CoinDCX Futures API Doc

Glossary

1. e - is the Event type
2. p - price (LTP)
3. q - quantity (trade quantity)
4. pr - product (futures)
5. f - futures
6. s - spot
7. T - timestamp
8. m - is maker. Boolean value that would be true if its maker and false if its taker
9. RT - range timestamp
10. ts - timestamp
11. vs - version
12. Ets - event timestamp as given by TPE (applicable to candlesticks data)
13. i - Interval
14. E - event timestamp (applicable to order book data)
15. pST - price sent time
16. v - volume 24h
17. ls - last price
18. pc - price change percent
19. btST - TPE Tick send time
20. mp - mark price
21. bmST - TPE mark price send time (The timestamp at which Third-Party exchange sent this event)

A few pointers about Futures implementation

- Any reference to instrument in Rest API/Websocket should be replaced with the actual instrument name. Instrument name is generally in the format B-BTC_USDT (Here, "B-" denotes the third party exchange. BTC denotes the underlying token of the futures contract. And USDT denotes the quote currency of the market)
- You can place Take Profit and Stop Loss only when an active position is present and not at the time of placing a limit order. Additionally please note that this is position TP SL. Position TPSL closes the entire position when the trigger price is reached.
- For every contract example BTCUSDT, there is a single position per user. Hence all orders would add/subtract the quantity of this position.
- Please implement a heartbeat check for every channel on the websockets. A sample code is given below , Otherwise your sockets will get disconnected automatically in some time

Get active instruments

Python

```python
import requests # Install requests module first.
url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/active_instrument
s"
response = requests.get(url)
data = response.json()
print(data)
```

JavaScript

```javascript
const request = require('request')
const url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/active_instrument
s"
request.get(url ,function(error, response, body) {
    console.log(body);
})
```

Sample Response:

Unset

```
[
  "B-MKR_USDT",
  "B-JTO_USDT",
  "B-1000SATS_USDT",
  "B-SFP_USDT",
  "B-POLYX_USDT",
  "B-BADGER_USDT",
  "B-CYBER_USDT",
  "B-MTL_USDT",
  "B-MDT_USDT",
  "B-BEAMX_USDT",
  "B-AUCTION_USDT",
  "B-ANKR_USDT",
 ]
```

Get instrument details

Python

```python
import requests # Install requests module first.
url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/instrument?pair={instrument}"
#sample_url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/instrument?pair=B-MKR_USDT"
response = requests.get(url)
data = response.json()
print(data)
```

JavaScript

```javascript
const request = require('request')
const url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/instrument?pair={instrument}"
//const sample_url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/instrument?pair=B-MKR_USDT"
request.get(url ,function(error, response, body) {
        console.log(body);})
```

Sample Response:

Unset


```
{
  "instrument": {
    "settle_currency_short_name": "USDT",
    "quote_currency_short_name": "USDT",
    "position_currency_short_name": "MATIC",
    "underlying_currency_short_name": "MATIC",
    "status": "inactive",
    "pair": "BM-MATIC_USDT",
    "kind": "perpetual",
    "settlement": "never",
```

```
    "max_leverage_long": 12,
    "max_leverage_short": 12,
    "unit_contract_value": 0.01,
    "price_increment": 0.0001,
    "quantity_increment": 1000,
    "min_trade_size": 1000,
    "min_price": 0.05,
    "max_price": 1000,
    "min_quantity": 1000,
    "max_quantity": 1000000000,
    "min_notional": null,
    "maker_fee": 0.01,
    "taker_fee": 0.075,
    "safety_percentage": 2.5,
    "quanto_to_settle_multiplier": 1,
    "is_inverse": false,
    "is_quanto": false,
    "allow_post_only": true,
    "allow_hidden": false,
    "max_market_order_quantity": null,
    "funding_frequency": null,
    "max_notional": 1000000,
    "expiry_time": 2524651260000,
    "time_in_force_options": [
      "good_till_cancel",
      "immediate_or_cancel",
      "fill_or_kill"
    ],
    "order_types": [
      "limit_order",
      "market_order",
      "stop_limit"
    ]
  }
}
```


Get instrument Real-time trade history

While rest APIs exist for this, we recommend using [Websockets]

Python

```python
import requests # Install requests module first.
url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/trades?pair={instrument_name}"
#sample_url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/trades?pair=B-MKR_USDT"
response = requests.get(url)
data = response.json()
print(data)
```

JavaScript

```javascript
const request = require('request')
const url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/trades?pair={instrument_name}"
//const sample_url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/data/trades?pair=B-MKR_USDT"


request.get(url ,function(error, response, body) {
        console.log(body);})
```

Sample Response:

Unset


```
[
  {
    "price": 1.1702,
    "quantity": 22000,
    "timestamp": 1675037938736,
    "is_maker": true
  },
  {
    "price": 1.1702,
    "quantity": 38000,
    "timestamp": 1675037950130,
    "is_maker": true
  }
  }]
```

Get instrument orderbook

While rest APIs exist for this, we recommend using Websockets
Here 50 denotes, the depth of the order book the other possible values are 10 and 20

Python

```python
import requests # Install requests module first.

url =
"https://public.coindcx.com/market_data/v3/orderbook/{instrument}-futures/50"
#sample_url =
"public.coindcx.com/market_data/v3/orderbook/B-MKR_USDT-futures/50"
#Here 50 denotes, the depth of the order book the other possible values are 10
and 20
response = requests.get(url)
data = response.json()
print(data)
```

JavaScript

```javascript
const request = require('request')
const url =
"https://public.coindcx.com/market_data/v3/orderbook/{instrument}-futures/50"
//const sample_url =
"public.coindcx.com/market_data/v3/orderbook/B-MKR_USDT-futures/50"
//Here 50 denotes, the depth of the order book the other possible values are 10
and 20
request.get(url ,function(error, response, body) {
        console.log(body);})
```

Sample Response:

Unset

```
{
  "ts": 1705483019891,
  "vs": 27570132,
```

```
  "asks": {
    "2001": "2.145",
    "2002": "4.453",
    "2003": "2.997"
  },
  "bids": {
    "1995": "2.618",
    "1996": "1.55"
  }
}
```

Get instrument candlesticks

While rest APIs exist for this, we recommend using Websockets

Unset

```
#Query parameters Explanation
{
"pair": "B-ID_USDT", # instrument.pair accepts string values
"from": "1707375997464", # EPOCH timestamp in seconds accepts integer values
"to": "1707375997464", # EPOCH timestamp in seconds accepts integer values
"resolution": "1", # '1' OR '5' OR '60' OR '1D' for 1min, 5min, 1hour, 1day
respectively
"pcode": "f" # Static
}
```

Python

```
import requests
url = "https://public.coindcx.com/market_data/candlesticks"
query_params = {
    "pair": "B-MKR_USDT",
    "from": 1704100940,
    "to": 1705483340,
    "resolution": "1D",  # '1' OR '5' OR '60' OR '1D'
    "pcode": "f"
}
response = requests.get(url, params=query_params)
```

```
if response.status_code == 200:
    data = response.json()
    # Process the data as needed
    print(data)
else:
    print(f"Error: {response.status_code}, {response.text}")
```

JavaScript

```
const request = require('request')
const url =
"https://public.coindcx.com/market_data/candlesticks?pair={pair}&from={from}&to
={to}&resolution={resolution}&pcode=f"
request.get(url ,function(error, response, body) {
        console.log(body);})
```

Sample Response:

Unset


```
{
  "s":"ok",
  "data":[
    {
      "open":1654.2,
      "high":1933.5,
      "low":1616.5,
      "volume":114433.544,
      "close":1831.9,
      "time":1704153600000
    },
    {
      "open":1832.2,
      "high":1961,
      "low":1438,
      "volume":158441.387,
      "close":1807.6,
      "time":1704240000000
    }
  ]
}
```

List Orders

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
             "timestamp": timeStamp , # EPOCH timestamp in seconds
             "status": "open", # Comma separated statuses as
open,filled,cancelled
             "side": "buy", # buy OR sell
             "page": "1", #// no.of pages needed
             "size": "10" #// no.of records needed
             }

json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(),
hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/orders"

headers = {
    'Content-Type': 'application/json',
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
}
```

```
response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
"timestamp": timeStamp , // EPOCH timestamp in seconds
"status": "open", // Comma separated statuses as open,filled,cancelled
"side": "buy", // buy OR sell
"page": "1", // no.of pages needed
"size": "10" // no.of records needed
}
const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')
const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/orders",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}
request.post(options, function(error, response, body) {
  console.log(body);
})
```

Unset


```
[
  {
    "id":"714d2080-1fe3-4c6e-ba81-9d2ac9a46003",
    "pair":"B-ETH_USDT",
    "side":"buy",
    "status":"open",
    "order_type":"limit_order",
    "stop_trigger_instruction":"last_price",
    "notification":"no_notification",
    "leverage":20.0,
    "maker_fee":0.025,
    "taker_fee":0.075,
    "fee_amount":0.0,
    "price":2037.69,
    "stop_price":0.0,
    "avg_price":0.0,
    "total_quantity":0.019,
    "remaining_quantity":0.019,
    "cancelled_quantity":0.0,
    "ideal_margin":1.93870920825,
    "order_category":"None",
    "stage":"default",
    "group_id":"None",
    "display_message":"ETH limit buy order placed!",
    "group_status":"None",
    "created_at":1705565256365,
    "updated_at":1705565256940
  },
  {
    "id":"ffb261ae-8010-4cec-b6e9-c111e0cc0c10",
    "pair":"B-ID_USDT",
    "side":"buy",
    "status":"filled",
    "order_type":"market_order",
    "stop_trigger_instruction":"last_price",
    "notification":"no_notification",
    "leverage":10.0,
    "maker_fee":0.025,
    "taker_fee":0.075,
    "fee_amount":0.011181375,
    "price":0.3312,
    "stop_price":0.0,
    "avg_price":0.3313,
```

```
   "total_quantity":45.0,
   "remaining_quantity":0.0,
   "cancelled_quantity":0.0,
   "ideal_margin":1.4926356,
   "order_category":"None",
   "stage":"default",
   "group_id":"None",
   "display_message":"ID market buy order filled!",
   "group_status":"None",
   "created_at":1705565061504,
   "updated_at":1705565062462
  }
]
```


Create Order

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp":timeStamp , # EPOCH timestamp in seconds
```

```
            "order": {
            "side": "sell", # buy OR sell
            "pair": "B-ID_USDT", # instrument.string
            "order_type": "market_order", # market_order OR limit_order
            "price": "0.2962", #numeric value
            "total_quantity": 33, #numerice value
            "leverage": 10, #numerice value
            "notification": "email_notification", # no_notification OR
email_notification OR push_notification
            "time_in_force": "good_till_cancel", # good_till_cancel OR
fill_or_kill OR immediate_or_cancel
            "hidden": False, # True or False
            "post_only": False # True or False
            }
            }

json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/orders/create"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);
```

```
// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
"timestamp": timeStamp , // EPOCH timestamp in seconds
"order": {
"side": "enum", // buy OR sell
"pair": "string", // instrument.string
"order_type": "enum", // market_order OR limit_order
"price": "numeric",
"total_quantity": "numeric",
"leverage": "integer",
"notification": "enum", // no_notification OR email_notification OR
push_notification
"time_in_force": "enum", // good_till_cancel OR fill_or_kill OR
immediate_or_cancel
"hidden": "boolean",
"post_only": "boolean"
}
}

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/orders/create",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

Unset


```
[
  {
    "id":"c87ca633-6218-44ea-900b-e86981358cbd",
    "pair":"B-ID_USDT",
    "side":"sell",
    "status":"initial",
    "order_type":"market_order",
    "notification":"email_notification",
    "leverage":10.0,
    "maker_fee":0.025,
    "taker_fee":0.075,
    "fee_amount":0.0,
    "price":0.2966,
    "avg_price":0.0,
    "total_quantity":33.0,
    "remaining_quantity":33.0,
    "cancelled_quantity":0.0,
    "ideal_margin":0.98024817,
    "order_category":"None",
    "stage":"default",
    "group_id":"None",
    "display_message":"None",
    "group_status":"None",
    "created_at":1705647376759,
    "updated_at":1705647376759
  }
]
```


Cancel Order

Python

```
import hmac
```

```python
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp":timeStamp , # EPOCH timestamp in seconds
              "id": "c87ca633-6218-44ea-900b-e86981358cbd" # order.id
              }


json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/orders/cancel"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
       "timestamp": timeStamp , // EPOCH timestamp in seconds
       "id": "string" // order.id
       }

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/orders/cancel",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response

Unset

```
{'message': 'success', 'status': 200, 'code': 200}
```

List Positions

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp":timeStamp , # EPOCH timestamp in seconds
              "page": "1", #no. of pages needed
              "size": "10" #no. of records needed
              }


json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/positions"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
              "timestamp": timeStamp , // EPOCH timestamp in seconds
              "page": "1", //no . of pages needed
              "size": "10" //no. of records needed
              }
const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

Unset

```
[
  {
    "id":"434dc174-6503-4509-8b2b-71b325fe417a",
```

```
    "pair":"B-ETH_USDT",
    "active_pos":0.0,
    "inactive_pos_buy":0.0,
    "inactive_pos_sell":0.0,
    "avg_price":0.0,
    "liquidation_price":0.0,
    "locked_margin":0.0,
    "locked_user_margin":0.0,
    "locked_order_margin":0.0,
    "take_profit_trigger":"None",
    "stop_loss_trigger":"None",
    "updated_at":1705644363791
  },
  {
    "id":"e65e8b77-fe7c-40c3-ada1-b1d4ea40465f",
    "pair":"B-ID_USDT",
    "active_pos":0.0,
    "inactive_pos_buy":0.0,
    "inactive_pos_sell":0.0,
    "avg_price":0.0,
    "liquidation_price":0.0,
    "locked_margin":0.0,
    "locked_user_margin":0.0,
    "locked_order_margin":0.0,
    "take_profit_trigger":0.0,
    "stop_loss_trigger":0.0,
    "updated_at":1705565210793
  }
]
```


Add Margin

Python

```python
import hmac
import hashlib
import base64
import json
import time
```

```
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp": timeStamp , // EPOCH timestamp in seconds
              "id": "434dc174-6503-4509-8b2b-71b325fe417a", // position.id
              "amount":1
              }


json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/add_margin"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}
response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"
```

```
const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
       "timestamp": timeStamp , // EPOCH timestamp in seconds
       "id": "434dc174-6503-4509-8b2b-71b325fe417a", // position.id
       "amount": 1
       }

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions/add_margin",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

```
Unset
{'message': 'success', 'status': 200, 'code': 200}
```

Remove Margin

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp": timeStamp , # EPOCH timestamp in seconds
              "id": "434dc174-6503-4509-8b2b-71b325fe417a", # position.id
              "amount": 10
              }



json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/remove_margi
n"

headers = {
  'Content-Type': 'application/json',
```

```
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"
const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
              "timestamp": timeStamp , // EPOCH timestamp in seconds
              "id": "434dc174-6503-4509-8b2b-71b325fe417a", // position.id
              "amount": 10
              }
const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')
const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions/remove_margin",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

```
Unset
{'message': 'success', 'status': 200, 'code': 200}
```

Cancel All Open Orders

```
Python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
"timestamp": timeStamp  // EPOCH timestamp in seconds
}



json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()
```

```
url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/cancel_all_open_orders"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json();
print(data);
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
                    "timestamp": timeStamp // EPOCH timestamp in seconds
                    }



const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl +
"/exchange/v1/derivatives/futures/positions/cancel_all_open_orders",
```

```
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

```
Unset
{'message': 'success', 'status': 200, 'code': 200}
```

Cancel All Open Orders for Position

```
Python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
```

```
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
       "timestamp":timeStamp , # EPOCH timestamp in seconds
       "id": "434dc174-6503-4509-8b2b-71b325fe417a" # position.id
       }



json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/cancel_all_o
pen_orders_for_position"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
```

```
const secret = "";


const body = {
              "timestamp":timeStamp , // EPOCH timestamp in seconds
              "id": "434dc174-6503-4509-8b2b-71b325fe417a" // position.id
              }



const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl +
"/exchange/v1/derivatives/futures/positions/cancel_all_open_orders_for_position
",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

```
Unset
{'message': 'success', 'status': 200, 'code': 200}
```

Exit Position

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
       "timestamp": timeStamp , # EPOCH timestamp in seconds
       "id": "434dc174-6503-4509-8b2b-71b325fe417a" # position.id
       }



json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/positions/exit"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
              "timestamp": timeStamp , // EPOCH timestamp in seconds
              "id": "434dc174-6503-4509-8b2b-71b325fe417a" // position.id
              }


const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions/exit",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

Unset

```
{'message': 'success', 'status': 200, 'code': 200, 'data': {'group_id':
'baf926e6B-ID_USDT1705647709'}}
```

Create Take Profit and Stop Loss Orders

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
              "timestamp": "1707375997464", # EPOCH timestamp in seconds
              "id": "e65e8b77-fe7c-40c3-ada1-b1d4ea40465f", # position.id
              "take_profit": {
              "stop_price": "1",
              "limit_price": "0.9", # required for take_profit_limit orders
              "order_type": "take_profit_limit" # take_profit_limit OR
take_profit_market
              },
              "stop_loss": {
              "stop_price": "0.271",
              "limit_price": "0.270", # required for stop_limit orders
              "order_type": "stop_limit" # stop_limit OR stop_market
              }
              }
```

```
json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/create_tpsl"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
              "timestamp": "integer", // EPOCH timestamp in seconds
              "id": "e65e8b77-fe7c-40c3-ada1-b1d4ea40465f", // position.id
              "take_profit": {
              "stop_price": "1",
              "limit_price": "0.9", // required for take_profit_limit orders
              "order_type": "take_profit_limit" // take_profit_limit OR
take_profit_market
              },
```

```
                "stop_loss": {
                "stop_price": "0.271",
                "limit_price": "0.270", // required for stop_limit orders
                "order_type": "stop_limit" // stop_limit OR stop_market
                }
                }

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions/create_tpsl",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

Unset


```
{
 "stop_loss":{
  "id":"8f8ee959-36cb-4932-bf3c-5c4294f21fec",
  "pair":"B-ID_USDT",
  "side":"sell",
  "status":"untriggered",
  "order_type":"stop_limit",
  "stop_trigger_instruction":"last_price",
  "notification":"email_notification",
  "leverage":1.0,
  "maker_fee":0.025,
  "taker_fee":0.075,
  "fee_amount":0.0,
  "price":0.27,
```

```
    "stop_price":0.271,
    "avg_price":0.0,
    "total_quantity":0.0,
    "remaining_quantity":0.0,
    "cancelled_quantity":0.0,
    "ideal_margin":0.0,
    "order_category":"complete_tpsl",
    "stage":"tpsl_exit",
    "group_id":"None",
    "display_message":"None",
    "group_status":"None",
    "created_at":1705915027938,
    "updated_at":1705915028003
  },
  "take_profit":{
    "success":false,
    "error":"TP already exists"
  }
}
```


Get Transactions

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
```

```
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
       "timestamp": timeStamp, # EPOCH timestamp in seconds
       "position_ids": "e65e8b77-fe7c-40c3-ada1-b1d4ea40465f", # Comma
separated position.id
       "stage": "all", # all OR default OR funding
       "page": "1", #no. of pages needed
       "size": "10" #no. of records needed
       }
json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url =
"https://api.coindcx.com/exchange/v1/derivatives/futures/positions/transactions
"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
```

```
const key = "";
const secret = "";


const body = {
              "timestamp": timeStamp, // EPOCH timestamp in seconds
              "position_ids": "e65e8b77-fe7c-40c3-ada1-b1d4ea40465f", //
Comma separated position.id
              "stage": "all", // all OR default OR funding
              "page": "1", //no. of pages needed
              "size": "10" //no. of records needed
              }

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/positions/transactions",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

```
Unset
[
  {
    "pair":"B-ID_USDT",
    "stage":"default",
    "amount":0.0495,
    "fee_amount":0.007368075,
    "price_in_inr":88.21,
    "price_in_btc":2.4085979238367e-05,
```

```
   "price_in_usdt":1.0,
   "source":"user",
   "parent_type":"Derivatives::Futures::Order",
   "parent_id":"34527a16-8238-4aca-a3b4-08950dc9ac90",
   "position_id":"e65e8b77-fe7c-40c3-ada1-b1d4ea40465f",
   "created_at":1705647710699,
   "updated_at":1705647710734
  }
]
```


Get Trades

Python

```python
import hmac
import hashlib
import base64
import json
import time
import requests

# Enter your API Key and Secret here. If you don't have one, you can generate
it from the website.
key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

# Generating a timestamp
timeStamp = int(round(time.time() * 1000))

body = {
       "timestamp":timeStamp, # EPOCH timestamp in seconds
```

```
        "pair": "B-ID_USDT", # instrument.pair
        "order_id": "9b37c924-d8cf-4a0b-8475-cc8a2b14b962", # order.id
        "from_date": "2024-01-01", # format YYYY-MM-DD
        "to_date": "2024-01-22", # format YYYY-MM-DD
        "page": "1", #no. of pages needed
        "size": "10" #no. of records needed
        }
json_body = json.dumps(body, separators = (',', ':'))

signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

url = "https://api.coindcx.com/exchange/v1/derivatives/futures/trades"

headers = {
  'Content-Type': 'application/json',
  'X-AUTH-APIKEY': key,
  'X-AUTH-SIGNATURE': signature
}

response = requests.post(url, data = json_body, headers = headers)
data = response.json()
print(data)
```

JavaScript

```
const request = require('request')
const crypto = require('crypto')

const baseurl = "https://api.coindcx.com"

const timeStamp = Math.floor(Date.now());
// To check if the timestamp is correct
console.log(timeStamp);

// Place your API key and secret below. You can generate it from the website.
const key = "";
const secret = "";


const body = {
              "timestamp": timeStamp, // EPOCH timestamp in seconds
              "pair": "B-ID_USDT", // instrument.pair
              "order_id": "9b37c924-d8cf-4a0b-8475-cc8a2b14b962", // order.id
```

```
                "from_date": "2024-01-01", // format YYYY-MM-DD
                "to_date": "2024-01-22", // format YYYY-MM-DD
                "page": "1", //no. of pages needed
                "size": "10" //no of records needed
                }

const payload = new Buffer(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

const options = {
  url: baseurl + "/exchange/v1/derivatives/futures/trades",
  headers: {
    'X-AUTH-APIKEY': key,
    'X-AUTH-SIGNATURE': signature
  },
  json: true,
  body: body
}

request.post(options, function(error, response, body) {
  console.log(body);
})
```

Sample Response:

Unset

```
[
  {
     "price":0.2962,
     "quantity":33.0,
     "is_maker":false,
     "fee_amount":0.00733095,
     "pair":"B-ID_USDT",
     "side":"buy",
     "timestamp":1705645534425.8374,
     "order_id":"9b37c924-d8cf-4a0b-8475-cc8a2b14b962"
  }
]
```

WEB SOCKETS

CHANNELS

coindcx

This is an authenticated channel that gives data about Futures Position updates, Order updates and Wallet Balance updates.
Channel Name: coindcx

Python

```python
import socketio
import hmac
import hashlib
import json
socketEndpoint = 'wss://stream.coindcx.com'
sio = socketio.Client()

sio.connect(socketEndpoint, transports = 'websocket')

key = "XXXX"
secret = "YYYY"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
# python2
secret_bytes = bytes(secret)

body = {"channel":"coindcx"}
json_body = json.dumps(body, separators = (',', ':'))
signature = hmac.new(secret_bytes, json_body.encode(), hashlib.sha256).hexdigest()

# Join channel
sio.emit('join', { 'channelName': 'coindcx', 'authSignature': signature, 'apiKey' :
key })

### Listen update on eventName
### Replace the <eventName> with the df-position-update, df-order-update,
###balance-update

@sio.on(<eventName>)
def on_message(response):
```

```
  print(response["data"])

# leave a channel
sio.emit('leave', { 'channelName' : 'coindcx' })
```

JavaScript

```
//For commonJS(NPM)
const io = require("socket.io-client");
const crypto = require('crypto');

/// ES6 import or TypeScript
import io from 'socket.io-client';
import crypto from 'crypto';


const socketEndpoint = "wss://stream.coindcx.com";

//connect to server.
const socket = io(socketEndpoint, {
 transports: ['websocket']
});

const secret = "secret";
const key = "key";


const body = { channel: "coindcx" };
const payload = Buffer.from(JSON.stringify(body)).toString();
const signature = crypto.createHmac('sha256',
secret).update(payload).digest('hex')

socket.on("connect", () => {
 //Join channel
 socket.emit('join', {
  'channelName': "coindcx",
  'authSignature': signature,
  'apiKey' : key
 });
});
```

```
//Listen update on eventName
//Replace the <eventName> with the df-position-update, df-order-update,
//balance-update
socket.on(<eventName>, (response) => {
 console.log(response.data);
});


// In order to leave a channel
socket.emit('leave', {
 'channelName': 'coindcx'
});

// NOTE : Need to use V2 Socket.io-client
```

EVENTS:

df-position-update

Python

```
@sio.on('df-position-update')
def on_message(response):
  print(response["data"])
```

JavaScript

```
socket.on("df-position-update", (response) => {
  console.log(response.data);
});
```

Sample Response:

Unset

```
[
  {
    "id":"e65e8b77-fe7c-40c3-ada1-b1d4ea40465f",
    "pair":"B-ID_USDT",
```

```
  "active_pos":35,
  "inactive_pos_buy":0,
  "inactive_pos_sell":0,
  "avg_price":0.2839,
  "liquidation_price":0.261452618175,
  "locked_margin":0.984388363875,
  "locked_user_margin":0.99478995,
  "locked_order_margin":0,
  "take_profit_trigger":0,
  "stop_loss_trigger":0.271,
  "updated_at":1705999727738
 }
]
```


df-order-update

Python

```
@sio.on('df-order-update')
def on_message(response):
  print(response["data"])
```

JavaScript

```
socket.on("df-order-update", (response) => {
  console.log(response.data);
});
```

Sample Response:

Unset


```
[
 {
  "id":"ff5a645f-84b7-4d63-b513-9e2f960855fc",
  "pair":"B-ID_USDT",
  "side":"sell",
  "status":"cancelled",
  "order_type":"take_profit_limit",
  "stop_trigger_instruction":"last_price",
```

```
   "notification":"email_notification",
   "leverage":1,
   "maker_fee":0.025,
   "taker_fee":0.075,
   "fee_amount":0,
   "price":0.9,
   "stop_price":1,
   "avg_price":0,
   "total_quantity":0,
   "remaining_quantity":0,
   "cancelled_quantity":0,
   "ideal_margin":0,
   "order_category":"complete_tpsl",
   "stage":"tpsl_exit",
   "created_at":1705915012812,
   "updated_at":1705999727686,
   "trades":[

   ],
   "display_message":null,
   "group_status":null,
   "group_id":null
 }
]
```


balance-update

Python

```python
@sio.on('df-order-update')
def on_message(response):
  print(response["data"])
```

JavaScript

```javascript
socket.on("df-order-update", (response) => {
  console.log(response.data);
});
```

Sample Response:

```
Unset
[
  {
    "id":"026ef0f2-b5d8-11ee-b182-570ad79469a2",
    "balance":"1.0221449",
    "locked_balance":"0.99478995",
    "currency_id":"c19c38d1-3ebb-47ab-9207-62d043be7447",
    "currency_short_name":"USDT"
  }
]
```

Candlesticks

Channel names : <instrument_name>_1m-futures , <instrument_name>_1h-futures, <instrument_name>_1d-futures

```
Python
["join",{"channelName": "B-BTC_USDT_1m-futures" }]
```

EVENTS

candlestick

```
Python

@sio.on('candlestick')
def on_message(response):
  print(response["data"])
```

```
JavaScript
socket.on("candlestick", (response) => {
  console.log(response.data);
});
```

Sample Response:

```
Unset
{
 "data":[
  {
   "open":"0.3524000",
   "close":"0.3472000",
   "high":"0.3531000",
   "low":"0.3466000",
   "volume":"5020395",
   "open_time":1705514400,
   "close_time":1705517999.999,
   "pair":"B-ID_USDT",
   "duration":"1h",
   "symbol":"IDUSDT",
   "quote_volume":"1753315.2309000"
  }
 ],
 "Ets":1705516366626,
 "i":"1h",
 "channel":"B-ID_USDT_1h-futures",
 "pr":"futures"
}
```

Orderbook

Channel name: <instrument_name>@orderbook@50-futures

#Here 50 denotes, the depth of the order book the other possible values are 10 and 20

<instrument_name> can be derived from Get active instruments

Example for Instrument Name :

```
Unset
Example of an instrument_name : B-ID_USDT
```

```
Python
sio.emit('join', {'channelName':"B-ID_USDT@
orderbook@50-futures"})
```

JavaScript

```
socket.on("connect", () => {
  //Join channel
  socket.emit('join', {
    'channelName': "B-ID_USDT@orderbook@50-futures"
  });
});
```

EVENTS:

depth-snapshot (order-book)

Python

```
@sio.on('depth-snapshot')
def on_message(response):
  print(response["data"])
```

JavaScript

```
socket.on("depth-snapshot", (response) => {
  console.log(response.data);
});
```

JavaScript


```
{
 "ts":1705913767265,
 "vs":53727235,
 "asks":{
  "2410":"112.442",
  "2409.77":"55.997",
  "2409.78":"5.912"
 },
 "bids":{
  "2409.76":"12.417",
  "2409.75":"1.516",
  "2409.74":"15.876"
 },
 "pr":"futures"
```

```
}
```

depth-update

Python

```python
@sio.on('depth-update')
def on_message(response):
  print(response["data"])
```

JavaScript

```javascript
socket.on("depth-update", (response) => {
  console.log(response.data);
});
```

Sample response:

Unset


```
{
 "ts":1705516361672,
 "vs":40167,
 "asks":{
  "0.3473":"8856",
  "0.3474":"11224",
  "0.3476":"17085"
 },
 "bids":{
  "0.3472":"3369",
  "0.3471":"15412",
  "0.347":"8234"
 },
 "E":1705516361573,
 "pr":"futures"
}
```

currentPrices@futures@rt

Python

```
sio.emit('join', {'channelName':"currentPrices@futures@rt"})
```

JavaScript

```
socket.on("connect", () => {
  socket.emit('join', {
    'channelName': "currentPrices@futures@rt"
  });
});
```

EVENTS:

currentPrices@futures#update

Python

```
@sio.on('currentPrices@futures#update')
def on_message(response):
  print(response["data"])
```

JavaScript

```
socket.on("currentPrices@futures#update", (response) => {
  console.log(response.data);
});
```

Sample Response:

Unset

```
{
   "vs":29358821,
```

```
   "ts":1707384027242,
   "pr":"futures",
   "pST":1707384027230,
   "prices":{
      "B-UNI_USDT":{
         "bmST":1707384027000,
         "cmRT":1707384027149
      },
      "B-LDO_USDT":{
         "mp":2.87559482,
         "bmST":1707384027000,
         "cmRT":1707384027149
      }
   }
}
```


Futures Trades

Channel name: <instrument_name>@trades-futures

Python

```
sio.emit('join', {'channelName':"B-ID_USDT@trades-futures"})
```

JavaScript

```
socket.on("connect", () => {
  //Join channel
  socket.emit('join', {
    'channelName': "B-ID_USDT@trades-futures"
  });
});
```

EVENTS:

new-trade

Python

```python
@sio.on('new-trade')
def on_message(response):
  print(response["data"])
```

JavaScript

```javascript
socket.on("new-trade", (response) => {
  console.log(response.data);
});
```

Sample response:

Unset

```
{
  "T":1705516361108,
  "RT":1705516416271.6133,
  "p":"0.3473",
  "q":"40",
  "m":1,
  "s":"B-ID_USDT",
  "pr":"f"
}
```

LTP

Channel name: <instrument_name>@prices-futures

Python

```python
sio.emit('join', {'channelName':"B-ID_USDT@prices-futures"})
```

JavaScript

```javascript
socket.on("connect", () => {
  //Join channel
  socket.emit('join', {
    'channelName': "B-ID_USDT@prices-futures"
  });
```

```
});
```

EVENTS:

Price-change (LTP)

Python

```python
@sio.on('price-change')
def on_message(response):
  print(response["data"])
```

JavaScript

```javascript
socket.on("price-change", (response) => {
  console.log(response.data);
});
```

Sample response:

Unset

```
{
  "T":1705516361108,
  "p":"0.3473",
  "pr":"f"
}
```

Sample Code:

Websocket connection implementation with ping check:

Python

```python
import socketio
```

```
import hmac
import hashlib
import json
import time
import asyncio
from datetime import datetime
from socketio.exceptions import TimeoutError
socketEndpoint = 'wss://stream.coindcx.com'
sio = socketio.AsyncClient()

key = "xxx"
secret = "xxx"

# python3
secret_bytes = bytes(secret, encoding='utf-8')
channelName = "coindcx"
body = {"channel": channelName}
json_body = json.dumps(body, separators=(',', ':'))
signature = hmac.new(secret_bytes, json_body.encode(),
hashlib.sha256).hexdigest()


async def ping_task():
    while True:
        await asyncio.sleep(25)
        try:
            await sio.emit('ping', {'data': 'Ping message'})
        except Exception as e:
            print(f"Error sending ping: {e}")


@sio.event
async def connect():
    print("I'm connected!")
    current_time = datetime.now()
       print("Connected Time:", current_time.strftime("%Y-%m-%d %H:%M:%S"))

    await sio.emit('join', {'channelName': "coindcx", 'authSignature':
signature, 'apiKey': key})
    await sio.emit('join', {'channelName': "B-ID_USDT@prices-futures"})


@sio.on('price-change')
async def on_message(response):
```

```
    current_time = datetime.now()
       print("Price Change Time:", current_time.strftime("%Y-%m-%d %H:%M:%S"))
    print("Price Change Response !!!")
    print(response)


async def main():
    try:
        await sio.connect(socketEndpoint, transports='websocket')
        # Wait for the connection to be established
        asyncio.create_task(ping_task())

        await sio.wait()
        while True:
            time.sleep(1)
            sio.event('price-change', {'channelName':
"B-ID_USDT@prices-futures"})
    except Exception as e:
        print(f"Error connecting to the server: {e}")
        raise  # re-raise the exception to see the full traceback


# Run the main function
if __name__ == '__main__':
    asyncio.run(main())
```

Support:

Please email us at futuresapi@coindcx.ocm if you have any questions.