload

```
{
  "e": "balanceUpdate",         //Event Type
  "E": 1573200697110,           //Event Time
  "a": "BTC",                   //Asset
  "d": "100.00000000",          //Balance Delta
  "T": 1573200697068            //Clear Time
}
```

Balance Update occurs during deposits or withdrawals from the account.