Basic Information

Introduction

This document details all the interface descriptions for accessing KCEX spot trading services, including information on parameter formats, signing, rate limits, error codes, and other interface rules. It also provides detailed explanations of all available market data interfaces, account trading interfaces, and Websocket push interfaces.

API Key

Before accessing, please contact KCEX business liaison personnel to apply for and obtain an API Key. You can only request account trading-related interfaces with the corresponding API Key.

Interface baseURL

Please use this URL to make requests: https://api.kcex.com/

HTTP Return Codes

- HTTP 4XX error codes are used to indicate incorrect request content, behavior, or format. The issue lies with the requester.
- HTTP 401 indicates authentication or permission errors.
- HTTP 403 error code indicates a violation of WAF restrictions (Web Application Firewall).
- HTTP 429 error code signals a warning that the access frequency limit has been exceeded, and the IP may soon be banned.

Request Format

The corresponding API accepts GET, POST, or DELETE type requests.

- For GET method interfaces, parameters must be sent in the query string.
- For POST, PUT, and DELETE method interfaces, parameters can be sent in the query string in content type application/x-www-form-urlencoded, or in the request body. If preferred, you can mix these two methods for sending parameters.
- The order of parameters is not required. However, if the same parameter name is present in both the query string and request body, the one in the query string will be given priority.

Return Format

The return data for all interfaces are in JSON format.

Header Operations

Request Header Signature Parameters

Components	Description
`X-KCEX-APIKEY`	The access key in the API key

Interface Signature Verification

- When calling non-market data interfaces, besides the parameters required by the interface itself, a signature parameter (signature) also needs to be passed in the query string or request body. In batch operation APIs, if the parameter values contain special symbols such as commas, these symbols need to be URL encoded for the signature (note that encoding only supports uppercase).
- The signature uses the HMAC SHA256 algorithm. The API-Secret corresponding to the API-KEY serves as the key for HMAC SHA256, and all other parameters act as operands for HMAC SHA256 to generate the output, which is the signature.
- The signature currently only supports lowercase.
- "totalParams" is defined as the "query string" concatenated with the "request body".

Time Security

> Pseudocode Example

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

- All signature interfaces require the timestamp parameter, which should be the Unix timestamp (in milliseconds) at the moment the request is sent.
- The server checks the timestamp in the request upon receipt. If it was issued more than 5000 milliseconds ago, the request will be considered invalid. This time window can be defined using the optional parameter recvWindow.

The stability and reliability of internet conditions are not guaranteed, thus latency from your local system to the KCEX servers may vary. This is the purpose of setting the recvWindow. If you engage in high-frequency trading and have high requirements for transaction timeliness, you can flexibly set the recvWindow to meet your needs.

It is recommended to use a recvWindow of less than 5 seconds! It must not exceed 60 seconds!

POST /api/v1/order Example

> Example 1

```
HMAC SHA256 signature:

    $ echo -n
"symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000
&timestamp=1644489390087" | openssl dgst -sha256 -hmac
"8e00bb339fa74c019efd3380954b4cc0"
    (stdin)=
323c96ab85a745712e95e63cad28903dd8292e4a905e99c4ee3932023843a117
```

```
curl command:

    (HMAC SHA256)
    $ curl -H "X-KCEX-APIKEY: kc0exknyYvIkVGaIf7" -X POST
'https://api.kcex.com/api/v1/order' -d
'symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000
&timestamp=1644489390087&signature=323c96ab85a745712e95e63cad28903dd829
2e4a905e99c4ee3932023843a117'
```

> Example 2

```
HMAC SHA256 signature:

    $ echo -n
"symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000
&timestamp=1644489390087" | openssl dgst -sha256 -hmac
"8e00bb339fa74c019efd3380954b4cc0"
    (stdin)=
fd3e4e8543c5188531eb7279d68ae7d26a573d0fc5ab0d18eb692451654d837a
```

```
curl command:

    (HMAC SHA256)
```

```
    $ curl -H "X-KCEX-APIKEY: kc0exknyYvIkVGaIf7" -X POST
'https://api.kcex.com/api/v1/order' -d
'symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000
&timestamp=1644489390087&signature=fd3e4e8543c5188531eb7279d68ae7d26a57
3d0fc5ab0d18eb692451654d837a'
```

> Example 3

```
HMAC SHA256 signature:

    $ echo -n
"symbol=BTCUSDT&side=BUY&type=LIMITquantity=1&price=11&recvWindow=5000&
timestamp=1644489390087" | openssl dgst -sha256 -hmac
"8e00bb339fa74c019efd3380954b4cc0"
    (stdin)=
d1a676610ceb39174c8039b3f548357994b2a34139a8addd33baadba65684592
```

```
curl command:

    (HMAC SHA256)
    $ curl -H "X-KCEX-APIKEY: kc0exknyYvIkVGaIf7" -X POST
'https://api.kcex.com/api/v1/order?symbol=BTCUSDT&side=BUY&type=LIMIT'
-d
'quantity=1&price=11&recvWindow=5000&timestamp=1644489390087&signature=
d1a676610ceb39174c8039b3f548357994b2a34139a8addd33baadba65684592'
```

Below is an example of using the echo, openssl, and curl tools in a Linux bash environment to make an API call for placing an order. The apikey and secret are for demonstration purposes only.

Key	Value
apiKey	kc0exknyYvIkVGaIf7
secretKey	8e00bb339fa74c019efd3380954b4cc0

Parameters	Values
symbol	BTCUSDT
side	BUY

type	LIMIT
quantity	1
price	11
recvWindow	5000
timestamp	1644489390087

Example 1: All parameters sent via request body

- requestBody:
  symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000&timestamp=1644489390087

All parameters sent via query string

- queryString:
  symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=1&price=11&recvWindow=5000&timestamp=1644489390087

Example 3: Using both query string and request body

- queryString:
  symbol=BTCUSDT&side=BUY&type=LIMIT
- requestBody:
  quantity=1&price=11&recvWindow=5000&timestamp=1644489390087

Please note, the signature differs from Example 3. There is no '&' between "LIMIT" and "quantity=1".

Rate Limit Rules

Access to the REST API is subject to frequency limits. When these limits are exceeded, a status of 429 is returned: Request Too Frequent.

Interfaces that require an access key for access use the account as the basic unit for rate limiting; interfaces that do not require an access key use the IP as the basic unit for rate limiting.

Rate Limit Explanation

Each interface will indicate whether it is counted by IP, by UID (account), or is exclusive to order placement interfaces, as well as the respective weight of a single request. Different interfaces have different weights, with more resource-intensive interfaces having higher weights.

Rate limiting by IP, by UID (account), and exclusive to order placement interfaces are counted separately, independently of each other.

- For interfaces with IP-based rate limiting, all interfaces share a limit of 300 every 10 seconds;

- For interfaces with UID-based rate limiting, all interfaces share a limit of 200 every 10 seconds;
- For interfaces exclusive to order placement, all interfaces share a limit of 2000 every 10 seconds.

Error Codes
Below are the possible error codes that may be returned by the interfaces:

Error Code	Description
10072	Invalid access key
10073	Invalid request time
30000	Current trading pair has suspended trading
30001	Current trading direction does not allow orders
30002	Less than the minimum trading amount
30003	Greater than the maximum trading amount
30004	Insufficient holdings
30005	Exceeds sell limit
30010	Order price exceeds range
30016	Pause trading
30019	API market orders not enabled
30020	Restricted trading pair, not supported for API access
30021	Invalid trading pair
700001	Invalid API key format
700002	Invalid request signature
700003	This request's timestamp is outside the recvWindow

700004	Necessary parameters not passed or in wrong format
700005	recvWindow must be less than 60000
700006	IP not whitelisted
700007	No permission to access the endpoint
700008	Parameter format does not meet validation rules
730001	Trading pair does not exist
730002	Illegal parameter
730000	Request failed, please contact customer service
730001	User information error
730002	Parameter error, please check
730003	Unsupported operation, please contact customer service

Market Data Interfaces

Test server connectivity

Test whether the Rest API can connect.

> Request Example

```
GET /api/v1/ping
```

> Response Example

```
{}
```

HTTP Request

- **GET** `/api/v1/ping`

Weight (IP): **1**

Request Parameters

NONE

Return Parameters

NONE

Get server time

Get the current timestamp of the server

> Request Example

```
GET /api/v1/time
```

> Response Example

```
{
    "serverTime" : 1645539742000
}
```

HTTP Request

- **GET** `/api/v1/time`

Weight (IP): **1**

Request Parameters

NONE

Trading Specifications Information

Get trading rules and trading pair information.

> Request Example

```
GET /api/v1/exchangeInfo?symbol=BTCUSDT
```

> Response Example


```
{
    "timezone": "CST",
    "serverTime": 1757256965785,
    "rateLimits": [],
    "exchangeFilters": [],
    "symbols":
{
            "symbol": "BTCUSDT",
            "status": "ENABLED",
            "baseAsset": "BTC",
            "baseAssetPrecision": 8,
            "quoteAsset": "USDT",
            "quotePrecision": 4,
            "quoteAssetPrecision": 8,
            "baseCommissionPrecision": 8,
            "quoteCommissionPrecision": 4,
            "orderTypes": [
                "LIMIT",
                "LIMIT_MAKER"
            ],
            "quoteOrderQtyMarketAllowed": false,
            "isSpotTradingAllowed": false,
            "quoteAmountPrecision": "5",
            "baseSizePrecision": "0.0001",
            "permissions": [
                "SPOT"
            ],
            "filters": [],
            "maxQuoteAmount": "5000000",
            "makerCommission": "0.001",
            "takerCommission": "0.001"
```

```
        }
}
```

HTTP Request

- **GET** `/api/v1/exchangeInfo`

Weight (IP): **10**

Request Parameters

Three Usage

Usage	Example
No trading pair required	curl -X GET "https://api.kcex.com/api/v3/exchangeInfo"
Single trading pair	curl -X GET "https://api.kcex.com/api/v3/exchangeInfo?symbol=BTCUSDT"
Multiple trading pairs	curl -X GET "https://api.kcex.com/api/v3/exchangeInfo?symbols=ETHUSDT,BTCUSDT"

Return Parameters

Parameter Name	Data Type	Description
timezone	string	Time zone
serverTime	long	Server time
rateLimits	Array	Rate limits
exchangeFilters	Array	Filters
symbol	String	Trading pair
status	String	Status
baseAsset	String	Trading coin
baseAssetPrecision	Int	Trading coin precision
quoteAsset	String	Quote currency
quotePrecision	Int	Quote currency price precision
quoteAssetPrecision	Int	Quote currency asset precision

baseCommissionPrecision	Int	Trading coin fee precision
quoteCommissionPrecision	Int	Quote currency fee precision
orderTypes	Array	Order types
quoteOrderQtyMarketAllowed	Boolean	Whether market orders are allowed
isSpotTradingAllowed	Boolean	Whether API spot trading is allowed
permissions	Array	Permissions
maxQuoteAmount	String	Maximum order amount
makerCommission	String	Maker fee
takerCommission	String	Taker fee
quoteAmountPrecision	string	Minimum order amount
baseSizePrecision	string	Minimum order quantity

Depth Information

Get the depth information for a specified trading pair, default returns 100 bids and asks.

> Request Example

```
GET /api/v1/depth?symbol=BTCUSDT&limit=200
```

> Response Example

```
{

  "lastUpdateId": 1377043284,
  "bids": [
        ["30225.77","2.132868"],
        ],
  "asks": [
        ["30225.80","1.130244"],
        ],
}
```

HTTP Request

- **GET** `/api/v1/depth`

Weight (IP): **1**

Request Parameters

Parameter Name	Data Type	Required	Description	Range
symbol	string	Yes	Trading pair name	e.g.:BTCUSDT
limit	integer	No	Number of returns	Default 100; Max 5000

Return Parameters

Parameter Name	Data Type	Description
lastUpdateId	long	Latest update ID
bids	list	Bid book [price, order quantity]
asks	list	Ask book [price, order quantity]

Recent Transaction List

Get recent transaction information for a specified trading pair, default returns the last 500 transactions.

> Request Example

```
GET /api/v1/trades?symbol=BTCUSDT&limit=600
```

> Response Example

```
[
```

```
  {
    "id": null,
    "price": "29919.62",
    "qty": "1.292918",
    "quoteQty": "38683.61525116",
    "time": 1652848049876,
    "isBuyerMaker": true
  },
]
```

HTTP Request

- **GET** `/api/v1/trades`

Weight (IP): **1**

Request Parameters

Parameter Name	Data Type	Required	Description	Range
symbol	string	Yes	Trading pair name	e.g.: BTCUSDT
limit	integer	No	Number of returns	Default 500; Max 1000

Return Parameters

Parameter Name	Description
id	Transaction ID
price	Price
qty	Quantity
quoteQty	Transaction amount
time	Transaction time
isBuyerMaker	Whether a maker order

Recent Aggregated Trades

The difference between aggregated trades and tick-by-tick trades is that trades at the same price, in the same direction, and at the same time will be aggregated into one entry.

> Request Example

```
GET /api/v1/aggTrades?symbol=BTCUSDT
```

> Response Example

```
[
  {
    "p": "29881.4",
    "q": "0.010068",
    "T": 1652848230000,
    "m": false
  },
]
```

HTTP Request

- **GET** `/api/v1/aggTrades`

Weight (IP): **1**

Request Parameters

Parameter Name	Data Type	Required	Description	Range
symbol	string	Yes	Trading pair name | e.g.: BTCUSDT	
startTime	long	No	Start returning results from this moment	
endTime	long	No	Return results up to this moment	
limit	integer	No	Number of returns	Default 500; Max 1000

Note: startTime and endTime must be used simultaneously.

Return Parameters

Parameter Name	Description
p	Transaction price

q	Transaction volume
T	Transaction time
m	Whether a sell order

K-line Data

Get K-line data for a specified trading pair; each K-line represents a trading pair. The open time of each K-line can be considered as a unique ID.

> Request Example

```
GET
/api/v1/klines?symbol=BTCUSDT&interval=1m&startTime=1652848049876&endTi
me=1652848650458
```

> Response Example

```
[
  [
    1652818380000,
    "30082.28",
    "30105.66",
    "30082.28",
    "30084.65",
    "5.838067",
    1652818440000,
    "175741.13"
  ],
]
```

HTTP Request

- **GET** `/api/v1/klines`

Weight (IP): **1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pair name | e.g.: BTCUSDT
interval	ENUM	Yes	See Enum definition: K-line interval | e.g.: 1m
startTime	long	No	e.g.: 1652848049876
endTime	long	No	e.g.: 1652848650458
limit	integer	No	Default 500; Max 1000

Note: startTime and endTime must be used simultaneously.

Return Parameters

Index	Description
0	Open time
1	Open price
2	Highest price
3	Lowest price
4	Close price
5	Volume
6	Close time
7	Transaction amount

24-hour Price Movement

Get the 24-hour price movement of a specified trading pair or all trading pairs (in 5-minute intervals).

> Request Example

```
GET /api/v1/ticker/24hr?symbol=BTCUSDT
```

> Response Example


```
{
    "symbol": "BTCUSDT",
    "baseCurrency": "BTC",
    "targetCurrency": "USDT",
    "priceChange": "1588.47",
    "priceChangePercent": "0.07791949",
    "prevClosePrice": "20386.04",
    "lastPrice": "21974.51",
    "bidPrice": "21974.48",
    "bidQty": "0.645732",
    "askPrice": "21974.51",
    "askQty": "5.801688",
    "openPrice": "20386.04",
    "highPrice": "22508.06",
    "lowPrice": "20269.12",
    "volume": "6381.884246",
    "quoteVolume": "135594952.21",
    "openTime": 1657258200000,
    "closeTime": 1657258407860
}
```


HTTP Request

- **GET** `/api/v1/ticker/24hr`

Weight (IP): **5**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	No	Trading pair name | Do not pass to check all (use with caution) | e.g.:

			BTCUSDT

Return Parameters

Parameter Name	Description
symbol	Trading pair
baseCurrency	Trading currency
targetCurrency	Quote currency
priceChange	Price change
priceChangePercent	Price change percentage
prevClosePrice	Previous close price
lastPrice	Last Price
bidPrice	Bid price
bidQty	Bid quantity
askPrice	Ask price
askQty	Ask quantity
openPrice	Open price
highPrice	Highest price
lowPrice	Lowest price
volume	Volume
quoteVolume	Transaction amount
openTime	Open time
closeTime	Close time

Last Price

Get the last price of a specified trading pair or all trading pairs.

> Request Example

```
GET /api/v1/ticker/price?symbol=BTCUSDT
```

> Response Example

```
{
    "symbol": "BTCUSDT",
    "price": "29805.02"
}
```

HTTP Request

- **GET** `/api/v1/ticker/price`

Weight (IP): **5**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	No	Trading pair name | Do not pass to check all | e.g.: BTCUSDT

Return Parameters

Parameter Name	Description
symbol	Trading pair
price	Last price

Current Best Order

Retrieve the current best order (highest bid, lowest ask)

> Request Example

```
GET /api/v1/ticker/bookTicker?symbol=BTCUSDT
```

> Response Example

```
{
    "symbol": "BTCUSDT",
    "bidPrice": "29820.79",
    "bidQty": "2.241948",
    "askPrice": "29820.82",
    "askQty": "2.301948"
}
```

HTTP Request

- **GET** `/api/v1/ticker/bookTicker`

Weight (IP): **5**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	No	Trading pair name | Do not pass to check all | e.g.: BTCUSDT

Return Parameters

Parameter Name	Description
symbol	Trading pair
bidPrice	Highest bid price
bidQty	Highest bid quantity
askPrice	Lowest ask price
askQty	Lowest ask quantity

Spot Account and Trading Interfaces

Test Order

Used to test the order request, but it will not be submitted to the matching engine.

> Request Example

```
POST /api/v1/order/test
```

> Response Example

```
{}
```

HTTP Request

- **POST** `/api/v1/order/test`

Weight (UID): **1**

Request Parameters

Same as POST /api/v1/order

Place Order

Only when your account has sufficient funds can you place an order.

> Request Example

```
POST
/api/v1/order?symbol=BTCUSDT&side=BUY&type=LIMIT&quantity=50&price=6000
0&timestamp={{timestamp}}&signature={{signature}}
```

> Response Example

```
{
    "symbol": "BTCUSDT",
    "orderId": "C02__407731549153275904022",
    "price": "60000",
    "origQty": "50",
    "type": "LIMIT",
    "side": "BUY",
    "transactTime": 1766676533741
}
```

HTTP Request

- **POST** `/api/v1/order`

Weight (Order-specific): **1**

Request Parameters

Name	Type	Required	Description
symbol	string	Yes	Trading pair
side	ENUM	Yes	See enum definition: Order direction
type	ENUM	Yes	See enum definition: Order type
quantity	decimal	No	Order quantity
quoteOrderQty	decimal	No	Total order amount
price	decimal	No	Order price
newClientOrderId	string	No	Custom unique order ID from the client
recvWindow	long	No	The value cannot be greater than 60000
timestamp	long	Yes	

Based on different order types, some parameters are mandatory:

Type	Mandatory Parameters
`LIMIT`	`quantity`, `price`
`MARKET`	`quantity or quoteOrderQty`

Other explanations:

MARKET: When the type is market, if it's a buy order, then quoteOrderQty is a mandatory parameter.
If it's a sell order, quantity is a mandatory parameter,

- For example, placing a market buy order on BTCUSDT explicitly specifies the amount of quote asset you want to spend. At this time, the order quantity will be calculated based on market liquidity and quoteOrderQty (the actual transaction quantity is subject to the final order details).
  For example, on BTCUSDT, quoteOrderQty=100: when placing a buy order, the order will buy as much BTC worth 100USDT as possible.
- For example, placing a market sell order on BTCUSDT, quantity specifies how much BTC the user can sell.

Return Parameters

Parameter Name	Data Type	Description
symbol	string	Trading pair
orderId	string	Order ID
price	string	Order ID
origQty	string	Order quantity
type	string	Order type
side	string	Order direction
transactTime	long	Order time

Batch Orders

Support single batch of up to 20 orders, must be the same trading pair.

> Request Example

```
POST /api/v1/batchOrders?batchOrders=[{"type": "LIMIT","price":
"60000","quantity": "0.0002","symbol": "BTCUSDT","side":
"BUY","newClientOrderId": 29088234},{"type": "LIMIT","price":
"60000","quantity": "0.0003","symbol": "BTCUSDT","side": "SELL"}]
```

> Response Example

```
{ //Successful return:
  [
    {
      "symbol": "BTCUSDT",
      "orderId": "C02__407731549153275904022"
    },
    {
      "symbol": "BTCUSDT",
      "orderId": "C02__407731549153275904023"
    }
  ],
  //Return with failures:
  [
    {
      "symbol": "BTCUSDT",
      "orderId": "C02__407731549153275904022",
      "newClientOrderId": "abncc1",
    },
    {
      "newClientOrderId": "abncc2",
      "msg": "The minimum transaction volume cannot be less than:
5USDT",
      "code": 30002
    },
    {
      "symbol": "BTCUSDT",
      "orderId": "C02__407731549153275904023"
    }
  ]
}
```

HTTP Request

- **POST** `/api/v1/batchOrders`

Weight (Order-specific):**20**

Request Parameters

Name	Type	Required	Description
batchOrders	LIST	Yes	List of orders, up to 20 orders (list of JSON format, see request example for order parameters)
symbol	string	Yes	Trading pair
side	ENUM	Yes	See enum definition: Order direction

type	ENUM	Yes	See enum definition: Order type
quantity	decimal	No	Order quantity
quoteOrderQty	decimal	No	Total order amount
price	decimal	No	Order price
newClientOrderId	string	No	Custom unique order ID from the client
recvWindow	long	No	The value cannot be greater than 60000
timestamp	long	Yes	

Based on different order types, some parameters are mandatory:

Type	Mandatory Parameters
`LIMIT`	`quantity`, `price`
`MARKET`	`quantity or quoteOrderQty`

Return Parameters

Parameter Name	Data Type	Description
symbol	string	Trading pair
orderId	string	Order ID

Cancel Order

Cancel valid orders.

> Request Example

```
DELETE
/api/v1/order?symbol=BTCUSDT&orderId=C02__407731549153275904022&timesta
mp={{timestamp}}&signature={{signature}}
```

> Response Example

```
{
  "symbol": "BTCUSDT",
  "origClientOrderId": "myOrder1",
  "orderId": "C02__407731549153275904022",
  "orderListId": -1, // OCOorderID, otherwise -1
  "clientOrderId": "myOrder1",
  "price": "29000.0000",
  "origQty": "1.00000000",
  "executedQty": "0.00000000",
  "cummulativeQuoteQty": "0.00000000",
  "status": "CANCELED",
  "timeInForce": "GTC",
  "type": "LIMIT",
  "side": "BUY"
}
```

HTTP Request

- **DELETE** `/api/v1/order`

Weight (UID):**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pair name
orderId	string	No	Order ID
origClientOrderId	string	No	Initial custom order ID
newClientOrderId	string	No	Custom unique order ID from the client
recvWindow	long	No	
timestamp	long	Yes	

orderId or origClientOrderId must be sent at least one.

Return Parameters

Parameter Name	Description
symbol	Trading pair

origClientOrderId	Original client order ID
orderId	Order ID
clientOrderId	Client ID
price	Price
origOty	Initial quantity
executedQty	Executed quantity
cummulativeQuoteQty	Executed amount
status	Current status
timeInForce	Order validity method
type	Order type
side	Order direction

Cancel All Orders for a Single Trading Pair

Cancel all pending orders for a single trading pair.

> Request Example

```
DELETE
/api/v1/openOrders?symbol=BTCUSDT&timestamp={{timestamp}}&signature={{s
ignature}}
```

> Response Example

```
[
  {
    "symbol": "BTCUSDT",
    "origClientOrderId": "clientOrderId1",
    "orderId": "C02__407731549153275904022",
    "clientOrderId": "clientOrderId1",
    "price": "0.089853",
    "origQty": "0.178622",
    "executedQty": "0.000000",
```

```
    "cummulativeQuoteQty": "0.000000",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY"
  },
  {
    "symbol": "BTCUSDT",
    "origClientOrderId": "clientOrderId2",
    "orderId": "C02__407731549153275904023",
    "clientOrderId": "clientOrderId2",
    "price": "0.090430",
    "origQty": "0.178622",
    "executedQty": "0.000000",
    "cummulativeQuoteQty": "0.000000",
    "status": "CANCELED",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY"
  }
]
```


HTTP Request

- **DELETE** `/api/v1/openOrders`

Weight (UID):**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pair
recvWindow	long	No	
timestamp	long	Yes	

Return Parameters

Parameter Name	Description
symbol	Trading pair
origClientOrderId	Original client order ID
orderId	Order ID
clientOrderId	Client ID
price	Price
origQty	Initial quantity

executedQty	Executed quantity
cummulativeQuoteQty	Executed amount
status	Status
timeInForce	Order validity method
type	Order type
side	Order direction

Query Order

Query the status of a specified trading pair's order, can query orders within the last 7 days, orders older than 7 days can be viewed on the web client.

> Request Example

```
GET
/api/v1/order?symbol=BTCUSDT&orderId=C02__407731549153275904023&timesta
mp={{timestamp}}&signature={{signature}}
```

> Response Example


```
{
   "symbol": "BTCUSDT",
   "orderId": "418985996516855808020000005",
   "clientOrderId": "111211cxx",
   "price": "60000",
   "origQty": "0.01",
   "executedQty": "0.01",
   "cummulativeQuoteQty": "611.9566",
   "status": "FILLED",
   "timeInForce": null,
   "type": "LIMIT",
   "side": "SELL",
   "stopPrice": null,
   "time": 1715676282000,
   "updateTime": 1715676282000,
   "isWorking": true,
   "origQuoteOrderQty": "600"
```

```
}
```

HTTP Request

- **GET** `/api/v1/order`

Weight (UID):**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	Trading pair	Yes	
origClientOrderId	Original client order ID	No	
orderId	Order ID	No	
recvWindow	long	No	
timestamp	long	Yes	

Return Parameters

Parameter Name	Description
symbol	Trading pair
orderId	System order ID
clientOrderId	Custom client ID
price	Price
origQty	Original order quantity
executedQty	Executed order quantity
cummulativeQuoteQty	Cumulative order amount
status	Order status
timeInForce	Order's time effectiveness
type	Order type
side	Order direction
stopPrice	Stop-loss price
time	Order time

updateTime	Last update time
isWorking	Whether in orderbook
origQuoteOrderQty	Original transaction amount

Current Orders

Retrieve current orders, supporting queries for multiple trading pairs, with a maximum of 5 symbols per request.
When querying 5 trading pairs in bulk, only up to 1000 orders will be returned.

> Request Example

```
GET
/api/v1/openOrders?symbol=BTCUSDT&timestamp={{timestamp}}&signature={{s
ignature}}
```

> Response Example

```
[
   {
       "symbol": "BTCUSDT",
       "orderId": "418985671261163520020000005",
       "clientOrderId": "111211c",
       "price": "70000",
       "origQty": "0.01",
       "executedQty": "0",
       "cummulativeQuoteQty": "0",
       "status": "NEW",
       "timeInForce": null,
       "type": "LIMIT",
       "side": "SELL",
       "stopPrice": null,
       "time": 1715676204272,
       "updateTime": null,
       "isWorking": true,
       "origQuoteOrderQty": "700"
   }
]
```

HTTP Request

- **GET** `/api/v1/openOrders`

Weight (UID)：**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pairs【Up to 5 symbols can be passed, represented by a string separated by "". e.g. "BTCUSDTADAUSDT"】
recvWindow	long	No	
timestamp	long	Yes	

Return Parameters

Parameter Name	Description
symbol	Trading pair
orderId	System order ID
clientOrderId	Custom client ID
price	Price
origOty	Original order quantity
executedQty	Executed order quantity
cummulativeQuoteQty	Cumulative order amount
status	Order status
timeInForce	Order's time effectiveness
type	Order type
side	Order direction
stopPrice	Stop-loss price
time	Order time
updateTime	Last update time
isWorking	Whether in orderbook
origQuoteOrderQty	Original transaction amount

Query All Orders

Get all valid, canceled, or completed orders for the account (query period defaults to the last 24 hours), can query up to the last 7 days of data.

> Request Example

```
GET
/api/v1/allOrders?symbol=BTCUSDT&timestamp={{timestamp}}&signature={{si
gnature}}
```

> Response Example

```
[
   {
       "symbol": "BTCUSDT",
       "orderId": "418985996516855808020000005",
       "clientOrderId": "111211cxx",
       "price": "60000",
       "origQty": "0.01",
       "executedQty": "0.01",
       "cummulativeQuoteQty": "611.9566",
       "status": "FILLED",
       "timeInForce": null,
       "type": "LIMIT",
       "side": "SELL",
       "stopPrice": null,
       "time": 1715676282000,
       "updateTime": 1715676282000,
       "isWorking": true,
       "origQuoteOrderQty": "600"
   }
]
```

HTTP Request

- **GET** `/api/v1/allOrders`

Weight (UID):**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pair
startTime	long	No	
endTime	long	No	
limit	int	No	Default 500; Max 1000;
recvWindow	long	No	
timestamp	long	Yes	

startTime and endTime need to be used simultaneously.

Return Parameters

Parameter Name	Description
symbol	Trading pair
orderId	System order ID
clientOrderId	Custom client ID
price	Price
origOty	Original order quantity
executedQty	Executed order quantity
cummulativeQuoteQty	Cumulative order amount
status	Order status
timeInForce	Order's time effectiveness
type	Order type
side	Order direction
stopPrice	Stop-loss price
time	Order time
updateTime	Last update time
isWorking	Whether in orderbook
origQuoteOrderQty	Original transaction amount

Account Information

Get current account information.

> Request Example

```
GET /api/v1/account?timestamp={{timestamp}}&signature={{signature}}
```

> Response Example

```
{
   "makerCommission": 200,
   "takerCommission": 400,
   "canTrade": true,
   "canWithdraw": true,
   "canDeposit": true,
   "accountType": "SPOT",
   "balances": [
       {
           "asset": "USDT",
           "free": "10000",
           "locked": "0"
       },
       {
           "asset": "BTC",
           "free": "10000",
           "locked": "0"
       }
   ],
   "permissions": [
       "SPOT"
   ]
}
```

HTTP Request

- **GET** `/api/v1/account`

Weight (UID):**5**

Request Parameters

Parameter Name	Data Type	Required	Description

recvWindow	long	No	
timestamp	long	Yes	

Return Parameters

Parameter Name	Description
makerCommission	Maker fee rate
takerCommission	Taker fee rate
canTrade	Whether can trade
canWithdraw	Whether can withdraw
canDeposit	Whether can deposit
accountType	Account type
balances	Balance
asset	Asset currency
free	Available quantity
locked	Frozen quantity
permissions	Permissions

Account Transaction History

Get account-specific trading pair transaction history, can only query recent 1 month of transaction records.

> Request Example

```
GET
/api/v1/myTrades?symbol=BTCUSDT&timestamp={{timestamp}}&signature={{sig
nature}}
```

> Response Example

```
[
   {
       "symbol": "BTCUSDT",
       "id": "c44c2ff3409d461cae7804878cced647",
       "orderId": "418985996516855808020000005",
       "price": "61195.66",
       "qty": "0.01",
       "quoteQty": "611.9566",
       "commission": "30.59783",
       "commissionAsset": "USDT",
       "time": 1715676282000,
       "isBuyer": false,
       "isMaker": false,
       "isBestMatch": true,
       "isSelfTrade": false,
       "clientOrderId": "111211cxx"
   }
]
```


HTTP Request

- **GET** `/api/v1/myTrades`

Weight (UID):**1**

Request Parameters

Parameter Name	Data Type	Required	Description
symbol	string	Yes	Trading pair
orderId	string	No	Must be used together with symbol
startTime	long	No	
endTime	long	No	
limit	int	No	Default 500; Max 1000;
recvWindow	long	No	
timestamp	long	Yes	

Return Parameters

Parameter Name	Description
symbol	Trading pair
id	Transaction ID

orderId	Order ID
price	Price
qty	Quantity
quoteQty	Transaction amount
time	Transaction time
commission	Fees
commissionAsset	Fess currency
isMaker	Whether maker
isBuyer	Whether buyer
isBestMatch	Whether best match
isSelfTrade	Whether self trade
clientOrderId	Custom client ID

Available Trading Pairs

Query current account's available trading pairs.

> Request Example

```
GET /api/v1/selfSymbols?timestamp={{timestamp}}&signature={{signature}}
```

> Response Example

```
[
   "BOMEUSDT",
   "AVAXUSDT",
   "BTCUSDT",
   "BUSDUSDT",
   "ADAUSDT",
   "AVAXBTC",
   "WETHUSDT",
   "BNBUSDT",
   "PEPEUSDT",
   "HBARUSDT"
]
```

HTTP Request

- **GET /api/v1/selfSymbols**

Weight (UID)：1

Request Parameters

Parameter Name	Data Type	Required	Description
recvWindow	long	No	
timestamp	long	Yes	

Return Parameters [string]

Websocket Market Data Push

- All wss interfaces listed in this section have the baseurl: wss://wbs.kcex.com/ws
- Each connection to wbs.kcex.com has a validity period of no more than 24 hours, please handle disconnections and reconnections properly
- In the symbol names, all trading pairs are uppercase, for example: spot@public.deals.v3.api@<symbol> Example: spot@public.deals.v3.api@BTCUSDT
- If there is no valid subscription on a websocket, the server will actively disconnect after 30 seconds, if there is a subscription but no traffic, the server will disconnect after 30 seconds, the client can send a ping to keep the connection
- A maximum of 30 subscriptions per ws connection

Real-time Subscribe/Unsubscribe Data Stream

- The following data can be sent via websocket to subscribe or unsubscribe to a data stream. Examples are as follows.
- The id in the response content is an unsigned integer, serving as a unique identifier for the messages.
- If the msg in the corresponding content is the corresponding request field, it indicates the request was sent successfully.

Subscribe to a Data Stream

> **subscription channel response**

```
{
 "id":0,
 "code":0,
 "msg":"spot@public.deals.v3.api@BTCUSDT"
}
```

- **Request**

```
{
 "method":"SUBSCRIPTION",
 "params":["spot@public.deals.v3.api@BTCUSDT"]
}
```

Unsubscribe from a Data Stream

> **Unsubscribe response**

```
 {
  "id":0,
  "code":0,
"msg":"spot@public.increase.depth.v3.api@BTCUSDT,spot@public.deals.v3.a
pi@BTCUSDT"
 }
```

- **Request**

```
{
 "method":"UNSUBSCRIPTION",

"params":["spot@public.deals.v3.api@BTCUSDT","spot@public.increase.depth.v3.api@BT
CUSDT"]
}
```

PING/PONG Mechanism

> **PING/PONG response**

```
 {
  "id":0,
  "code":0,
  "msg":"PONG"
 }
```

- **Request**

```
{"method":"PING"}
```

Tick-by-tick Trades (Real-time)

request:

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@public.deals.v3.api@BTCUSDT"
    ]
}
```

response:

```
{
    "c":"spot@public.deals.v3.api@BTCUSDT",
    "d":{
            "deals":[{
                    "S":2,                                  //tradeType
                    "p":"20233.84",                         //filled price
                    "t":1661927587825,                      //dealTime
                    "V":"0.001028"                          //filled
quantity
                    }],
            "e":"spot@public.deals.v3.api"                  //eventType
        },
    "s":"BTCUSDT",                                          //trading pair
symbol
    "t":1761927587836                                       //eventTime
}
```

Request Parameters: `spot@public.deals.v3.api@<symbol>`

Tick-by-tick trade push provides information for each transaction. A transaction, or trade, is defined as only having one taker and one maker interacting.

Return Parameters:

Parameter Name	Data Type	Description
deals	array	transaction information
> S	int	trade type 1: buy 2: sell
> p	string	filled price
> t	long	filled time
> v	string	filled quantity

e	string	event type
s	string	trading pair
t	long	event time

K-line Streams

request:

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@public.kline.v3.api@BTCUSDT@Min15"
    ]
}
```

response:

```
{
    "c":"spot@public.kline.v3.api@BTCUSDT@Min15",
    "d":{
            "k":{
                "T":1661931900,      //End time of this K-line
                "a":29043.48804658,     //Transaction amount during
this K-line
                "c":20279.43,                //Last transaction price
during this K-line

                "h":20284.93,                //Highest transaction
price during this K-line
                "i":"Min15",                //K-line interval
                "l":20277.52,                //Lowest transaction
price during this K-line
                "o":20284.93,                //First transaction price
during this K-line
                "s":"BTCUSDT",             //Trading pair
                "t":1761931000,             //Start time of this K-line
                "v":1.43211},                //Transaction volume
during this K-line
            "e":"spot@public.kline.v3.api"},           //event type
```

```
    "s":"BTCUSDT",                          //Trading pair
    "t":1761931016878                      //event time
}
```

K-line pushed per second provides updates for the requested K-line category (latest K-line).

Request Parameters: `spot@public.kline.v3.api@<symbol>@<interval>`

Return Parameters:

Parameter Name	Data Type	Description
e	string	event type
k	object	k-line information
> T	long	End time of this K-line
> a	bigDecimal	Transaction amount during this K-line
> c	bigDecimal	Last transaction price during this K-line
> h	bigDecimal	Highest transaction price during this K-line
> i	interval	K-line interval
> l	bigDecimal	Lowest transaction price during this K-line
> o	bigDecimal	First transaction price during this K-line
> t	long	Start time of this K-line
> v	bigDecimal	Transaction volume during this K-line
s	string	trading pair
t	long	event time

K-line interval parameters:

Min -> minutes; Hour -> hours; Day -> days; Week -> weeks, M -> month

- Min1
- Min5
- Min15
- Min30
- Min60

- Hour4
- Hour8
- Day1
- Week1
- Month1

Incremental Depth Information (Real-time)

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@public.increase.depth.v3.api@BTCUSDT"
    ]
}
```

> **response:**

```
{
    "c":"spot@public.increase.depth.v3.api@BTCUSDT",
    "d":{
        "asks":[{                   //bids: buy orders, asks: sell
orders
            "p":"20290.89",         //price tier changed
            "v":"0.000000"}],       //quantity
            "e":"spot@public.increase.depth.v3.api", },     //event
type    "s":"BTCUSDT",                              //trading pair
    "t":1661932660144                         //event time
}
```

If the order quantity (v) at a certain price is 0, it indicates that the order at that price has been canceled or filled, and the price level should be removed.

Request Parameters: `spot@public.increase.depth.v3.api@<symbol>`

Return Parameters:

Parameter Name	Data Type	Description

p	string	Price tier changed
v	string	Quantity
e	string	Event type
s	string	Trading pair
t	long	event time

Limited Tier Depth Information

Push limited tier depth information. The "levels" parameter indicates the number of buy and sell tiers, with options for 5, 10, or 20 tiers.

request:

```
{
    "method": "SUBSCRIPTION",
    "params": [
                "spot@public.limit.depth.v3.api@BTCUSDT@5"

   ]
}
```

response:

```
{
  "c":"spot@public.limit.depth.v3.api@BTCUSDT@5",
  "d":{
    "asks":[{                  //bids: buy orders, asks: sell orders
            "p":"20290.89",    //changed price tiers
            "v":"0.000000"}], //quantity
    "e":"spot@public.limit.depth.v3.api",  //event type
    "r":"3407459756"},  //version number
  "s":"BTCUSDT",               //trading pair
  "t":1661932660144            //event time
}
```

Request Parameters: `spot@public.limit.depth.v3.api@<symbol>@<level>`

Return Parameters:

Parameter Name	Data Type	Description

p	string	Changed price tiers
v	string	Quantity
e	string	Event type
r	string	Version number
s	string	Trading pair
t	long	Event time

Best Orders Information by Symbol

Real-time push of the best orders information for specified trading pairs.

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
                "spot@public.bookTicker.v3.api@BTCUSDT"
      ]
}
```

> **response:**

```
{
 "c":"spot@public.bookTicker.v3.api@<BTCUSDT>",
 "d":{
    "A":"40.66000000"},  //Best sell order quantity
    "B":"31.21000000",  //Best buy order quantity
    "a":"25.36520000", //Best sell order price
    "b":"25.35190000", // Best buy order price
  },
 "s":"BTCUSDT", //trading pair
 "t":1661932660144 //event time
}
```

Request Parameters: `spot@public.bookTicker.v3.api@<symbol>`

Return Parameters:

Parameter Name	Data Type	Description
A	string	Best sell order quantity
B	string	Best buy order quantity
a	string	Best sell order price
b	string	Best buy order price
s	string	Trading pair
t	long	Event time

MiniTicker

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@public.miniTicker.v3.api@BTCUSDT@UTC+8"
    ]
}
```

> **response:**

```
{
  "d":
   {"s":"BTCUSDT",
    "p":"36474.74",
    "r":"0.0354",
    "tr":"0.0354",
    "h":"36549.72",
    "l":"35101.68",
    "v":"375173478.65",
    "q":"10557.72895",
    "t":"1699502456050"},
  "c":"spot@public.miniTicker.v3.api@BTCUSDT@UTC+8",
  "t":1699502456051,
  "s":"BTCUSDT"
}
```

Request Parameters: `spot@public.miniTicker.v3.api@<symbol>@<timezone>`

Return Parameters:

Parameter Name	Data Type	Description
c	string	Subscription channel name
d	data	data
>s	string	trading pair symbol
>p	string	Last transaction price
>r	string	Subscription utc8 timezone change rate
>tr	string	Subscription timezone change rate
>h	string	24-hour highest price
>l	string	24-hour lowest price
>v	string	24-hour transaction amount
>q	string	24-hour transaction volume

MiniTickers

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@public.miniTickers.v3.api@UTC+8"
    ]
}
```

> **response:**

```
{
  "d":
  [{"s":"SENSOUSDT",
    "p":"0.07642",
    "r":"-0.0383",
    "tr":"-0.0383",
    "h":"0.08032",
    "l":"0.07463",
    "v":"25052.6533",
    "q":"323777.17"},
   {"s":"BTCUSDT",
    "p":"36474.74",
    "r":"0.0354",
    "tr":"0.0354",
    "h":"36549.72",
    "l":"35101.68",
    "v":"375173478.65",
    "q":"10557.72895"},
  "c":"spot@public.miniTickers.v3.api@UTC+8",
  "t":1699502456051,
}
```

Request Parameters: `spot@public.miniTickers.v3.api@<timezone>`

Return Parameters:

Parameter Name	Data Type	Description
c	string	Subscription channel name
d	data	data
>s	string	trading pair symbol

>p	string	Last transaction price
>r	string	Subscription utc8 timezone change rate
>tr	string	Subscription timezone change rate
>h	string	24-hour highest price
>l	string	24-hour lowest price
>v	string	24-hour transaction amount
>q	string	24-hour transaction volume

How to Properly Maintain a Local Orderbook Copy

1. Subscribe to spot@public.increase.depth.v3.api@ to get full depth information, save the current version.
2. Subscribe to ws depth information, when updates are received, if the received data version>current version at the same price level, the latter update will override the former.
3. Access the Rest interface https://api.kcex.com/api/v3/depth?symbol=BTCUSDT&limit=1000 to obtain a 1000-level depth snapshot
4. Discard data in the current cached depth information at the same price, where the version< the version obtained in step 3 from the snapshot.
5. Update the content of the depth snapshot to the local cache, and continue to update from the events received from ws.
6. Every new event's version should exactly equal the previous event's version+1, otherwise, there may have been packet loss. If packet loss occurs or the obtained event's version is not continuous, please reinitialize from step 3.
7. Each event's order quantity represents the absolute number of orders at that price, not the relative change.
8. If the order quantity at a certain price is 0, it indicates that the order at that price has been canceled or filled, and the price level should be removed.

Note:
Due to the limitation on the number of price tiers in the depth snapshot, price tiers outside the initial snapshot that do not experience quantity changes will not appear in the incremental depth updates. Therefore, even if all updates from the incremental depth are applied, these price tiers will not be visible in the local order book. As a result, there may be some differences between the local order book and the actual order book. However, for most

use cases, the depth limit of 5000 is sufficient to effectively understand the market and execute trades.

WebSocket Account Information Push

- The listenKey for subscribing to account data is valid for 60 minutes from the time of creation.
- You can extend the validity period by 60 minutes by sending a PUT request with the listenKey.
- You can close the current data stream and invalidate the listenKey immediately by sending a DELETE request with the listenKey.
- A POST request to an account with a valid listenKey will return the current valid listenKey and extend its validity period by 60 minutes.
- WebSocket interface base URL: wss://wbs.kcex.com/ws
- The stream name for subscribing to account data is /ws?listenKey=listenKey, e.g., wss://wbs.kcex.com/ws?listenKey=3db6899a87803a478bbd5162c8444aac1162a4750379fb47103182144178d789
- Each link is valid for no more than 24 hours. Please handle disconnections and reconnections properly.
- Each UID can apply for up to 60 listen keys (excluding invalid listen keys).
- WebSocket link limit: Each listen key can have up to 5 WebSocket links (i.e., each UID can apply for up to 60 listen keys, resulting in 300 WebSocket links).

Listen Key (Spot Account)

Generate Listen Key

> **Response**

```
{
  "listenKey":
"3db6899a87803a478bbd5162c8444aac1162a4750379fb47103182144178d789"
}
```

HTTP Request

- **POST** `/api/v1/listen-key/userDataStream?timestamp={{timestamp}}&signature={{signature}}`

Starts a new data stream. Unless a keepalive is sent, the data stream will close after 60 minutes. If the account has a valid listenKey, it will return the listenKey and extend its validity period by 60 minutes.

Parameters:

NONE

Extend Listen Key Validity

> **Response**

```
{
    "listenKey":
"3db6899a87803a478bbd5162c8444aac1162a4750379fb47103182144178d789"
}
```

HTTP Request

- **PUT** `/api/v1/listen-key/userDataStream?timestamp={{timestamp}}&signature={{signature}}&listenKey=`

Extends the validity period by 60 minutes from the time of this call. It is recommended to send a request every 30 minutes.

Request Parameters:

Parameter Name	Data Type	Required	Description
listenKey	string	Yes	

Close Listen Key

> **Response**

```
{
```

```
    "listenKey":
"3db6899a87803a478bbd5162c8444aac1162a4750379fb47103182144178d789"
}
```

HTTP Request

- **DELETE**
  `/api/v1/listen-key/userDataStream?timestamp={{timestamp}}&signature={{signature}}&listenKey=`

Closes the user data stream.

Request Parameters:

Parameter Name	Data Type	Required	Description
listenKey	string	Yes	

Spot Account Information (Real-Time)

Upon successful subscription, the server will push updates on account assets whenever there is a change in balance or available balance.

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@private.account.v3.api"
    ]
}
```

> **response:**

```
{
    "c": "spot@private.account.v3.api",
    "d": {
        "a": "USDT",
        "c": 1678185928428,
```

```
        "f": "302.185113007893322435",
        "fd": "-4.990689704",
        "l": "4.990689704",
        "ld": "4.990689704",
        "o": "ENTRUST_PLACE"
    },
    "t": 1678185928435
}
```

Request Parameters: `spot@private.account.v3.api`

Return Parameters:

Parameter Name	Data Type	Description
d	json	Account information
> a	string	Asset name
> c	long	Settlement time
> f	string	Available balance
> fd	string	Available balance change amount
> l	string	Frozen balance
> ld	string	Frozen balance change amount
> o	string	Change type
t	long	Event time

Account Trades (Real-Time)

> **request:**

```
{
    "method": "SUBSCRIPTION",
    "params": [
        "spot@private.deals.v3.api"
    ]
}
```

response:


```
{
    "c": "spot@private.deals.v3.api",
    "d": {
        "S": 1,
        "T": 1661938980268,
        "c": "",
        "i": "c079b0fcb80a46e8b128b281ce4e4f38",
        "m": 1,
        "p": "1.008",
        "st": 0,
        "t": "4079b1522a0b40e7919f609e1ea38d44",
        "v": "5"
    },
    "s": "BTCUSDT",
    "t": 1761938980285
}
```


Request Parameters: `spot@private.deals.v3.api`

Return Parameters:

Parameter Name	Data Type	Description
d	json	Account trade information Trade type (1: Buy, 2: Sell) Trade time User-defined order ID (clientOrderId) Order ID Is maker order (1: Yes, 0: No) Trade price Is self-trade (0: No, 1: Yes) Trade ID Trade quantity Trading pair Event time
> S	int	Trade type (1: Buy, 2: Sell)
> T	long	Trade time
> c	string	User-defined order ID (clientOrderId)
> i	string	Order ID: orderId
> m	int	Whether is Maker Order:

		isMaker
> p	string	Transaction pirce
> st	byte	Whether is Self Trade: isSelfTrade
> t	string	Trade ID: tradeId
> v	string	Trade quantity
s	string	Trading pair
t	long	Event time

Account Orders (Real-Time)

> **request:**

```
{
  "method": "SUBSCRIPTION",
  "params": [
      "spot@private.orders.v3.api"
  ]
}
```

Request Parameters: `spot@private.orders.v3.api`

> **response:**

```
{
  "c": "spot@private.orders.v3.api",
  "d": {
        "A":8.0,
        "O":1661938138000,
        "S":1,
        "V":10,
        "a":8,
        "c":"",
        "i":"C02__407731549153275904023",
        "m":0,
        "o":1,
```

```
        "p":0.8,
        "s":1,
        "v":10
  },
  "s": "BTCUSDT",
  "t": 1661938138193
}
```

Return Parameters:

Parameter Name	Data Type	Description
d	json	Account order information
> A	bigDecimal	Actual remaining amount (remainAmount)
> O	long	Order creation time
> S	int	Trade type (1: Buy, 2: Sell)
> V	bigDecimal	Actual remaining quantity (remainQuantity)
> a	bigDecimal	Total order amount
> c	string	User-defined order ID (clientOrderId)
> i	string	orderid
> m	int	Whether is Maker Order: isMaker
> o	int	Order type LIMIT_ORDER(1),POST_ONLY(2),IMMEDIATE_OR_CANCEL(3),FILL_OR_KILL(4),MARKET_ORDER(5);
> p	bigDecimal	Order price
> s	int	Order status (1: Unfilled, 2: Filled, 3: Partially filled, 4: Canceled, 5: Partially canceled)
> v	bigDecimal	Order quantity
t	long	Event time
s	string	Trading pair

Public API Parameters

Enumeration Definitions

Order Direction

- BUY
- SELL

Order Type

- LIMIT Limit order
- MARKET Market order
- LIMIT_MAKER Limit maker order
- IMMEDIATE_OR_CANCEL IOC order (parts that cannot be executed immediately are canceled, the order will try to fill as much as possible before expiring)
- FILL_OR_KILL FOK order (if the entire order cannot be executed immediately, it is canceled; if it cannot be fully filled, the order will expire)

Order Status

- NEW Unfilled
- FILLED Filled
- PARTIALLY_FILLED Partially filled
- CANCELED Canceled
- PARTIALLY_CANCELED Partially canceled

K-line Intervals

- 1m: 1 minute
- 5m: 5 minutes
- 15m: 15 minutes
- 30m: 30 minutes
- 60m: 1 hour
- 4h: 4 hours
- 1d: 1 day
- 1W: 1 week
- 1M: 1 month

Change Types

- WITHDRAW: Withdrawal
- WITHDRAW_FEE: Withdrawal fee
- DEPOSIT: Deposit

- DEPOSIT_FEE: Deposit fee
- ENTRUST: Order trade
- ENTRUST_PLACE: Order placement
- ENTRUST_CANCEL: Order cancellation
- TRADE_FEE: Trade fee
- ENTRUST_UNFROZEN: Order frozen funds return
- SUGAR: Airdrop
- ETF_INDEX: ETF order