Introduction

The Breeze API is a powerful API through which you can fetch live/historical data, automate your trading strategies, and monitor your portfolio in real time.

An AppKey and secret_key pair is issued that allows you to connect to the server. You have to register a redirect url where you will be sent after the login.

Breeze API is designed to accept 100 API calls per minute and 5000 API calls per day.

All inputs are form-encoded parameters and responses are in the form of JSON. The success and the error states are indicated by standard HTTP codes and are accompanied by JSON data.

You can also use Breeze API alongside with the third-party applications. Our documentation is a one stop reference to all your questions. You can also reach out to us through the iCommunity.

Asset Class	NSE	MCX
Equity	Yes	NA
Equity Futures	Yes	NA
Equity Options	Yes	NA
Currency Futures	Coming Soon	NA
Currency Options	Coming Soon	NA
Commodity Futures	No	Coming So
Commodity Options	No	Coming So

Currently, securities listed on BSE, MSEI and NCDEX are not available on Breeze API.

How To Navigate

How to Navigate?

The documentation specifies various API endpoints which can be used to perform many functions as stated above.

In this documentation, you will find about the following:

- The computation of the checksum
- Sending the common request headers
- API Parameters, its data type and description
- Language specific guides
- Sample JSON Response

Checksum Computation & Login

```
import json
import hashlib
from datetime import datetime

# App related Secret Key
secret_key = 'your_SECRET_goes_here'

# 'body' is the request-body of your current request
payload = json.dumps(body, separators=(',', ':'))

#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
```

REGISTRATION

Authentication is done following the OAuth2.0 protocol. It ensures that all the fields in the request are safe from tampering.

Visit the Breeze API portal to register an app by providing the following information:

- App Name
- Redirect URL
- Breeze User ID

After the app is successfully registered, you should create the AppKey and secret_key pair. These keys will be unique to the app registered. The AppKey is the identity of the app for the API system and secret_key is used to encrypt messages sent to the API system by the client system.

All requests must contain:

- App Key (AppKey)
- Checksum
- Timestamp

CHECKSUM COMPUTATION

- Checksum is computed via SHA256 hash (Time Stamp + JSON Post Data + secret_key)
- Redirect response on login request is form post
- All JSON Data should be stringified before sending it to the I-Direct system.

LOGIN

Navigate to the Breeze API Login with your URL-encoded AppKey to initiate the login flow. https://api.icicidirect.com/apiuser/login?api_key=AppKey

Upon successful login you will have an API_Session in response payload. Use this API_Session value against key named SessionToken in the CustomerDetails API to obtain a session_token value, which is then used as SessionToken for signing all subsequent requests.

Request Headers

Key	Value	Description
X-Checksum		token followed by combination of ((ISO8601 UTC DateTime Format with 0 milliseconds) + JSONPostData + secret_key)

Key	Value	Description
X-Timestamp	2022-06-01T10:23:56.000Z	ISO8601 UTC DateTime Format with 0 milliseconds Request will be honoured only if server time stamp and client timestamp is between 60 seconds
X-AppKey		AppKey which is received during registration
X-SessionToken		SessionToken value is received in CustomerDetails API

The Customer Details and Customer Login API endpoints do not require headers

Errors

Exceptions and Errors

The API server generates name of the exception for error responses. While writing your code, you can raise corresponding exceptions after defining them in your language/library. The table below shows the list of exceptions and common error codes.

Instruments

You can download the Security Master file for token mapping at your end here. It is generated/updated daily at 8:00 AM.

Kindly note that the rate limit is set to 100 API calls per minute and 5000 API calls per day.

CustomerDetails

CustomerDetails

```
import http.client

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = "{\r\n    \"SessionToken\": \"58593\",\r\n    \"AppKey\": \"8g791^N029R47I831B8153=^O2f#7u8g\"\r\n}"
headers = {
           "Content-Type": "application/json"
       }
conn.request("GET", "/breezeapi/api/v1/customerdetails", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
   "Success": {
      "exg_trade_date": {
          "NSE": "01-Jun-2022",
          "BSE": "01-Jun-2022",
          "FNO": "31-May-2022",
          "NDX": "31-May-2022"
      },
      "exg_status": {
          "NSE": "O",
          "BSE": "O",
          "FNO": "O",
          "NDX": "X"
      },
      "segments_allowed": {
          "Trading": "Y",
          "Equity": "Y",
          "Derivatives": "Y",
          "Currency": "Y"
      },
      "idirect_userid": "hulk",
      "session_token": "aHVsazoxMjQzMjQ5",
      "idirect_user_name": "RUPENDRA DHONDIRAM CHAVAN",
      "idirect_ORD_TYP": "",
      "idirect_lastlogin_time": "01-Jun-2022 10:20:16",
      "mf_holding_mode_popup_flg": "N",
```

```
        "commodity_exchange_status": "O",
        "commodity_trade_date": "01-Jun-2022",
        "commodity_allowed": "O"
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/customerdetails

REQUEST BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
SessionToken	String	SessionToken is received in CustomerLogin API as API_Session	Yes
AppKey	String	AppKey is received during registration	Yes

DematHoldings

GetDematHoldings

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({})

#checksum computation
#time_stamp & checksum generation for request-headers

appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "your SECRET_KEY goes here"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token ' + checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/dematholdings", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "stock_code": "ICINIF",
            "stock_ISIN": "INF109K012R6",
            "quantity": "16",
            "demat_total_bulk_quantity": "16",
            "demat_avail_quantity": "0",
            "blocked_quantity": "0",
            "demat_allocated_quantity": "0"
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/dematholdings

BODY PARAMETERS

Parameter	Data Type	Description
--	--	--

Funds

GetFunds

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({})

#checksum computation
#time_stamp & checksum generation for request-headers
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token ' + checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("GET", "/breezeapi/api/v1/funds", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "bank_account": "000401012540",
        "total_bank_balance": 1143033633.04,
        "allocated_equity": 4037129.08,
        "allocated_fno": 1138996503.96,
        "block_by_trade_equity": 0,
        "block_by_trade_fno": 403992.28,
        "block_by_trade_balance": 403992.28,
        "unallocated_balance": "25215061.88"
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/funds

BODY PARAMETERS

Parameter	Data Type	Description
--	--	--

SetFunds

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "transaction_type": "Credit",
    "amount": "10000",
    "segment": "FNO"
})
secret_key = "your SECRET_KEY goes here"

#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
```

```
headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': 'insert your appkey here',
    'X-SessionToken': 'you will get this from customer detail API call'
}
conn.request("POST", "/breezeapi/api/v1/funds", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "status": "Success"
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/funds

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
transaction_type	String	"Debit", "Credit"	Yes
amount	String	Numeric string of Currency	Yes
segment	String	"Equity", "FNO"	Yes

HistoricalCharts

GetHistoricalChartsList

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "interval": "day",
    "from_date": "2022-05-02T07:00:00.000Z",
    "to_date": "2022-05-03T07:00:00.000Z",
    "stock_code": "ITC",
    "exchange_code": "NSE",
    "product_type": "Cash"
})


appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"
#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/historicalcharts", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "datetime": "2022-05-02 12:05:00",
            "stock_code": "ITC",
```

```
            "exchange_code": "NSE",
            "product_type": null,
            "expiry_date": null,
            "right": null,
            "strike_price": null,
            "open": "258",
            "high": "264.5",
            "low": "257.05",
            "close": "263.15",
            "volume": "18965820",
            "open_interest": null,
            "count": 0
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/historicalcharts

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
interval	String	"minute","5minute","30minute","day"	Yes
from_date	String	ISO 8601	Yes
to_date	String	ISO 8601	Yes
stock_code	String	"AXIBAN", "TATMOT"	Yes
exchange_code	String	"NSE", "NFO", "BSE"	Yes
product_type	String	"futures","options","furtureplus","futureplus_sltp","optionplus", "cash","eatm","btst","margin","marginplus"	No
expiry_date	String	ISO 8601	No
right	String	"call","put","others"	No
strike_price	String	Numeric String of Currency	No

Margin Calculator

margin calculator

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "list_of_positions": [
        {
            "strike_price": "0",
            "quantity": "15",
            "right": "others",
            "product": "futures",
            "action": "buy",
            "price": "46230.85",
            "expiry_date": "31-Aug-2023",
            "stock_code": "CNXBAN",
            "cover_order_flow": "N",
            "fresh_order_type": "N",
            "cover_limit_rate": "0",
            "cover_sltp_price": "0",
            "fresh_limit_rate": "0",
            "open_quantity": "0"
        },
        {
            "strike_price": "37000",
            "quantity": "15",
            "right": "Call",
            "product": "options",
            "action": "buy",
            "price": "9100",
            "expiry_date": "27-Jul-2023",
            "stock_code": "CNXBAN",
            "cover_order_flow": "N",
            "fresh_order_type": "N",
            "cover_limit_rate": "0",
```

```
            "cover_sltp_price": "0",
            "fresh_limit_rate": "0",
            "open_quantity": "0"
        },
        {
            "strike_price": "0",
            "quantity": "50",
            "right": "others",
            "product": "futureplus",
            "action": "buy",
            "price": "19800",
            "expiry_date": "27-Jul-2023",
            "stock_code": "NIFTY",
            "cover_order_flow": "N",
            "fresh_order_type": "N",
            "cover_limit_rate": "0",
            "cover_sltp_price": "0",
            "fresh_limit_rate": "0",
            "open_quantity": "0"
        },
        {
            "strike_price": "19600",
            "quantity": "50",
            "right": "call",
            "product": "optionplus",
            "action": "buy",
            "price": "245.05",
            "expiry_date": "27-Jul-2023",
            "stock_code": "NIFTY",
            "cover_order_flow": "sell",
            "fresh_order_type": "limit",
            "cover_limit_rate": "180.00",
            "cover_sltp_price": "200.00",
            "fresh_limit_rate": "245.05",
            "open_quantity": "50"
        }
    ],
    "exchange_code": "NFO"
})

#checksum computation

#time_stamp & checksum generation for request-headers
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token ' + checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': enter your base64sessiontoken
}
conn.request("POST", "/breezeapi/api/v1/margincalculator", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response

```
{'Success': {'margin_calulation': [{'strike_price': '19500',
    'quantity': '50',
    'right': 'Call',
    'product': 'Options',
    'action': 'Buy',
    'price': '250',
    'expiry_date': '26-Oct-2023',
    'stock_code': 'NIFTY '},
   {'strike_price': '19500',
    'quantity': '50',
    'right': 'Put',
    'product': 'Options',
    'action': 'Buy',
    'price': '200',
    'expiry_date': '26-Oct-2023',
    'stock_code': 'NIFTY '}],
  'non_span_margin_required': '0',
  'order_value': '22500',
  'order_margin': '0',
  'trade_margin': None,
  'block_trade_margin': '0',
  'span_margin_required': '22500'},
 'Status': 200,
 'Error': None}
```

REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/margincalculator

BODY PARAMETERS

Parameter	Data Type	Description
strike_price	String	Numeric String of Currency
Quantity	Integer	Number of quantity to place the order
right	String	"call","put","others"
product	String	"options","futures"
action	String	"buy","sell"
price	Double	Numeric Currency
expiry_date	String	ISO 8601
stock_code	String	"NIFTY","CNXBAN"

Margin

AddMargins

```
import http.client
import json
secret_key = "9e12o8179&J695!141J150147277V7"

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "BSE",
    "product_type": "EATM",
    "stock_code": "ACC",
    "cover_quantity": "1",
    "category_index_per_stock": "Stock",
    "margin_amount": "54590456",
    "expiry_date": "",
    "right": "",
    "strike_price": "",
    "settlement_id": "2021107",
    "add_amount": "100",
    "open_quantity": "2"
})

#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
var appkey = "9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+ checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("POST", "/breezeapi/api/v1/margin", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response

```
{
    "Success": null,
    "Status": 500,
    "Error": "Cannot place orders when exchange is in Expiry.:"
}
```

REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/margin

BODY PARAMETERS

Parameter	Data Type	Description
exchange_code	String	"NSE", "NFO", "BSE"

Parameter	Data Type	Description
product_type	String	"futures","options","furtureplus","futureplus_sltp","optionplus", "cash","eatm","btst","margin","marginplus"
stock_code	String	"AXIBAN", "TATMOT"
settlement_id	String	Numeric String of order's settlement-id to add margin
add_amount	String	Numeric String of Currency
margin_amount	String	Numeric String of Currency
open_quantity	String	Numeric String for Number of Open Order Quantity
cover_quantity	String	Numeric String for Number of Cover Order Quantity
category_index_per_stock	String	Index & Stock Value
expiry_date	String	ISO 8601
right	String	"call","put","others"
strike_price	Integer	Numeric Currency

GetMargins

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "NSE"
})
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"

#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token ' + checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken':  Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/margin", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "limit_list": [
            {
                "trade_date": "27-May-2022",
                "amount": 678618.3,
                "exchange_code": "NFO",
                "payin_date": "-",
                "payout_date": "30-May-2022"
            }
        ],
        "cash_limit": 305009999.9699,
        "amount_allocated": 5010000,
        "block_by_trade": 0,
        "isec_margin": 199999999.98
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/margin

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
exchange_code	String	"NSE", "NFO", "BSE"	Yes

Order

OrderPlacement

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "stock_code": "ITC",
    "exchange_code": "NSE",
    "product": "cash",
    "action": "buy",
    "order_type": "market",
    "quantity": "1",
    "price": "263.15",
    "validity": "ioc"
})

#checksum computation
#time_stamp & checksum generation for request-headers
app_key = "9e12o8179&J695!`141J150147277(V7"

secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("POST", "/breezeapi/api/v1/order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response

```
{
    "Success": {
        "order_id": "Equity CASH Order placed successfully through RI reference no 20220601N100000019",
        "message": null
    },
    "Status": 200,
    "Error": null
}
```

REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
stock_code	String	"AXIBAN", "TATMOT"	Yes
exchange_code	String	"NSE", "NFO", "BSE"	Yes
product	String	"futures","options","furtureplus","futureplus_sltp","optionplus", "cash","eatm","btst","margin","marginplus"	Yes
action	String	"buy", "sell"	Yes
order_type	String	"limit","market","stoploss"	Yes
stoploss	Double	Numeric Currency	No
quantity	Integer	Number of quantity to place the order	Yes
price	Double	Numeric Currency	Yes

Parameter	Data Type	Description	Mandatory
validity	String	"day","ioc","vtc"	Yes
validity_date	String	ISO 8601	No
disclosed_quantity	Integer	Number of quantity to be disclosed	No
expiry_date	String	ISO 8601	No
right	String	"call","put","others"	No
strike_price	Integer	Numeric Currency	No
user_remark	String	Users are free to add their comment/tag to the order	No

GetOrderDetail

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "NSE",
    "order_id": "20220601N100000019"
})

#checksum computation
#time_stamp & checksum generation for request-headers

secret_key = "9e12o8179&J695!`141J150147277(V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
app_key = "9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "order_id": "20220601N100000019",
            "exchange_order_id": null,
            "exchange_code": "NSE",
            "stock_code": "ITC",
            "product_type": "Cash",
            "action": "Buy",
            "order_type": "Limit",
            "stoploss": "0.00",
            "quantity": "1",
            "price": "263.15",
            "validity": "",
            "disclosed_quantity": "0",
            "expiry_date": null,
            "right": null,
            "strike_price": 0,
            "average_price": "0",
            "cancelled_quantity": "0",
            "pending_quantity": "1",
            "status": "Requested",
            "user_remark": "",
            "order_datetime": "01-Jun-2022 10:48",
            "parent_order_id": null,
            "modification_number": null,
            "exchange_acknowledgement_date": null,
            "SLTP_price": null,
            "exchange_acknowledge_number": null,
            "initial_limit": null,
            "intial_sltp": null,
            "LTP": null,
            "limit_offset": null,
            "mbc_flag": null,
            "cutoff_price": null
        }
    ],
    "Status": 200,
    "Error": null
}
```

REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
exchange_code	String	"NSE", "NFO", "BSE"	Yes
order_id	String	Order Reference to get detailed data of order	Yes

---

GetOrderList

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "NSE",
    "from_date": "2022-05-29T10:00:00.000Z",
    "to_date": "2022-05-31T10:00:00.000Z"
})

#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "order_id": "20220531N100000107",
            "exchange_order_id": null,
            "exchange_code": "NSE",
            "stock_code": "ITC",
            "product_type": "Cash",
            "action": "Buy",
            "order_type": "Limit",
            "stoploss": "0.00",
            "quantity": "1",
            "price": "265.00",
            "validity": "",
            "disclosed_quantity": "0",
            "expiry_date": null,
            "right": null,
            "strike_price": 0,
            "average_price": "0",
            "cancelled_quantity": "0",
            "pending_quantity": "0",
            "status": "Expired",
            "user_remark": "",
            "order_datetime": "31-May-2022 15:29",
            "parent_order_id": null,
            "modification_number": null,
            "exchange_acknowledgement_date": null,
            "SLTP_price": null,
            "exchange_acknowledge_number": null,
            "initial_limit": null,
            "intial_sltp": null,
            "LTP": null,
            "limit_offset": null,
            "mbc_flag": null,
            "cutoff_price": null
        },
        {
            "order_id": "20220530N100000185",
            "exchange_order_id": null,
            "exchange_code": "NSE",
            "stock_code": "ITC",
```

```
        "product_type": "Cash",
        "action": "Buy",
        "order_type": "Limit",
        "stoploss": "0.00",
        "quantity": "1",
        "price": "269.05",
        "validity": "",
        "disclosed_quantity": "0",
        "expiry_date": null,
        "right": null,
        "strike_price": 0,
        "average_price": "0",
        "cancelled_quantity": "1",
        "pending_quantity": "0",
        "status": "Cancelled",
        "user_remark": "",
        "order_datetime": "30-May-2022 18:24",
        "parent_order_id": null,
        "modification_number": null,
        "exchange_acknowledgement_date": null,
        "SLTP_price": null,
        "exchange_acknowledge_number": null,
        "initial_limit": null,
        "intial_sltp": null,
        "LTP": null,
        "limit_offset": null,
        "mbc_flag": null,
        "cutoff_price": null
    },
    {
        "order_id": "20220530N100000184",
        "exchange_order_id": null,
        "exchange_code": "NSE",
        "stock_code": "IDFC",
        "product_type": "Cash",
        "action": "Buy",
        "order_type": "Limit",
        "stoploss": "0.00",
        "quantity": "2",
        "price": "49.80",
        "validity": "",
        "disclosed_quantity": "0",
        "expiry_date": null,
        "right": null,
        "strike_price": 0,
        "average_price": "0",
        "cancelled_quantity": "0",
        "pending_quantity": "0",
        "status": "Expired",
        "user_remark": "",
        "order_datetime": "30-May-2022 18:24",
        "parent_order_id": null,
        "modification_number": null,
        "exchange_acknowledgement_date": null,
        "SLTP_price": null,
        "exchange_acknowledge_number": null,
        "initial_limit": null,
        "intial_sltp": null,
        "LTP": null,
        "limit_offset": null,
        "mbc_flag": null,
        "cutoff_price": null
    },
    {
        "order_id": "20220530N100000183",
        "exchange_order_id": null,
        "exchange_code": "NSE",
        "stock_code": "DLFLIM",
        "product_type": "Cash",
        "action": "Buy",
        "order_type": "Limit",
        "stoploss": "0.00",
        "quantity": "1",
        "price": "346.85",
        "validity": "",
        "disclosed_quantity": "0",
        "expiry_date": null,
        "right": null,
        "strike_price": 0,
        "average_price": "0",
        "cancelled_quantity": "0",
        "pending_quantity": "0",
        "status": "Expired",
        "user_remark": "",
        "order_datetime": "30-May-2022 18:21",
        "parent_order_id": null,
        "modification_number": null,
        "exchange_acknowledgement_date": null,
        "SLTP_price": null,
        "exchange_acknowledge_number": null,
        "initial_limit": null,
        "intial_sltp": null,
        "LTP": null,
        "limit_offset": null,
        "mbc_flag": null,
        "cutoff_price": null
    },
    {
        "order_id": "20220530N100000182",
        "exchange_order_id": null,
        "exchange_code": "NSE",
        "stock_code": "DLFLIM",
```

```
            "product_type": "Cash",
            "action": "Buy",
            "order_type": "Limit",
            "stoploss": "0.00",
            "quantity": "1",
            "price": "346.65",
            "validity": "",
            "disclosed_quantity": "0",
            "expiry_date": null,
            "right": null,
            "strike_price": 0,
            "average_price": "0",
            "cancelled_quantity": "1",
            "pending_quantity": "0",
            "status": "Cancelled",
            "user_remark": "",
            "order_datetime": "30-May-2022 18:20",
            "parent_order_id": null,
            "modification_number": null,
            "exchange_acknowledgement_date": null,
            "SLTP_price": null,
            "exchange_acknowledge_number": null,
            "initial_limit": null,
            "intial_sltp": null,
            "LTP": null,
            "limit_offset": null,
            "mbc_flag": null,
            "cutoff_price": null
        },
        {
            "order_id": "20220530N100000081",
            "exchange_order_id": null,
            "exchange_code": "NSE",
            "stock_code": "ITC",
            "product_type": "Cash",
            "action": "Buy",
            "order_type": "Limit",
            "stoploss": "0.00",
            "quantity": "1",
            "price": "265.00",
            "validity": "",
            "disclosed_quantity": "0",
            "expiry_date": null,
            "right": null,
            "strike_price": 0,
            "average_price": "0",
            "cancelled_quantity": "0",
            "pending_quantity": "0",
            "status": "Expired",
            "user_remark": "",
            "order_datetime": "30-May-2022 12:49",
            "parent_order_id": null,
            "modification_number": null,
            "exchange_acknowledgement_date": null,
            "SLTP_price": null,
            "exchange_acknowledge_number": null,
            "initial_limit": null,
            "intial_sltp": null,
            "LTP": null,
            "limit_offset": null,
            "mbc_flag": null,
            "cutoff_price": null
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
exchange_code	String	"NSE", "NFO", "BSE"	Yes
from_date	String	ISO 8601 - Day should not be less than 10 days from to_date	No
to_date	String	ISO 8601 - Day should not be more than 10 days from from_date	No

OrderCancellation

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
```

```
payload = json.dumps({
    "order_id": "20220601N100000019",
    "exchange_code": "NSE"
})
#checksum computation
#time_stamp & checksum generation for request-headers

secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
appkey = "9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("DELETE", "/breezeapi/api/v1/order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "order_id": "20220601N100000019",
        "message": "Your Order Canceled successfully."
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	DELETE
URL	https://api.icicidirect.com/breezeapi/api/v1/order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
order_id	String	Order Reference to cancel order	Yes
exchange_code	String	"NSE", "NFO", "BSE"	Yes

OrderModification

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "order_id": "20220601N100000019",
    "exchange_code": "NSE",
    "quantity": "2",
    "price": "263.15",
    "stoploss": "",
    "disclosed_quantity": "",
    "order_type": "limit",
    "validity": "",
    "expiry_date": "",
    "right": "",
    "strike_price": ""
})

#checksum computation
#time_stamp & checksum generation for request-headers

secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
appkey = ""9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("PUT", "/breezeapi/api/v1/order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "message": "Your Order Modified successfully.",
        "order_id": "20220601N100000019"
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	PUT
URL	https://api.icicidirect.com/breezeapi/api/v1/order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
order_id	String	Order Reference to update data of order	Yes
exchange_code	String	"NSE", "NFO", "BSE"	Yes
order_type	String	"limit","market","stoploss"	No
stoploss	String	Numeric String of Currency	No
quantity	String	Modify number of quantity on placed order	No
price	String	Numeric String of Currency	No
validity	String	"day","ioc","vtc"	No
disclosed_quantity	String	Modify number of disclosed quantity on placed order	No
expiry_date	String	ISO 8601	No
right	String	"call","put","others"	No
strike_price	Integer	Numeric Currency	No

Breeze Limit price calculation

limit calculator

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "strike_price": "19200",
    "product_type": "optionplus",
    "expiry_date": "13-JUL-2023",
    "underlying": "NIFTY",
    "exchange_code": "NFO",
    "order_flow": "Buy",
    "stop_loss_trigger": "200.00",
    "option_type": "Call",
    "source_flag": "P",
    "limit_rate": "",
    "order_reference": "",
    "available_quantity": "",
    "market_type": "limit",
    "fresh_order_limit": "218.65"
})

#checksum computation

#time_stamp & checksum generation for request-headers
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token ' + checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': enter your base64sessiontoken
}
```

```
conn.request("POST", "/breezeapi/api/v1/fnolmtpriceandqtycal", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{ 'Success':
{
    'available_quantity': '0',
  'action_id': '0',
   'order_margin': '0',
   'limit_rate': '224'
},
 'Status': 200,
 'Error': None
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/fnolmtpriceandqtycal

BODY PARAMETERS

Parameter	Data Type	Description
strike_price	String	Numeric String of Currency
product_type	String	"optionplus"
expiry_date	String	ISO 8601
exchange_code	String	"NSE","BSE"
stop_loss_trigger	Double	Numeric Currency
option_type	String	"call","put"
order_reference	String	--
available_quantity	--	--
market_type	String	"limit","market"
fresh_order_limit	Double	Numeric currency

PortfolioHoldings

GetPortfolioHoldings

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "NSE",
    "from_date": "",
    "to_date": "",
    "stock_code": "JKPAP",
    "portfolio_type": ""
})

#checksum computation
#time_stamp & checksum generation for request-headers

secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
appkey = "9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("GET", "/breezeapi/api/v1/portfolioholdings", payload, headers)
```

```
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "stock_code": "JKPAP",
            "exchange_code": "BSE",
            "quantity": "900",
            "average_price": "151.67",
            "booked_profit_loss": "-63660.16",
            "current_market_price": "349.1",
            "change_percentage": "121.229404309252",
            "answer_flag": "N",
            "product_type": null,
            "expiry_date": null,
            "strike_price": null,
            "right": null,
            "category_index_per_stock": null,
            "action": null,
            "realized_profit": null,
            "unrealized_profit": null,
            "open_position_value": null,
            "portfolio_charges": null
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/portfolioholdings

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
exchange_code	String	"NSE", "NFO", "BSE"	Yes
from_date	String	ISO 8601	No
to_date	String	ISO 8601	No
stock_code	String	"AXIBAN", "TATMOT"	No
portfolio_type	String	--	No

PortfolioPositions

GetPortfolioPositions

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({})

#checksum computation
#time_stamp & checksum generation for request-headers

secret_key = "9e12o8179&J695!141J150147277V7"
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()
appkey = "9e12o8179&J695!`141J150147277(V7"

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': timestamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/portfoliopositions", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "segment": "fno",
            "product_type": "Futures",
            "trade_date": null,
            "exchange_code": "NFO",
            "stock_code": null,
            "expiry_date": "30-Jun-2022",
            "right": "Others",
            "stock_index_indicator": "Stock",
            "action": "",
            "quantity": "150",
            "price": "0.01",
            "cover_quantity": "150",
            "stoploss_trigger": "3451.35",
            "stoploss": null,
            "take_profit": null,
            "ltp": "3561.45",
            "available_margin": null,
            "squareoff_mode": null,
            "mtf_sell_quantity": null,
            "mtf_net_amount_payable": null,
            "mtf_expiry_date": null,
            "order_id": "",
            "cover_order_flow": null,
            "cover_order_executed_quantity": null,
            "pledge_status": null,
            "strike_price": "0",
            "underlying": "TCS"
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/portfoliopositions

BODY PARAMETERS

Parameter	Data Type	Description
--	--	--

Quotes

GetQuotes

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "stock_code": "CNXBAN",
    "exchange_code": "NFO",
    "expiry_date": "2022-05-26T06:00:00.000Z",
    "product_type": "Futures",
    "right": "Others",
    "strike_price": "0"
})
secret_key = "9e12o8179&J695!141J150147277V7"
appkey = "9e12o8179&J695!`141J150147277(V7"

#checksum computation
#time_stamp & checksum generation for request-headers

time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken':  Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/quotes", payload, headers)
res = conn.getresponse()
```

```
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
   "Success": [
      {
         "exchange_code": "NFO",
         "product_type": "Future",
         "stock_code": "CNXBAN",
         "expiry_date": "26-May-2022",
         "right": "*",
         "strike_price": 0,
         "ltp": 35310,
         "ltt": "07-May-2022 11:53:15",
         "best_bid_price": 35310,
         "best_bid_quantity": "9725",
         "best_offer_price": 35390,
         "best_offer_quantity": "1225",
         "open": 37860,
         "high": 38089.75,
         "low": 33255.65,
         "previous_close": 34140.2,
         "ltp_percent_change": 196,
         "upper_circuit": 40160.05,
         "lower_circuit": 32858.2,
         "total_quantity_traded": "101775",
         "spot_price": "33765.25"
      }
   ],
   "Status": 200,
   "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/quotes

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
stock_code	String	"AXIBAN","TATMOT"	Yes
exchange_code	String	"NSE","NFO","BSE"	Yes
expiry_date	String	ISO 8601	No
product_type	String	"futures","options","furtureplus","futureplus_sltp","optionplus", "cash","eatm","btst","margin","marginplus"	No
right	String	"call","put","others"	No
strike_price	String	Numeric String of Currency	No

SquareOff

SquareOff

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "source_flag": "",
    "stock_code": "NIFTY",
    "exchange_code": "NFO",
    "quantity": "50",
    "price": "0",
    "action": "Sell",
    "order_type": "Market",
    "validity": "Day",
    "stoploss_price": "0",
    "disclosed_quantity": "0",
    "protection_percentage": "",
    "settlement_id": "",
    "margin_amount": "",
    "open_quantity": "",
    "cover_quantity": "",
    "product_type": "Futures",
    "expiry_date": "2021-12-30",
    "right": "Others",
```

```
    "strike_price": "0",
    "validity_date": "2021-12-16T06:00:00.000Z",
    "alias_name": "",
    "trade_password": ""
})

#checksum computation
#time_stamp & checksum generation for request-headers
secret_key = "9e12o8179&J695!141J150147277V7"
appkey = "9e12o8179&J695!`141J150147277(V7"

time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("POST", "/breezeapi/api/v1/squareoff", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "order_id": "20211116110000232",
        "message": "Successfully Placed the order",
        "indicator": "0"
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	POST
URL	https://api.icicidirect.com/breezeapi/api/v1/squareoff

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
source_flag	String	--	No
stock_code	String	"NIFTY","SENSEX"	No
exchange_code	String	"NSE","NFO","BSE"	Yes
quantity	String	Number of quantity to sqaure off the orders	No
price	String	Numeric String of Currency	No
action	String	"buy","sell"	No
order_type	String	"limit","market","stoploss"	No
validity	String	"day","ioc","vtc"	No
stoploss_price	String	Numeric String of Currency	No
disclosed_quantity	String	Number of disclosed quantity to squareoff order	No
protection_percentage	String	--	No
settlement_id	String	Numeric String of Currency	No
margin_amount	String	Numeric String of Currency	No
open_quantity	String	Number of open quantity to squareoff order	No
cover_quantity	String	Number of cover quantity to squareoff order	No
product_type	String	"futures","options","cash","eatm","margin"	No
expiry_date	String	Date when squareoff order will be expired	No
right	String	"call","put","others"	No
strike_price	String	Numeric String of Currency	No
validity_date	String	Validity date of squareoff order	No

Parameter	Data Type	Description	Mandatory
alias_name	--	--	
trade_password	String	--	No

Trades

GetTradeList

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "from_date": "2022-05-28T06:00:00.000Z",
    "to_date": "2022-06-01T06:00:00.000Z",
    "exchange_code": "NSE",
    "product_type": "",
    "action": "",
    "stock_code": ""
})
appkey = "9e12o8179&J695!`141J150147277(V7"
secret_key = ""9e12o8179&J695141J150147277V7"

#checksum computation
#time_stamp & checksum generation for request-headers

time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/trades", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
   "Success": {
       "trade_book": [
           {
               "match_account": "8509003752",
               "order_trade_date": "28-Sep-2021",
               "order_stock_code": "LARTOU",
               "order_flow": "S",
               "order_quantity": "1",
               "order_average_executed_rate": "1720.90",
               "order_trans_value": "1720.90",
               "order_brokerage": "15.00",
               "order_product": "M",
               "order_exchange_code": "NSE",
               "order_reference": "20210928N100000067",
               "order_segment_code": "N",
               "order_settlement": "2021183",
               "dp_id": "IN303028",
               "client_id": "80003129",
               "LTP": "1952.00",
               "order_eATM_withheld": "0.00",
               "order_csh_withheld": "0.00",
               "order_total_taxes": "3.21",
               "order_type": "77"
           },
           {
               "match_account": "8509003752",
               "order_trade_date": "28-Sep-2021",
               "order_stock_code": "RELIND",
               "order_flow": "S",
               "order_quantity": "1",
               "order_average_executed_rate": "2404.69",
               "order_trans_value": "2404.69",
               "order_brokerage": "15.00",
               "order_product": "M",
               "order_exchange_code": "NSE",
               "order_reference": "20210928N100000069",
               "order_segment_code": "N",
               "order_settlement": "2021183",
               "dp_id": "IN303028",
               "client_id": "80003129",
               "LTP": "2529.90",
               "order_eATM_withheld": "0.00",
               "order_csh_withheld": "0.00",
```

```
            "order_total_taxes": "3.40",
            "order_type": "77"
        },
        {
            "match_account": "8509003752",
            "order_trade_date": "28-Sep-2021",
            "order_stock_code": "LARTOU",
            "order_flow": "B",
            "order_quantity": "1",
            "order_average_executed_rate": "1720.90",
            "order_trans_value": "1720.90",
            "order_brokerage": "15.00",
            "order_product": "M",
            "order_exchange_code": "NSE",
            "order_reference": "20210928N100000074",
            "order_segment_code": "N",
            "order_settlement": "2021183",
            "dp_id": "IN303028",
            "client_id": "80003129",
            "LTP": "1952.00",
            "order_eATM_withheld": "0.00",
            "order_csh_withheld": "0.00",
            "order_total_taxes": "2.78",
            "order_type": "77"
        },
        {
            "match_account": "8509003752",
            "order_trade_date": "28-Sep-2021",
            "order_stock_code": "RELIND",
            "order_flow": "B",
            "order_quantity": "1",
            "order_average_executed_rate": "2404.69",
            "order_trans_value": "2404.69",
            "order_brokerage": "15.00",
            "order_product": "M",
            "order_exchange_code": "NSE",
            "order_reference": "20210928N100000075",
            "order_segment_code": "N",
            "order_settlement": "2021183",
            "dp_id": "IN303028",
            "client_id": "80003129",
            "LTP": "2529.90",
            "order_eATM_withheld": "0.00",
            "order_csh_withheld": "0.00",
            "order_total_taxes": "2.80",
            "order_type": "77"
        }
    ]
},
"Status": 200,
"Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/trades

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
from_date	String	ISO 8601	No
to_date	String	ISO 8601	No
exchange_code	String	"NSE","NFO","BSE"	Yes
product_type	String	"futures","options","furtureplus","futureplus_sltp","optionplus", "cash","eatm","btst","margin","marginplus"	No
action	String	"buy","sell"	No
stock_code	String	"AXIBAN","TATMOT"	No

GetTradeDetail

```
import http.client
import json

secret_key = "9e12o8179&J695!141J150147277V7"
appkey = "9e12o8179&J695!141J150147277V7"

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "exchange_code": "NSE",
    "order_id": "20210928N100000067"
})
```

```
#checksum computation
#time_stamp & checksum generation for request-headers

time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/trades", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
            "order_settlement": "2021183",
            "order_exchange_trade_number": "81",
            "order_executed_quantity": "1",
            "order_flow": "S",
            "order_brokerage": "17.71",
            "order_pure_brokerage": "15",
            "order_taxes": "3.2056",
            "order_eATM_withheld": "0",
            "order_cash_withheld": "0",
            "order_executed_rate": "1720.9",
            "order_stock_code": "LARTOU",
            "order_exchange_code": "NSE",
            "match_account": "8509003752",
            "order_trade_reference": "2021/0928/00587863",
            "order_exchange_trade_tm": "28-Sep-2021 12:02:02",
            "order_segment_dis": "Rolling",
            "order_segment_code": "N"
        }
    ],
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	`GET`
URL	https://api.icicidirect.com/breezeapi/api/v1/trades

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
exchange_code	String	"NSE","NFO","BSE"	Yes
order_id	String	Order Reference to get detailed data of trade	Yes

OptionChain

GetOptionChain

```
import http.client
import json
import hashlib
from datetime import datetime


# App related Secret Key
secret_key = '9e12o8179&J695!141J150147277V7'

# here is the example of payload
payload = json.dumps({
    "stock_code": "NIFTY",
    "exchange_code": "NFO",
    "expiry_date": "2022-12-15T06:00:00.000Z",
    "product_type": "options",
    "right": "call",
    "strike_price": "19000"
})

appkey = "9e12o8179&J695!`141J150147277(V7"
```

```
#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()

headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn = http.client.HTTPSConnection("api.icicidirect.com")
conn.request("GET", "/breezeapi/api/v1/OptionChain", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
   'Success': [
      {
          'exchange_code': 'NFO',
          'product_type': 'Options',
          'stock_code': 'NIFTY',
          'expiry_date': '15-Dec-2022',
          'right': 'Call',
          'strike_price': 19000.0,
          'ltp': 0.85,
          'ltt': '15-Dec-2022 13:50:55',
          'best_bid_price': 0.85,
          'best_bid_quantity': '68050',
          'best_offer_price': 0.9,
          'best_offer_quantity': '27450',
          'open': 0.85,
          'high': 1.7,
          'low': 0.7,
          'previous_close': 1.6,
          'ltp_percent_change': -4688.0,
          'upper_circuit': 38.05,
          'lower_circuit': 0.05,
          'total_quantity_traded': '69463100',
          'spot_price': '18508',
          'ltq': '50',
          'open_interest': 13010500.0,
          'chnge_oi': 20.66,
          'total_buy_qty': '16912500',
          'total_sell_qty': '284650'
          }
      ],
          'Status': 200,
          'Error': None
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/optionchain

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
stock_code	String	"NIFTY","ICIBAN","AXIBAN"	Yes
exchange_code	String	"NFO"	Yes
expiry_date	String	ISO 8601	No
product_type	String	"options","futures"	Yes
right	String	"call","put","others"	No
strike_price	String	Numeric String Of Currency	No

Note : atleast 2 of the 3 (strike_price, right, expirydate) field should be entered by user.

Preview Order

GetBrokeragecharges - Equity

```
import http.client
import json

# App related Secret Key
secret_key = '9e12o8179&J695!141J150147277V7'

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "stock_code": "ICIBAN",
    "exchange_code": "NSE",
    "product": "margin",
    "order_type": "limit",
    "price": "907.05",
    "action": "buy",
    "quantity": "1",
    "specialflag": "N"
})

appkey = "9e12o8179&J695!`141J150147277(V7"

#checksum computation
#time_stamp & checksum generation for request-headers

time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}

conn.request("GET", "/breezeapi/api/v1/preview_order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": {
        "brokerage": 6.8029,
        "exchange_turnover_charges": 0.0254,
        "stamp_duty": 0.1361,
        "stt": 0.9071,
        "sebi_charges": 0.0009,
        "gst": 1.2293,
        "total_turnover_and_sebi_charges": 0.0263,
        "total_other_charges": 2.2987,
        "total_brokerage": 9.1015
    },
    "Status": 200,
    "Error": null
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/preview_order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
stock_code	String	"NIFTY","SENSEX"	Yes
exchange_code	String	"NSE","BSE"	Yes
product	String	"futures","options","optionplus","margin"	Yes
order_type	String	"limit","market","stoploss"	Yes
price	String	Numeric String of Currency	Yes
action	String	"buy","sell"	Yes
quantity	String	Number of quantity in string formatted	Yes

GetBrokeragecharges - Fno

```
import http.client
import json

conn = http.client.HTTPSConnection("api.icicidirect.com")
payload = json.dumps({
    "stock_code": "NIFTY",
    "exchange_code": "NFO",
    "product": "optionplus",
    "order_type": "limit",
    "price": "171",
    "action": "buy",
    "quantity": "50",
    "expiry_date": "25-Jan-2023",
    "right": "call",
    "strike_price": "19000",
    "specialflag": "S",
    "stoploss": "190",
    "order_rate_fresh": "195.50"
})
appkey = "9e12o8179&J695!`141J150147277(V7"

secret_key= "9e12o8179&J695!141J150147277V7"
#checksum computation
#time_stamp & checksum generation for request-headers
time_stamp = datetime.utcnow().isoformat()[:19] + '.000Z'
checksum = hashlib.sha256((time_stamp+payload+secret_key).encode("utf-8")).hexdigest()


headers = {
    'Content-Type': 'application/json',
    'X-Checksum': 'token '+checksum,
    'X-Timestamp': time_stamp,
    'X-AppKey': appkey,
    'X-SessionToken': Session token as received  from customer detail api response
}
conn.request("GET", "/breezeapi/api/v1/preview_order", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://api.icicidirect.com/breezeapi/api/v1/preview_order

BODY PARAMETERS

Parameter	Data Type	Description	Mandatory
stock_code	String	"NIFTY","SENSEX"	Yes
exchange_code	String	"NFO"	Yes
product	String	"futures","options","optionplus","margin"	Yes
order_type	String	"limit","market","stoploss"	Yes
price	String	Numeric String of Currency	Yes
action	String	"buy","sell"	Yes
quantity	String	Number of quantity in string formatted	Yes
expiry_date	String	ISO 8601	Yes
right	String	"call","put","others"	Yes
strike_price	String	Numeric Currency	Yes
stoploss	String	Numeric Currency	Yes
order_rate_fresh	String	Numeric Currency	Yes

HistoricalChartsv2

GetHistoricalChartsListv2

```
import http.client

conn = http.client.HTTPSConnection("breezeapi.icicidirect.com")
appkey = "9e12o8179&J695!`141J150147277(V7"
payload = None
headers = {
```

```
    'X-SessionToken': 'insert your sesion token from customer detail API call',
    'apikey': appkey
}
conn.request("GET", "/api/v2/historicalcharts?stock_code=NIFTY&exch_code=NFO&from_date=2022-11-10T00:00:00.000Z&to_date=2022-11-11T00:00:00.000Z&interval=1day&product_type=Options&expiry_date=2022-11-24T00:00:00.000Z&right=Call&strike_price=18000", payload, he
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Sample JSON Response


```
{
    "Success": [
        {
        "close": "253.50",
        "datetime": "2022-11-10 00:00:00",
        "exchange_code": "NFO",
        "expiry_date": "24-NOV-2022",
        "high": "296.45",
        "low": "221.25",
        "open": "289.70",
        "open_interest": 2757700,
        "product_type": "Options",
        "right": "Call",
        "stock_code": "NIFTY",
        "strike_price": "18000",
        "volume": 3881000
        },
        {
        "close": "486.00",
        "datetime": "2022-11-11 00:00:00",
        "exchange_code": "NFO",
        "expiry_date": "24-NOV-2022",
        "high": "486.00",
        "low": "390.15",
        "open": "418.70",
        "open_interest": 12250,
        "product_type": "Options",
        "right": "Call",
        "stock_code": "NIFTY",
        "strike_price": "18000",
        "volume": 826929750
        }
    ],
    "Error": null,
    "Status": 200
}
```


REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://breezeapi.icicidirect.com/api/v2/historicalcharts

HEADERS

Key	Description
X-SessionToken	Session Token
apikey	API Key

QUERY PARAMETERS

Parameter	Data Type	Description	Mandatory
interval	String	"1second","1minute","5minute","30minute","1day"	Yes
from_date	String	ISO 8601	Yes
to_date	String	ISO 8601	Yes
stock_code	String	"AXIBAN", "TATMOT"	Yes
exch_code	String	"NSE", "NFO", "BSE","NDX","MCX"	Yes
product_type	String	Cash, Options,Futures (Required for NFO, NDX,MCX)	No
expiry_date	String	ISO 8601 (Required for NFO, NDX,MCX)	No
right	String	Call, Put, Others (Required for NFO, NDX,MCX Options)	No
strike_price	String	Numeric String of Currency (Required for NFO, NDX,MCX Options)	No

Order Notifications

live feeds for Order Updates

```
import base64
import socketio

#Get User ID and Session Token
session_key = "SESSION_TOKEN_FROM_CUSTOMER_DETAILS_API"
#e.g session_key = "QUYyOTUzMTM6NjY5ODc5NzY="

user_id, session_token = base64.b64decode(session_key.encode('ascii')).decode('ascii').split(":")
#e.g Decoded value - AF296713:66987976, after split user_id = AF295313, session_token = 6698797

# Python Socket IO Client
sio = socketio.Client()

auth = {"user": user_id, "token": session_token}

sio.connect("https://livefeeds.icicidirect.com", headers={"User-Agent":"python-socketio[client]/socket"},
               auth=auth, transports="websocket", wait_timeout=3)

tux_to_user_value =
{
    "orderFlow": {
        "B": "Buy",
        "S": "Sell",
        "N": "NA"
    },
    "limitMarketFlag": {
        "L": "Limit",
        "M": "Market",
        "S": "StopLoss"
    },
    "orderType": {
        "T": "Day",
        "I": "IoC",
        "V": "VTC"
    },
    "productType": {
        "F": "Futures",
        "O": "Options",
        "P": "FuturePlus",
        "U": "FuturePlus_sltp",
        "I": "OptionPlus",
        "C": "Cash",
        "Y": "eATM",
        "B": "BTST",
        "M": "Margin",
        "T": "MarginPlus"
    },
    "orderStatus": {
        "A": "All",
        "R": "Requested",
        "Q": "Queued",
        "O": "Ordered",
        "P": "Partially Executed",
        "E": "Executed",
        "J": "Rejected",
        "X": "Expired",
        "B": "Partially Executed And Expired",
        "D": "Partially Executed And Cancelled",
        "F": "Freezed",
        "C": "Cancelled"
    },
    "optionType": {
        "C": "Call",
        "P": "Put",
        "*": "Others"
    },
}


# parse market depth

def parse_market_depth(self, data, exchange):
    depth = []
    counter = 0
    for lis in data:
        counter += 1
        dict = {}
        if exchange == '1':
            dict["BestBuyRate-"+str(counter)] = lis[0]
            dict["BestBuyQty-"+str(counter)] = lis[1]
            dict["BestSellRate-"+str(counter)] = lis[2]
            dict["BestSellQty-"+str(counter)] = lis[3]
            depth.append(dict)
        else:
            dict["BestBuyRate-"+str(counter)] = lis[0]
            dict["BestBuyQty-"+str(counter)] = lis[1]
            dict["BuyNoOfOrders-"+str(counter)] = lis[2]
            dict["BuyFlag-"+str(counter)] = lis[3]
            dict["BestSellRate-"+str(counter)] = lis[4]
            dict["BestSellQty-"+str(counter)] = lis[5]
            dict["SellNoOfOrders-"+str(counter)] = lis[6]
```

```
            dict["SellFlag-"+str(counter)] = lis[7]
            depth.append(dict)
    return depth


# parsing logic
def parse_data(data):
    if data and type(data) == list and len(data) > 0 and type(data[0]) == str and "!" not in data[0]:
        order_dict = {}
        order_dict["sourceNumber"] = data[0]                        #Source Number
        order_dict["group"] = data[1]                               #Group
        order_dict["userId"] = data[2]                              #User_id
        order_dict["key"] = data[3]                                 #Key
        order_dict["messageLength"] = data[4]                       #Message Length
        order_dict["requestType"] = data[5]                         #Request Type
        order_dict["messageSequence"] = data[6]                     #Message Sequence
        order_dict["messageDate"] = data[7]                         #Date
        order_dict["messageTime"] = data[8]                         #Time
        order_dict["messageCategory"] = data[9]                     #Message Category
        order_dict["messagePriority"] = data[10]                    #Priority
        order_dict["messageType"] = data[11]                        #Message Type
        order_dict["orderMatchAccount"] = data[12]                  #Order Match Account
        order_dict["orderExchangeCode"] = data[13]                  #Exchange Code
        if data[11] == '4' or data[11] == '5':
            order_dict["stockCode"] = data[14]                  #Stock Code
            order_dict["orderFlow"] = tux_to_user_value['orderFlow'].get(str(data[15]).upper(),str(data[15]))                       # Order Flow
            order_dict["limitMarketFlag"] = tux_to_user_value['limitMarketFlag'].get(str(data[16]).upper(),str(data[16]))                 #Limit Market Flag
            order_dict["orderType"] = tux_to_user_value['orderType'].get(str(data[17]).upper(),str(data[17]))                       #OrderType
            order_dict["orderLimitRate"] = data[18]                 #Limit Rate
            order_dict["productType"] = tux_to_user_value['productType'].get(str(data[19]).upper(),str(data[19]))                     #Product Type
            order_dict["orderStatus"] = tux_to_user_value['orderStatus'].get(str(data[20]).upper(),str(data[20]))                     # Order Status
            order_dict["orderDate"] = data[21]                      #Order  Date
            order_dict["orderTradeDate"] = data[22]                 #Trade Date
            order_dict["orderReference"] = data[23]                 #Order Reference
            order_dict["orderQuantity"] = data[24]                  #Order Quantity
            order_dict["openQuantity"] = data[25]                   #Open Quantity
            order_dict["orderExecutedQuantity"] = data[26]          #Order Executed Quantity
            order_dict["cancelledQuantity"] = data[27]              #Cancelled Quantity
            order_dict["expiredQuantity"] = data[28]                #Expired Quantity
            order_dict["orderDisclosedQuantity"] = data[29]         # Order Disclosed Quantity
            order_dict["orderStopLossTrigger"] = data[30]           #Order Stop Loss Triger
            order_dict["orderSquareFlag"] = data[31]                #Order Square Flag
            order_dict["orderAmountBlocked"] = data[32]             # Order Amount Blocked
            order_dict["orderPipeId"] = data[33]                    #Order PipeId
            order_dict["channel"] = data[34]                        #Channel
            order_dict["exchangeSegmentCode"] = data[35]            #Exchange Segment Code
            order_dict["exchangeSegmentSettlement"] = data[36]      #Exchange Segment Settlement
            order_dict["segmentDescription"] = data[37]             #Segment Description
            order_dict["marginSquareOffMode"] = data[38]            #Margin Square Off Mode
            order_dict["orderValidDate"] = data[40]                 #Order Valid Date
            order_dict["orderMessageCharacter"] = data[41]          #Order Message Character
            order_dict["averageExecutedRate"] = data[42]            #Average Exited Rate
            order_dict["orderPriceImprovementFlag"] = data[43]      #Order Price Flag
            order_dict["orderMBCFlag"] = data[44]                   #Order MBC Flag
            order_dict["orderLimitOffset"] = data[45]               #Order Limit Offset
            order_dict["systemPartnerCode"] = data[46]              #System Partner Code
        elif data[11] == '6' or data[11] == '7':
            order_dict["stockCode"] = data[14]                     #stockCode
            order_dict["productType"] =  tux_to_user_value['productType'].get(str(data[15]).upper(),str(data[15]))                    #Product Type
            order_dict["optionType"] = tux_to_user_value['optionType'].get(str(data[16]).upper(),str(data[16]))                     #Option Type
            order_dict["exerciseType"] = data[17]                   #Exercise Type
            order_dict["strikePrice"] = data[18]                    #Strike Price
            order_dict["expiryDate"] = data[19]                     #Expiry Date
            order_dict["orderValidDate"] = data[20]                 #Order Valid Date
            order_dict["orderFlow"] = tux_to_user_value['orderFlow'].get(str(data[21]).upper(),str(data[21]))                       #Order  Flow
            order_dict["limitMarketFlag"] = tux_to_user_value['limitMarketFlag'].get(str(data[22]).upper(),str(data[22]))                 #Limit Market Flag
            order_dict["orderType"] = tux_to_user_value['orderType'].get(str(data[23]).upper(),str(data[23]))                       #Order Type
            order_dict["limitRate"] = data[24]                      #Limit Rate
            order_dict["orderStatus"] = tux_to_user_value['orderStatus'].get(str(data[25]).upper(),str(data[25]))                     #Order Status
            order_dict["orderReference"] = data[26]                 #Order Reference
            order_dict["orderTotalQuantity"] = data[27]             #Order Total Quantity
            order_dict["executedQuantity"] = data[28]               #Executed Quantity
            order_dict["cancelledQuantity"] = data[29]              #Cancelled Quantity
            order_dict["expiredQuantity"] = data[30]                #Expired Quantity
            order_dict["stopLossTrigger"] = data[31]                #Stop Loss Trigger
            order_dict["specialFlag"] = data[32]                    #Special Flag
            order_dict["pipeId"] = data[33]                         #PipeId
            order_dict["channel"] = data[34]                        #Channel
            order_dict["modificationOrCancelFlag"] = data[35]       #Modification or Cancel Flag
            order_dict["tradeDate"] = data[36]                      #Trade Date
            order_dict["acknowledgeNumber"] = data[37]              #Acknowledgement Number
            order_dict["stopLossOrderReference"] = data[37]         #Stop Loss Order Reference
            order_dict["totalAmountBlocked"] = data[38]             # Total Amount Blocked
            order_dict["averageExecutedRate"] = data[39]            #Average Executed Rate
            order_dict["cancelFlag"] = data[40]                     #Cancel Flag
            order_dict["squareOffMarket"] = data[41]                #SquareOff Market
            order_dict["quickExitFlag"] = data[42]                  #Quick Exit Flag
            order_dict["stopValidTillDateFlag"] = data[43]          #Stop Valid till Date Flag
            order_dict["priceImprovementFlag"] = data[44]           #Price Improvement Flag
            order_dict["conversionImprovementFlag"] = data[45]      #Conversion Improvement Flag
            order_dict["trailUpdateCondition"] = data[45]           #Trail Update Condition
            order_dict["systemPartnerCode"] = data[46]              #System Partner Code
        return order_dict
    exchange = str.split(data[0], '!')[0].split('.')[0]
    data_type = str.split(data[0], '!')[0].split('.')[1]
    if exchange == '6':
        data_dict = {}
        data_dict["symbol"] = data[0]
        data_dict["AndiOPVolume"] = data[1]
        data_dict["Reserved"] = data[2]
        data_dict["IndexFlag"] = data[3]
        data_dict["ttq"] = data[4]
```

```
        data_dict["last"] = data[5]
        data_dict["ltq"] = data[6]
        data_dict["ltt"] = datetime.fromtimestamp(data[7]).strftime('%c')
        data_dict["AvgTradedPrice"] = data[8]
        data_dict["TotalBuyQnt"] = data[9]
        data_dict["TotalSellQnt"] = data[10]
        data_dict["ReservedStr"] = data[11]
        data_dict["ClosePrice"] = data[12]
        data_dict["OpenPrice"] = data[13]
        data_dict["HighPrice"] = data[14]
        data_dict["LowPrice"] = data[15]
        data_dict["ReservedShort"] = data[16]
        data_dict["CurrOpenInterest"] = data[17]
        data_dict["TotalTrades"] = data[18]
        data_dict["HightestPriceEver"] = data[19]
        data_dict["LowestPriceEver"] = data[20]
        data_dict["TotalTradedValue"] = data[21]
        marketDepthIndex = 0
        for i in range(22, len(data)):
            data_dict["Quantity-"+str(marketDepthIndex)] = data[i][0]
            data_dict["OrderPrice-"+str(marketDepthIndex)] = data[i][1]
            data_dict["TotalOrders-"+str(marketDepthIndex)] = data[i][2]
            data_dict["Reserved-"+str(marketDepthIndex)] = data[i][3]
            data_dict["SellQuantity-"+str(marketDepthIndex)] = data[i][4]
            data_dict["SellOrderPrice-"+str(marketDepthIndex)] = data[i][5]
            data_dict["SellTotalOrders-"+str(marketDepthIndex)] = data[i][6]
            data_dict["SellReserved-"+str(marketDepthIndex)] = data[i][7]
            marketDepthIndex += 1
    elif data_type == '1':
        data_dict = {
            "symbol": data[0],
            "open": data[1],
            "last": data[2],
            "high": data[3],
            "low": data[4],
            "change": data[5],
            "bPrice": data[6],
            "bQty": data[7],
            "sPrice": data[8],
            "sQty": data[9],
            "ltq": data[10],
            "avgPrice": data[11],
            "quotes": "Quotes Data"
        }
        # For NSE & BSE conversion
        if len(data) == 21:
            data_dict["ttq"] = data[12]
            data_dict["totalBuyQt"] = data[13]
            data_dict["totalSellQ"] = data[14]
            data_dict["ttv"] = data[15]
            data_dict["trend"] = data[16]
            data_dict["lowerCktLm"] = data[17]
            data_dict["upperCktLm"] = data[18]
            data_dict["ltt"] = datetime.fromtimestamp(
                data[19]).strftime('%c')
            data_dict["close"] = data[20]
        # For FONSE & CDNSE conversion
        elif len(data) == 23:
            data_dict["OI"] = data[12]
            data_dict["CHNGOI"] = data[13]
            data_dict["ttq"] = data[14]
            data_dict["totalBuyQt"] = data[15]
            data_dict["totalSellQ"] = data[16]
            data_dict["ttv"] = data[17]
            data_dict["trend"] = data[18]
            data_dict["lowerCktLm"] = data[19]
            data_dict["upperCktLm"] = data[20]
            data_dict["ltt"] = datetime.fromtimestamp(
                data[21]).strftime('%c')
            data_dict["close"] = data[22]
    else:
        data_dict = {
            "symbol": data[0],
            "time": datetime.fromtimestamp(data[1]).strftime('%c'),
            "depth": parse_market_depth(data[2], exchange),
            "quotes": "Market Depth"
        }
    if exchange == '4' and len(data) == 21:
        data_dict['exchange'] = 'NSE Equity'
    elif exchange == '1':
        data_dict['exchange'] = 'BSE'
    elif exchange == '13':
        data_dict['exchange'] = 'NSE Currency'
    elif exchange == '4' and len(data) == 23:
        data_dict['exchange'] = 'NSE Futures & Options'
    elif exchange == '6':
        data_dict['exchange'] = 'Commodity'
    return data_dict

# CallBack functions to receive feeds
def on_ticks(ticks):
    ticks = parse_data(ticks)
    print(ticks)

#Connect to receive feeds

sio.on('order', on_ticks)



#Disconnect from the server
```

```
sio.emit("disconnect", "transport close")
```

REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://livefeeds.icicidirect.com/
SourceNumber	Source Number
userId	User ID
messageLength	Message Length
requestType	Request Type
messageSequence	Message Sequence
messageDate	Message Date
messageTime	Message Time
messageCategory	Message Category
Intraday Calls	Intraday Calls
messagePriority Message Priority	
messageType	Message Type
orderMatchAccount	Order Match Account
orderExchangeCode	Order Exchange Code
stockCode	Stock Code
orderFlow	Order Flow
limitMarketFlag	Limit Market Flag
orderType	Order Type
orderLimitRate	Order Limit Rate
productType	Product Type
orderStatus	Order Status
orderDate	Order Date
orderTradeDate	Order Trade Date
orderReference	Order Reference
orderQuantity	Order Quantity
openQuantity	Open Quantity
orderExecutedQuantity	Order Executed Quantity
cancelledQuantity	Cancelled Quantity
expiredQuantity	Expired Quantity
orderDisclosedQuantity	Order Disclosed Quantity
orderStopLossTrigger	Order Stop Loss Trigger
orderSquareFlag	Order Square Flag
orderAmountBlocked	Order Amount Blocked
orderPipeId	Order PipeId
channel	Channel
exchangeSegmentCode	Exchange Segment Code
exchangeSegmentSettlement	Exchange Segment Settlement
segmentDescription	Segment Description
marginSquareOffMode	Margin Square Off Mode

Category	Value
orderMessageCharacter	Order Message Character
averageExecutedRate	Average Executed Rate
orderPriceImprovementFlag	Order Price Improvement Flag
orderMBCFlag	Order MBC Flag
orderLimitOffset	Order Limit Offset
systemPartnerCode	System Partner Code

Note : After every state change in order status, new notification would be received instantaneously.

Tick Data Stream

Tick By Tick Market Data

```
import base64
import socketio

#Get User ID and Session Token
session_key = "SESSION_TOKEN_FROM_CUSTOMER_DETAILS_API"
#e.g session_key = "QUYyOTUzMTM6NjY5ODc5NzY="

user_id, session_token = base64.b64decode(session_key.encode('ascii')).decode('ascii').split(":")
#e.g Decoded value - AF296713:66987976, after split user_id = AF295313, session_token = 6698797

# Python Socket IO Client
sio = socketio.Client()


script_code = "4.1!1594" #Subscribe more than one stock at a time
channel_name = 'stock'


auth = {"user": user_id, "token": session_token}

sio.connect("https://livestream.icicidirect.com", headers={"User-Agent":"python-socketio[client]/socket"},
            auth=auth, transports="websocket", wait_timeout=3)

tux_to_user_value = dict()

# parse market depth

def parse_market_depth(self, data, exchange):
    depth = []
    counter = 0
    for lis in data:
        counter += 1
        dict = {}
        if exchange == '1':
            dict["BestBuyRate-"+str(counter)] = lis[0]
            dict["BestBuyQty-"+str(counter)] = lis[1]
            dict["BestSellRate-"+str(counter)] = lis[2]
            dict["BestSellQty-"+str(counter)] = lis[3]
            depth.append(dict)
        else:
            dict["BestBuyRate-"+str(counter)] = lis[0]
            dict["BestBuyQty-"+str(counter)] = lis[1]
            dict["BuyNoOfOrders-"+str(counter)] = lis[2]
            dict["BuyFlag-"+str(counter)] = lis[3]
            dict["BestSellRate-"+str(counter)] = lis[4]
            dict["BestSellQty-"+str(counter)] = lis[5]
            dict["SellNoOfOrders-"+str(counter)] = lis[6]
            dict["SellFlag-"+str(counter)] = lis[7]
            depth.append(dict)
    return depth


# parsing logic
def parse_data(data):
    if data and type(data) == list and len(data) > 0 and type(data[0]) == str and "!" not in data[0]:
        order_dict = {}
        order_dict["sourceNumber"] = data[0]                        #Source Number
        order_dict["group"] = data[1]                               #Group
        order_dict["userId"] = data[2]                              #User_id
        order_dict["key"] = data[3]                                 #Key
        order_dict["messageLength"] = data[4]                       #Message Length
        order_dict["requestType"] = data[5]                         #Request Type
        order_dict["messageSequence"] = data[6]                     #Message Sequence
        order_dict["messageDate"] = data[7]                         #Date
        order_dict["messageTime"] = data[8]                         #Time
        order_dict["messageCategory"] = data[9]                     #Message Category
        order_dict["messagePriority"] = data[10]                    #Priority
        order_dict["messageType"] = data[11]                        #Message Type
        order_dict["orderMatchAccount"] = data[12]                  #Order Match Account
        order_dict["orderExchangeCode"] = data[13]                  #Exchange Code
```

```
    if data[11] == '4' or data[11] == '5':
        order_dict["stockCode"] = data[14]                      #Stock Code
        order_dict["orderFlow"] = tux_to_user_value['orderFlow'].get(str(data[15]).upper(),str(data[15]))                       # Order Flow
        order_dict["limitMarketFlag"] = tux_to_user_value['limitMarketFlag'].get(str(data[16]).upper(),str(data[16]))                 #Limit Market Flag
        order_dict["orderType"] = tux_to_user_value['orderType'].get(str(data[17]).upper(),str(data[17]))                       #OrderType
        order_dict["orderLimitRate"] = data[18]                 #Limit Rate
        order_dict["productType"] = tux_to_user_value['productType'].get(str(data[19]).upper(),str(data[19]))                    #Product Type
        order_dict["orderStatus"] = tux_to_user_value['orderStatus'].get(str(data[20]).upper(),str(data[20]))                    # Order Status
        order_dict["orderDate"] = data[21]                      #Order  Date
        order_dict["orderTradeDate"] = data[22]                 #Trade Date
        order_dict["orderReference"] = data[23]                 #Order Reference
        order_dict["orderQuantity"] = data[24]                  #Order Quantity
        order_dict["openQuantity"] = data[25]                   #Open Quantity
        order_dict["orderExecutedQuantity"] = data[26]          #Order Executed Quantity
        order_dict["cancelledQuantity"] = data[27]              #Cancelled Quantity
        order_dict["expiredQuantity"] = data[28]                #Expired Quantity
        order_dict["orderDisclosedQuantity"] = data[29]         # Order Disclosed Quantity
        order_dict["orderStopLossTrigger"] = data[30]           #Order Stop Loss Triger
        order_dict["orderSquareFlag"] = data[31]                #Order Square Flag
        order_dict["orderAmountBlocked"] = data[32]             # Order Amount Blocked
        order_dict["orderPipeId"] = data[33]                    #Order PipeId
        order_dict["channel"] = data[34]                        #Channel
        order_dict["exchangeSegmentCode"] = data[35]            #Exchange Segment Code
        order_dict["exchangeSegmentSettlement"] = data[36]      #Exchange Segment Settlement
        order_dict["segmentDescription"] = data[37]             #Segment Description
        order_dict["marginSquareOffMode"] = data[38]            #Margin Square Off Mode
        order_dict["orderValidDate"] = data[40]                 #Order Valid Date
        order_dict["orderMessageCharacter"] = data[41]          #Order Message Character
        order_dict["averageExecutedRate"] = data[42]            #Average Exited Rate
        order_dict["orderPriceImprovementFlag"] = data[43]      #Order Price Flag
        order_dict["orderMBCFlag"] = data[44]                   #Order MBC Flag
        order_dict["orderLimitOffset"] = data[45]               #Order Limit Offset
        order_dict["systemPartnerCode"] = data[46]              #System Partner Code
    elif data[11] == '6' or data[11] == '7':
        order_dict["stockCode"] = data[14]                     #stockCode
        order_dict["productType"] =  tux_to_user_value['productType'].get(str(data[15]).upper(),str(data[15]))                  #Product Type
        order_dict["optionType"] = tux_to_user_value['optionType'].get(str(data[16]).upper(),str(data[16]))                     #Option Type
        order_dict["exerciseType"] = data[17]                   #Exercise Type
        order_dict["strikePrice"] = data[18]                    #Strike Price
        order_dict["expiryDate"] = data[19]                     #Expiry Date
        order_dict["orderValidDate"] = data[20]                 #Order Valid Date
        order_dict["orderFlow"] = tux_to_user_value['orderFlow'].get(str(data[21]).upper(),str(data[21]))                       #Order  Flow
        order_dict["limitMarketFlag"] = tux_to_user_value['limitMarketFlag'].get(str(data[22]).upper(),str(data[22]))                 #Limit Market Flag
        order_dict["orderType"] = tux_to_user_value['orderType'].get(str(data[23]).upper(),str(data[23]))                       #Order Type
        order_dict["limitRate"] = data[24]                      #Limit Rate
        order_dict["orderStatus"] = tux_to_user_value['orderStatus'].get(str(data[25]).upper(),str(data[25]))                    #Order Status
        order_dict["orderReference"] = data[26]                 #Order Reference
        order_dict["orderTotalQuantity"] = data[27]             #Order Total Quantity
        order_dict["executedQuantity"] = data[28]               #Executed Quantity
        order_dict["cancelledQuantity"] = data[29]              #Cancelled Quantity
        order_dict["expiredQuantity"] = data[30]                #Expired Quantity
        order_dict["stopLossTrigger"] = data[31]                #Stop Loss Trigger
        order_dict["specialFlag"] = data[32]                    #Special Flag
        order_dict["pipeId"] = data[33]                         #PipeId
        order_dict["channel"] = data[34]                        #Channel
        order_dict["modificationOrCancelFlag"] = data[35]       #Modification or Cancel Flag
        order_dict["tradeDate"] = data[36]                      #Trade Date
        order_dict["acknowledgeNumber"] = data[37]              #Acknowledgement Number
        order_dict["stopLossOrderReference"] = data[37]         #Stop Loss Order Reference
        order_dict["totalAmountBlocked"] = data[38]             # Total Amount Blocked
        order_dict["averageExecutedRate"] = data[39]            #Average Executed Rate
        order_dict["cancelFlag"] = data[40]                     #Cancel Flag
        order_dict["squareOffMarket"] = data[41]                #SquareOff Market
        order_dict["quickExitFlag"] = data[42]                  #Quick Exit Flag
        order_dict["stopValidTillDateFlag"] = data[43]          #Stop Valid till Date Flag
        order_dict["priceImprovementFlag"] = data[44]           #Price Improvement Flag
        order_dict["conversionImprovementFlag"] = data[45]      #Conversion Improvement Flag
        order_dict["trailUpdateCondition"] = data[45]           #Trail Update Condition
        order_dict["systemPartnerCode"] = data[46]              #System Partner Code
    return order_dict
exchange = str.split(data[0], '!')[0].split('.')[0]
data_type = str.split(data[0], '!')[0].split('.')[1]
if exchange == '6':
    data_dict = {}
    data_dict["symbol"] = data[0]
    data_dict["AndiOPVolume"] = data[1]
    data_dict["Reserved"] = data[2]
    data_dict["IndexFlag"] = data[3]
    data_dict["ttq"] = data[4]
    data_dict["last"] = data[5]
    data_dict["ltq"] = data[6]
    data_dict["ltt"] = datetime.fromtimestamp(data[7]).strftime('%c')
    data_dict["AvgTradedPrice"] = data[8]
    data_dict["TotalBuyQnt"] = data[9]
    data_dict["TotalSellQnt"] = data[10]
    data_dict["ReservedStr"] = data[11]
    data_dict["ClosePrice"] = data[12]
    data_dict["OpenPrice"] = data[13]
    data_dict["HighPrice"] = data[14]
    data_dict["LowPrice"] = data[15]
    data_dict["ReservedShort"] = data[16]
    data_dict["CurrOpenInterest"] = data[17]
    data_dict["TotalTrades"] = data[18]
    data_dict["HightestPriceEver"] = data[19]
    data_dict["LowestPriceEver"] = data[20]
    data_dict["TotalTradedValue"] = data[21]
    marketDepthIndex = 0
    for i in range(22, len(data)):
        data_dict["Quantity-"+str(marketDepthIndex)] = data[i][0]
        data_dict["OrderPrice-"+str(marketDepthIndex)] = data[i][1]
        data_dict["TotalOrders-"+str(marketDepthIndex)] = data[i][2]
        data_dict["Reserved-"+str(marketDepthIndex)] = data[i][3]
```

```
            data_dict["SellQuantity-"+str(marketDepthIndex)] = data[i][4]
            data_dict["SellOrderPrice-"+str(marketDepthIndex)] = data[i][5]
            data_dict["SellTotalOrders-"+str(marketDepthIndex)] = data[i][6]
            data_dict["SellReserved-"+str(marketDepthIndex)] = data[i][7]
            marketDepthIndex += 1
    elif data_type == '1':
        data_dict = {
            "symbol": data[0],
            "open": data[1],
            "last": data[2],
            "high": data[3],
            "low": data[4],
            "change": data[5],
            "bPrice": data[6],
            "bQty": data[7],
            "sPrice": data[8],
            "sQty": data[9],
            "ltq": data[10],
            "avgPrice": data[11],
            "quotes": "Quotes Data"
        }
        # For NSE & BSE conversion
        if len(data) == 21:
            data_dict["ttq"] = data[12]
            data_dict["totalBuyQt"] = data[13]
            data_dict["totalSellQ"] = data[14]
            data_dict["ttv"] = data[15]
            data_dict["trend"] = data[16]
            data_dict["lowerCktLm"] = data[17]
            data_dict["upperCktLm"] = data[18]
            data_dict["ltt"] = datetime.fromtimestamp(
                data[19]).strftime('%c')
            data_dict["close"] = data[20]
        # For FONSE & CDNSE conversion
        elif len(data) == 23:
            data_dict["OI"] = data[12]
            data_dict["CHNGOI"] = data[13]
            data_dict["ttq"] = data[14]
            data_dict["totalBuyQt"] = data[15]
            data_dict["totalSellQ"] = data[16]
            data_dict["ttv"] = data[17]
            data_dict["trend"] = data[18]
            data_dict["lowerCktLm"] = data[19]
            data_dict["upperCktLm"] = data[20]
            data_dict["ltt"] = datetime.fromtimestamp(
                data[21]).strftime('%c')
            data_dict["close"] = data[22]
    else:
        data_dict = {
            "symbol": data[0],
            "time": datetime.fromtimestamp(data[1]).strftime('%c'),
            "depth": parse_market_depth(data[2], exchange),
            "quotes": "Market Depth"
        }
    if exchange == '4' and len(data) == 21:
        data_dict['exchange'] = 'NSE Equity'
    elif exchange == '1':
        data_dict['exchange'] = 'BSE'
    elif exchange == '13':
        data_dict['exchange'] = 'NSE Currency'
    elif exchange == '4' and len(data) == 23:
        data_dict['exchange'] = 'NSE Futures & Options'
    elif exchange == '6':
        data_dict['exchange'] = 'Commodity'
    return data_dict

# CallBack functions to receive feeds
def on_ticks(ticks):
    ticks = parse_data(ticks)
    print(ticks)

sio.emit('join', script_code)
sio.on(channel_name, on_ticks)

#Unwatch from the stock
sio.emit("leave", script_code)


#Disconnect from the server
sio.emit("disconnect", "transport close")
```

REQUEST INFORMATION

The WebSocket API uses WebSocket protocol to establish a single long standing TCP connection after an HTTP handshake to receive streaming quotes.To connect to the ICICI's WebSocket API, you will need a WebSocket client library in your choice of programming language we will provide the sample code supported in 4 languages(Python, C#, Javascript and Java). You can subscribe for stock-token on a single WebSocket connection and receive live stream for them.

Category	Value
HTTP Request	GET
URL	https://livestream.icicidirect.com/
Symbol Stock	Token Value
Open	Open Price

Category	Value
Last	Last Price
high	High Price
Low	Low Price
change	change
bPrice	Buy Price
bQty	Buy Quantity
sPrice	Selling Price
sQty	Selling Quantity
ltq	Last Traded Quantity
avgPrice	Average Price
quotes	Quotes
ttq	Total Traded Quantity
totalBuyQt	Total Buy Quantity
totalSellQt	Total Sell Quantity
ttv	Total Traded Volume
trend	trend
lowerCktLM Lower	Circuit Limit
upperCktLM	Upper Circuit Limit
ltt	Last Traded Time
close Close	Price
exchange	Exchange
stock_name	Stock Name

One Click F&O Stream

Oneclick Strategy

```
#import required libraries
import base64
import socketio

#Get User ID and Session Token
session_key = "SESSION_TOKEN_FROM_CUSTOMER_DETAILS_API"
#e.g session_key = "QUYyOTUzMTM6NjY5ODc5NzY="

user_id, session_token = base64.b64decode(session_key.encode('ascii')).decode('ascii').split(":")
#e.g Decoded value - AF296713:66987976, after split user_id = AF295313, session_token = 6698797

# Python Socket IO Client
sio = socketio.Client()
auth = {"user": user_id, "token": session_token}
sio.connect("https://livefeeds.icicidirect.com", headers={"User-Agent":"python-socketio[client]/socket"},
                auth=auth, transports="websocket", wait_timeout=3)

# Script Code of Stock or Instrument  e.g 4.1!1594, 1.1!500209 , 13.1!5023, 6.1!247457.
script_code = ["one_click_fno"] #Subscribe more than one stock at a time

channel_name = 'stock'

#parsing logic
def parse_data(data):

    if data and type(data) == list and len(data) > 0 and type(data[0]) == str and "!" not in data[0] and len(data) == 28:
        strategy_dict = dict()
        strategy_dict['strategy_date'] = data[0]
        strategy_dict['modification_date'] = data[1]
        strategy_dict['portfolio_id'] = data[2]
        strategy_dict['call_action'] = data[3]
        strategy_dict['portfolio_name'] = data[4]
        strategy_dict['exchange_code'] = data[5]
        strategy_dict['product_type'] = data[6]
        #strategy_dict['INDEX/STOCK'] = data[7]
```

```
        strategy_dict['underlying'] = data[8]
        strategy_dict['expiry_date'] = data[9]
        #strategy_dict['OCR_EXER_TYP'] = data[10]
        strategy_dict['option_type'] = data[11]
        strategy_dict['strike_price'] = data[12]
        strategy_dict['action'] = data[13]
        strategy_dict['recommended_price_from'] = data[14]
        strategy_dict['recommended_price_to'] = data[15]
        strategy_dict['minimum_lot_quantity'] = data[16]
        strategy_dict['last_traded_price'] = data[17]
        strategy_dict['best_bid_price'] = data[18]
        strategy_dict['best_offer_price'] = data[19]
        strategy_dict['last_traded_quantity'] = data[20]
        strategy_dict['target_price'] = data[21]
        strategy_dict['expected_profit_per_lot'] = data[22]
        strategy_dict['stop_loss_price'] = data[23]
        strategy_dict['expected_loss_per_lot'] = data[24]
        strategy_dict['total_margin'] = data[25]
        strategy_dict['leg_no'] = data[26]
        strategy_dict['status'] = data[27]
        return(strategy_dict)

#CallBack functions to receive feeds
def on_ticks(ticks):
    ticks = parse_data(ticks)
    print(ticks)

#Connect to receive feeds
sio.emit('join', script_code)
sio.on(channel_name, on_ticks)

#Unwatch from the stock
sio.emit("leave", script_code)

#Disconnect from the server
sio.emit("disconnect", "transport close")
```

Sample JSON Response


```
{
    'strategy_date': '2023-02-21 10:35:20',
    'modification_date': '2023-02-21 10:35:20',
    'portfolio_id': '32587',
    'call_action': 'Call Initiated',
    'portfolio_name': 'Long Future',
    'exchange_code': 'NFO',
    'product_type': 'futures',
    'underlying': 'POWGRI',
    'expiry_date': '2023-02-23 00:00:00',
    'option_type': 'others',
    'strike_price': '0',
    'action': 'buy',
    'recommended_price_from': '217',
    'recommended_price_to': '217.2',
    'minimum_lot_quantity': '2700',
    'last_traded_price': '217.35',
    'best_bid_price': '217.2',
    'best_offer_price': '217.35',
    'last_traded_quantity': '217.25',
    'target_price': '220',
    'expected_profit_per_lot': '7830',
    'stop_loss_price': '215.95',
    'expected_loss_per_lot': '3105',
    'total_margin': '109354.72',
    'leg_no': '1',
    'status': 'active'
}
```


The One Click F&O is a functionality which lets you chose from range of strategies curated by our award winning research team. It helps you save hours and hours of research and analysis time required to make a trading decision. With this function of One Click F&O, our research team does all the heavy lifting of research and analysis of over 170 F&O stocks with different expiries in real time and develop high probability winning opportunities for our customers. For the strategies available, you can either chose the desired strategy and place order in a single click or opt for customising by copying our recommended strategy to suit your requirement. You can also view margin required, maximum profit/loss, timeframe to hold the strategy before making a trade.

GET USER ID AND SESSION TOKEN

The User Id and session Token can be obtained by decoding the 'session_token' from base64 , This session token obtained via get_customer_details API. An example is given in python

CONNECT TO WEBSOCKET SERVER

Websocket endpoint is https://livefeeds.icicidirect.com/. Please make sure to pass the auth credentials and User-Agent in headers

REQUEST INFORMATION

Category	Value
HTTP Request	`GET`
URL	https://livefeeds.icicidirect.com/

One Click Equity Stream

Oneclick Equity Strategy(iclick_2_gain)

```
#import required libraries
import base64
import socketio

#Get User ID and Session Token
session_key = "SESSION_TOKEN_FROM_CUSTOMER_DETAILS_API"
#e.g session_key = "QUYyOTUzMTM6NjY5ODc5NzY="

user_id, session_token = base64.b64decode(session_key.encode('ascii')).decode('ascii').split(":")
#e.g Decoded value - AF296713:66987976, after split user_id = AF295313, session_token = 6698797

# Python Socket IO Client
sio = socketio.Client()
auth = {"user": user_id, "token": session_token}
sio.connect("https://livefeeds.icicidirect.com", headers={"User-Agent":"python-socketio[client]/socket"},
                auth=auth, transports="websocket", wait_timeout=3)

# Script Code of Stock or Instrument  e.g 4.1!1594, 1.1!500209 , 13.1!5023, 6.1!247457.
script_code = ["i_click_2_gain"] #Subscribe more than one stock at a time

channel_name = 'stock'

#parsing logic
def parse_data(data):
    if data and type(data) == list and len(data) > 0 and type(data[0]) == str and "!" not in data[0] and len(data) == 19:
        iclick_data = dict()
        #iclick_data['sequence_number'] = data[0]
        iclick_data['stock_name'] = data[0]
        iclick_data['stock_code'] = data[1]
        iclick_data['action_type'] = data[2]
        iclick_data['expiry_date'] = data[3]
        iclick_data['strike_price'] = data[4]
        iclick_data['option_type'] = data[5]
        iclick_data['stock_description'] = data[6]
        iclick_data['recommended_price_and_date'] = data[7]
        iclick_data['recommended_price_from'] = data[8]
        iclick_data['recommended_price_to'] = data[9]
        iclick_data['recommended_date'] = data[10]
        iclick_data['target_price'] = data[11]
        iclick_data['sltp_price'] = data[12]
        iclick_data['part_profit_percentage'] = data[13]
        iclick_data['profit_price'] = data[14]
        iclick_data['exit_price'] = data[15]
        iclick_data['recommended_update'] = data[16]
        iclick_data['iclick_status'] = data[17]
        iclick_data['subscription_type'] = data[18]
        return(iclick_data)



#CallBack functions to receive feeds
def on_ticks(ticks):
    ticks = parse_data(ticks)
    print(ticks)

#Connect to receive feeds
sio.emit('join', script_code)
sio.on(channel_name, on_ticks)

#Unwatch from the stock
sio.emit("leave", script_code)

#Disconnect from the server
sio.emit("disconnect", "transport close")
```

Sample JSON Response


```
{
    'strategy_date': '2023-02-21 10:35:20',
    'modification_date': '2023-02-21 10:35:20',
    'portfolio_id': '32587',
    'call_action': 'Call Initiated',
    'portfolio_name': 'Long Future',
    'exchange_code': 'NFO',
    'product_type': 'futures',
    'underlying': 'POWGRI',
    'expiry_date': '2023-02-23 00:00:00',
    'option_type': 'others',
    'strike_price': '0',
    'action': 'buy',
    'recommended_price_from': '217',
    'recommended_price_to': '217.2',
    'minimum_lot_quantity': '2700',
    'last_traded_price': '217.35',
    'best_bid_price': '217.2',
    'best_offer_price': '217.35',
    'last_traded_quantity': '217.25',
    'target_price': '220',
```

```
    'expected_profit_per_lot': '7830',
    'stop_loss_price': '215.95',
    'expected_loss_per_lot': '3105',
    'total_margin': '109354.72',
    'leg_no': '1',
    'status': 'active'
}
```

REQUEST INFORMATION

Category	Value
HTTP Request	GET
URL	https://livefeeds.icicidirect.com/

The One Click Equity Stream also known as iclick2gain stream is a functionality which lets you chose from range of strategies curated by our award winning research team. It helps you save hours and hours of research and analysis time required to make a trading decision. With this function of One Click Equity, our research team does all the heavy lifting of research and analysis of over Equity stocks with different expiries in real time and develop high probability winning opportunities for our customers.

For the strategies available, you can either chose the desired strategy and place order in a single click or opt for customising by copying our recommended strategy to suit your requirement. You can also view margin required, maximum profit/loss, timeframe to hold the strategy before making a trade.

Candle Stream

Candle Data LIVE(StreamLiveOHLCV)

This feature allows the user to stream OHLCV candle data in real time via websockets with lower latency. Candle intervals supported are 1 second, 1 minute, 5 minutes and 30 minutes. Below, is the set instructions to use this feature.

```
import base64
import socketio

#Get User ID and Session Token
session_key = "SESSION_TOKEN_FROM_CUSTOMER_DETAILS_API"
#e.g session_key = "QUYyOTUzMTM6NjY5ODc5NzY="

user_id, session_token = base64.b64decode(session_key.encode('ascii')).decode('ascii').split(":")
#e.g Decoded value - AF296713:66987976, after split user_id = AF295313, session_token = 6698797

# Python Socket IO Client
sio = socketio.Client()
auth = {"user": user_id, "token": session_token}
sio.connect("https://breezeapi.icicidirect.com/", socketio_path='ohlcvstream', headers={"User-Agent":"python-socketio[client]/socket"},
                auth=auth, transports="websocket", wait_timeout=3)

# Script Code of Stock or Instrument  e.g 4.1!1594, 1.1!500209 , 13.1!5023, 6.1!247457.
script_code = ["4.1!1594"] #Subscribe more than one stock at a time

#Channel name i.e 1SEC,1MIN,5MIN,30MIN
channel_name = "1SEC"

#CallBack functions to receive feeds
def on_ticks(ticks):
       print(ticks)

#Connect to receive feeds
sio.emit('join', script_code)
sio.on(channel_name, on_ticks)

#Unwatch from the stock
sio.emit("leave", script_code)

#Disconnect from the server
sio.emit("disconnect", "transport close")
```

Sample Response Examples

```
For Equity -



NSE,NIFTY,18687.95,18687.95,18687.95,18687.95,0,2022-12-02 14:13:53,1SEC



For Derivatives -



Options - NFO,NIFTY,08-Dec-2022,18700.0,CE,120.5,120.5,120.5,120.5,2500,7592550,2022-12-02 14:10:14,1SEC
```

```
Futures - NFO,NIFTY,29-Dec-2022,18807.35,18807.35,18807.35,18807.35,0,11771450,2022-12-02 14:19:21,1SEC
```

GET USER ID AND SESSION TOKEN

The User Id and session Token can be obtained by decoding the 'session_token' from base64 , This session token obtained via get_customer_details API. An example is given in python

CONNECT TO WEBSOCKET SERVER

Websocket endpoint is **https://breezeapi.icicidirect.com/** and the socket.io path is **'/ohlcvstream'**. Please make sure to pass the auth credentials and User-Agent in headers

SCRIPT CODE OF A STOCK

Real streaming ohlcv data of a stock is obtained by subscribing to a particular stock using the Script Code of the stock. Script Code of a stock has the following pattern as given below.

'ExchangeType.Qualiflier!Scripid'

Examples of script code are 4.1!1594, 1.1!500209 , 13.1!5023, 6.1!247457

The Qualifier of an ExchangeType corresponds to a particular Exchange Code. Please refer to the table below

Qualifier	Exchange Code
4.1	NSE,NFO
1.1	BSE
13.1	NDX
6.1	MCX

Note: Suffix '.1' in Qualifier refers to Exchange Quotes whereas '.2' refers to Market Depth. In Streaming OHLCV, we just need Exchange Quotes

For ScriptId, please refer to the CSV attachment in the link below

https://traderweb.icicidirect.com/Content/File/txtFile/ScripFile/StockScriptNew.csv

Column **'TK'** in the CSV File refers to ScriptId.

SUBSCRIBE TO OHLCV LIVE STREAM OF A PARTICULAR STOCK

To subscribe to a particular stock, we require to have the Script Code of a particular stocks. We can subscribe to more than one stock for a particular interval.

To subscribe to stock, we need to connect to a channels. Channels are divided as per the candle intervals. The channel names are as follows

Channel Name	Interval
1SEC	1 second
1MIN	1 minute
5MIN	5 minutes
30MIN	30 minutes

STREAMING RESPONSE DATA CONVENTION

The streaming respone Data has the following structure as shown in tables below. The response is a comma separated string.

For equity i.e NSE,BSE

Attribute	Description
exchange code	Exchange Code (NSE,BSE)
stock code	Stock Code
low	Low Price
high	High Price
open	Open Price
close	Close Price
volume	Traded Volume
datetime	Date in YYYY-MM-DD HH:MM:SS format
interval	1SEC,1MIN,5MIN,30MIN

For derivative options/futues i.e NFO,NDX,MCX

Attribute	Description
exchange code	Exchange Code (NFO,NDX,MCX)
stock code	Stock Code

Attribute	Description
expiry date	Date in YYYY-MM-DD format
strike price	Strike Price (Only For Options)
right type	Right Type (Only For Options)
low	Low Price
high	High Price
open	Open Price
close	Close Price
volume	Traded Volume
oi	Open Interest
datetime	Date in YYYY-MM-DD HH:MM:SS format
interval	1SEC,1MIN,5MIN,30MIN

Note: interval - 1SEC,1MIN,5MIN,30MIN (for 1second, 1minute, 5minutes, 30minutes respectively)