Introduction

Kite Connect is a set of REST-like HTTP APIs that expose many capabilities required to build a complete stock market investment and trading platform. It lets you execute orders in real time (equities, commodities, mutual funds), manage user portfolios, stream live market data over WebSockets, and more.

All inputs are form-encoded parameters and responses are JSON (apart from a couple exceptions). The responses may be Gzipped. Standard HTTP codes are used to indicate success and error states with accompanying JSON data. The API endpoints are *not* cross site request enabled, hence cannot be called directly from browsers.

An `api_key` + `api_secret` pair is issued and you have to register a redirect url where a user is sent after the login flow. For mobile and desktop applications, there has to be a remote backend which does the handshake on behalf of the mobile app and the `api_secret` should never be embedded in the app.

Note

If you don't have a Kite Connect developer account, read more about it and signup here.

Libraries and SDKs

Below is a list of pre-built client libraries for Kite Connect written in various programming languages that can be used used to interact with the APIs without having to make raw HTTP calls.

- Kite Connect Python library (*Official*)
- Kite Connect Go library (*Official*)
- Kite Connect Java library (*Official*)
- Kite Connect PHP library (*Official*)
- Kite Connect Node JS library (*Official*)
- Kite Connect .Net/C# library (*Official*)
- Kite Connect Rust (*Official*)
- Kite Connect R (*3rd party*)

Version and API endpoint

The current major stable version of the API is **3**. All requests go to it by default. It is recommended that a specific version be requested explicitly for production applications as major releases may break older implementations.

> **Note**
>
> This version is a KiteConnect backend API version and should not be confused with the specific library release version.

Root API endpoint

`https://api.kite.trade`

Requesting a particular version

To request a particular version of the API, set the HTTP header `X-Kite-version: v` where `v` is the version number, major or full (eg: 1 or 1.3 or 3)

Response structure

All GET and DELETE request parameters go as query parameters, and POST and PUT parameters as form-encoded (`application/x-www-form-urlencoded`) parameters, responses from the API are always JSON.

Successful request

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status": "success",
    "data": {}
}
```

All responses from the API server are JSON with the content-type `application/json` unless explicitly stated otherwise. A successful `200 OK` response always has a JSON response body with a `status` key with the value `success` . The `data` key contains the full response payload.

Failed request

```
HTTP/1.1 500 Server error
Content-Type: application/json

{
    "status": "error",
    "message": "Error message",
    "error_type": "GeneralException"
}
```

A failure response is preceded by the corresponding `40x` or `50x` HTTP header. The `status` key in the response envelope contains the value `error` . The `message` key contains a textual description of the error and `error_type` contains the name of the exception. There may be an optional `data` key with additional payload.

Data types

Values in JSON responses are of types string, int, float, or bool.

Timestamp (datetime) strings in the responses are represented in the form `yyyy-mm-dd hh:mm:ss`, set under the Indian timezone (IST) — UTC+5.5 hours.

A date string is represented in the form `yyyy-mm-dd`.

Exceptions and errors

In addition to the `40x` and `50x` headers, error responses come with the name of the exception generated internally by the API server. You can define corresponding exceptions in your language or library, and raise them by doing a switch on the returned exception name.

Example

```
HTTP/1.1 500 Server error
Content-Type: application/json

{
    "status": "error",
    "message": "Error message",
    "error_type": "GeneralException"
}
```

exception	
TokenException	Preceded by a `403` header, this indicates the expiry or invalidation of an authenticated session. This can be caused by the user logging out, a natural expiry, or the user logging into another Kite instance. When you encounter this error, you should clear the user's session and re-initiate a login.
UserException	Represents user account related errors
OrderException	Represents order related errors such placement failures, a corrupt fetch etc
InputException	Represents missing required fields, bad values for parameters etc.
MarginException	Represents insufficient funds, required for the order placement
HoldingException	Represents insufficient holdings, available to place sell order for specified instrument

exception	
NetworkException	Represents a network error where the API was unable to communicate with the OMS (Order Management System)
DataException	Represents an internal system error where the API was unable to understand the response from the OMS to inturn respond to a request
GeneralException	Represents an unclassified error. This should only happen rarely

Common HTTP error codes

code	
`400`	Missing or bad request parameters or values
`403`	Session expired or invalidate. Must relogin
`404`	Request resource was not found
`405`	Request method (GET, POST etc.) is not allowed on the requested endpoint
`410`	The requested resource is gone permanently
`429`	Too many requests to the API (rate limiting)
`500`	Something unexpected went wrong
`502`	The backend OMS is down and the API is unable to communicate with it
`503`	Service unavailable; the API is down
`504`	Gateway timeout; the API is unreachable

API rate limit

end-point	rate-limit
Quote	1req/second
Historical candle	3req/second
Order placement	10req/second
All other endpoints	10req/second

Note

There are limitations at 200 orders per minute and 10 orders per second.
As a risk management measure, at Zerodha, a single user/API key will not be able to place more than 3000 orders per day. This restriction is across all segments and varieties.
Rate limitations also apply for order modification where a maximum of 25 modifications are allowed per order. Post that user has to cancel the order and place it again.

User

Login flow

The login flow starts by navigating to the public Kite login endpoint.

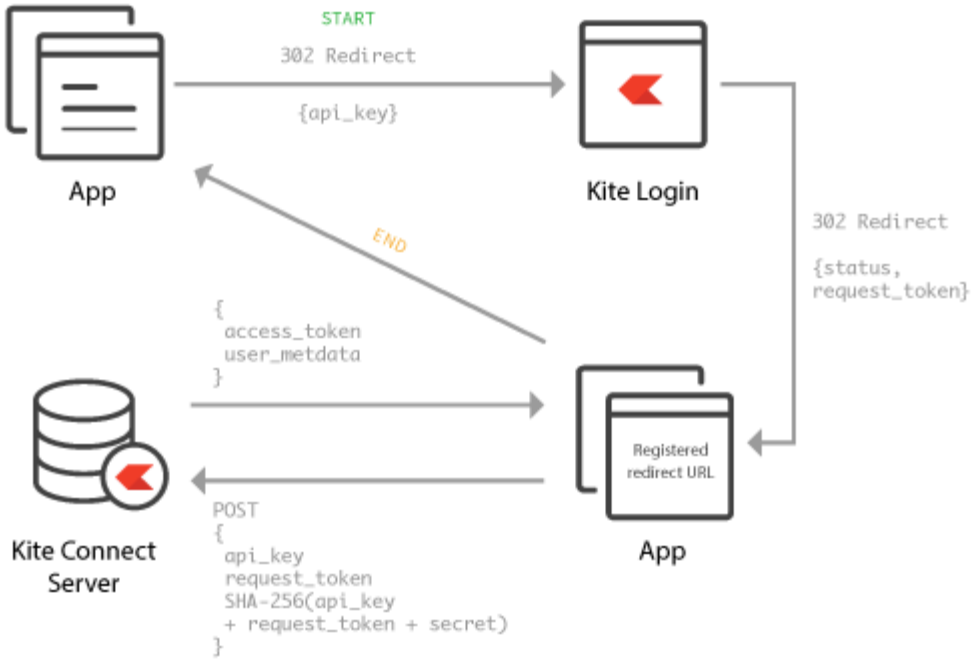
> `https://kite.zerodha.com/connect/login?v=3&api_key=xxx`

A successful login comes back with a `request_token` as a URL query parameter to the redirect URL registered on the developer console for that `api_key`. This `request_token`, along with a `checksum` (SHA-256 of `api_key + request_token + api_secret`) is POSTed to the token API to obtain an `access_token`, which is then used for signing all subsequent requests. In summary:

1. Navigate to the Kite Connect login page with the api_key
2. A successful login comes back with a `request_token` to the registered redirect URL
3. POST the `request_token` and `checksum` (SHA-256 of `api_key + request_token + api_secret`) to `/session/token`
4. Obtain the `access_token` and use that with all subsequent requests

> An optional `redirect_params` param can be appended to public Kite login endpoint, that will be sent back to the redirect URL. The value is URL encoded query params string, eg: `some=X&more=Y`). eg: `https://kite.zerodha.com/connect/login?v=3&api_key=xxx&redirect_params=some%3DX%26more%3DY`

Here's a webinar that shows the login flow and other interactions.



Warning

Never expose your `api_secret` by embedding it in a mobile app or a client side application. Do not expose the `access_token` you obtain for a session to the public either.

type	endpoint	
POST	/session/token	Authenticate and obtain the access_token after the login flow
GET	/user/profile	Retrieve the user profile
GET	/user/margins/:segment	Retrieve detailed funds and margin information

type	endpoint	
DELETE	/session/token	Logout and invalidate the API session and access_token

Authentication and token exchange

Once the `request_token` is obtained from the login flow, it should be POSTed to `/session/token` to complete the token exchange and retrieve the `access_token`.

```
curl https://api.kite.trade/session/token \
   -H "X-Kite-Version: 3" \
   -d "api_key=xxx" \
   -d "request_token=yyy" \
   -d "checksum=zzz"
```


```
{
    "status": "success",
    "data": {
        "user_type": "individual",
        "email": "XXXXXX",
        "user_name": "Kite Connect",
        "user_shortname": "Connect",
        "broker": "ZERODHA",
        "exchanges": [
            "NSE",
            "NFO",
            "BFO",
            "CDS",
            "BSE",
            "MCX",
            "BCD",
            "MF"
        ],
        "products": [
            "CNC",
            "NRML",
            "MIS",
            "BO",
            "CO"
        ],
        "order_types": [
            "MARKET",
            "LIMIT",
            "SL",
```

```
            "SL-M"
        ],
        "avatar_url": "abc",
        "user_id": "XX0000",
        "api_key": "XXXXXX",
        "access_token": "XXXXXX",
        "public_token": "XXXXXXXX",
        "enctoken": "XXXXXX",
        "refresh_token": '',
        "silo": '',
        "login_time": "2021-01-01 16:15:14",
        "meta": {
            "demat_consent": "physical"
        }
    }
}
```

Request parameters

parameter	
`api_key`	The public API key
`request_token`	The one-time token obtained after the login flow. This token's lifetime is only a few minutes and it is meant to be exchanged for an `access_token` immediately after being obtained
`checksum`	SHA-256 hash of (api_key + request_token + api_secret)

Response attributes

attribute	
`user_id` string	The unique, permanent user id registered with the broker and the exchanges
`user_name` string	User's real name
`user_shortname` string	Shortened version of the user's real name

attribute	
`email` string	User's email
`user_type` string	User's registered role at the broker. This will be `individual` for all retail users
`broker` string	The broker ID
`exchanges` string[]	Exchanges enabled for trading on the user's account
`products` string[]	Margin product types enabled for the user
`order_types` string[]	Order types enabled for the user
`api_key` string	The API key for which the authentication was performed
`access_token` string	The authentication token that's used with every subsequent request Unless this is invalidated using the API, or invalidated by a master-logout from the Kite Web trading terminal, it'll expire at `6 AM` on the next day (regulatory requirement)
`public_token` string	A token for public session validation where requests may be exposed to the public
`refresh_token` string	A token for getting long standing read permissions. This is only available to certain approved platforms
`login_time` string	User's last login time
`meta` map	`demat_consent`: empty, consent or physical

attribute	
`avatar_url` string	Full URL to the user's avatar (PNG image) if there's one

Signing requests

Once the authentication is complete, all requests should be signed with the HTTP `Authorization` header with `token` as the authorisation scheme, followed by a space, and then the `api_key:access_token` combination. For example:

```
curl -H "Authorization: token api_key:access_token"
curl -H "Authorization: token xxx:yyy"
```

User profile

While a successful token exchange returns the full user profile, it's possible to retrieve it any point of time with the `/user/profile` API. Do note that the profile API does not return any of the tokens.

```
curl https://api.kite.trade/user/profile \
   -H "X-Kite-Version: 3" \
   -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": {
    "user_id": "AB1234",
    "user_type": "individual",
    "email": "xxxyyy@gmail.com",
    "user_name": "AxAx Bxx",
    "user_shortname": "AxAx",
    "broker": "ZERODHA",
    "exchanges": [
      "BFO",
      "MCX",
      "NSE",
      "CDS",
      "BSE",
      "BCD",
      "MF",
      "NFO"
    ],
```

```
    "products": [
      "CNC",
      "NRML",
      "MIS",
      "BO",
      "CO"
    ],
    "order_types": [
      "MARKET",
      "LIMIT",
      "SL",
      "SL-M"
    ],
    "avatar_url": null,
    "meta": {
      "demat_consent": "physical"
    }
  }
}
```

Response attributes

attribute	
`user_id` string	The unique, permanent user id registered with the broker and the exchanges
`user_name` string	User's real name
`user_shortname` string	Shortened version of the user's real name
`email` string	User's email
`user_type` string	User's registered role at the broker. This will be `individual` for all retail users
`broker` string	The broker ID

attribute	
`exchanges` string[]	Exchanges enabled for trading on the user's account
`products` string[]	Margin product types enabled for the user
`order_types` string[]	Order types enabled for the user
`meta` map	`demat_consent` : empty, consent or physical
`avatar_url` string	Full URL to the user's avatar (PNG image) if there's one

Funds and margins

A GET request to `/user/margins` returns funds, cash, and margin information for the user for equity and commodity segments.

A GET request to `/user/margins/:segment` returns funds, cash, and margin information for the user. `segment` in the URI can be either `equity` or `commodity`.

```
curl "https://api.kite.trade/user/margins" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```


```
{
    "status": "success",
    "data": {
      "equity": {
        "enabled": true,
        "net": 99725.05000000002,
        "available": {
          "adhoc_margin": 0,
          "cash": 245431.6,
          "opening_balance": 245431.6,
          "live_balance": 99725.05000000002,
          "collateral": 0,
```

```
        "intraday_payin": 0
      },
      "utilised": {
        "debits": 145706.55,
        "exposure": 38981.25,
        "m2m_realised": 761.7,
        "m2m_unrealised": 0,
        "option_premium": 0,
        "payout": 0,
        "span": 101989,
        "holding_sales": 0,
        "turnover": 0,
        "liquid_collateral": 0,
        "stock_collateral": 0,
        "delivery": 0
      }
    },
    "commodity": {
      "enabled": true,
      "net": 100661.7,
      "available": {
        "adhoc_margin": 0,
        "cash": 100661.7,
        "opening_balance": 100661.7,
        "live_balance": 100661.7,
        "collateral": 0,
        "intraday_payin": 0
      },
      "utilised": {
        "debits": 0,
        "exposure": 0,
        "m2m_realised": 0,
        "m2m_unrealised": 0,
        "option_premium": 0,
        "payout": 0,
        "span": 0,
        "holding_sales": 0,
        "turnover": 0,
        "liquid_collateral": 0,
        "stock_collateral": 0,
        "delivery": 0
      }
    }
  }
}
```


Response attributes

attribute	
`enabled` bool	Indicates whether the segment is enabled for the user
`net` float64	Net cash balance available for trading (`intraday_payin` + `adhoc_margin` + `collateral`)
`available.cash` float64	Raw cash balance in the account available for trading (also includes `intraday_payin`)
`available.opening_balance` float64	Opening balance at the day start
`available.live_balance` float64	Current available balance
`available.intraday_payin` float64	Amount that was deposited during the day
`available.adhoc_margin` float64	Additional margin provided by the broker
`available.collateral` float64	Margin derived from pledged stocks
`utilised.m2m_unrealised` float64	Un-booked (open) intraday profits and losses
`utilised.m2m_realised` float64	Booked intraday profits and losses
`utilised.debits` float64	Sum of all utilised margins (unrealised M2M + realised M2M + SPAN + Exposure + Premium + Holding sales)
`utilised.span` float64	SPAN margin blocked for all open F&O positions

attribute	
`utilised.option_premium` float64	Value of options premium received by shorting
`utilised.holding_sales` float64	Value of holdings sold during the day
`utilised.exposure` float64	Exposure margin blocked for all open F&O positions
`utilised.liquid_collateral` float64	Margin utilised against pledged liquidbees ETFs and liquid mutual funds
`utilised.delivery` float64	Margin blocked when you sell securities (20% of the value of stocks sold) from your demat or T1 holdings
`utilised.stock_collateral` float64	Margin utilised against pledged stocks/ETFs
`utilised.turnover` float64	Utilised portion of the maximum turnover limit (only applicable to certain clients)
`utilised.payout` float64	Funds paid out or withdrawn to bank account during the day

Logout

This call invalidates the access_token and destroys the API session. After this, the user should be sent through a new login flow before further interactions. This does not log the user out of the official Kite web or mobile applications.

```
curl --request DELETE \
  -H "X-Kite-Version: 3" \
  "https://api.kite.trade/session/token?api_key=xxx&access_token=yyy"
```

```
{
  "status": "success",
```

```
  "data": true
}
```

Orders

The order APIs let you place orders of different varities, modify and cancel pending orders, retrieve the daily order and more.

type	endpoint	
POST	/orders/:variety	Place an order of a particular variety
PUT	/orders/:variety/:order_id	Modify an open or pending order
DELETE	/orders/:variety/:order_id	Cancel an open or pending order
GET	/orders	Retrieve the list of all orders (open and executed) for the day
GET	/orders/:order_id	Retrieve the history of a given order
GET	/trades	Retrieve the list of all executed trades for the day
GET	/orders/:order_id/trades	Retrieve the trades generated by an order

Glossary of constants

Here are several of the constant enum values used for placing orders.

param	values	
`variety`	`regular`	Regular order
	`amo`	After Market Order

param	values	
	`co`	Cover Order ?
	`iceberg`	Iceberg Order ?
	`auction`	Auction Order ?
`order_type`	`MARKET`	Market order
	`LIMIT`	Limit order
	`SL`	Stoploss order ?
	`SL-M`	Stoploss-market order ?
`product`	`CNC`	Cash & Carry for equity ?
	`NRML`	Normal for futures and options ?
	`MIS`	Margin Intraday Squareoff for futures and options ?
`validity`	`DAY`	Regular order
	`IOC`	Immediate or Cancel
	`TTL`	Order validity in minutes

Placing orders

Placing an order implies registering it with the OMS via the API. This does not guarantee the order's receipt at the exchange. The fate of an order is dependent on several factors including market hours, availability of funds, risk checks and so on. Under normal circumstances, order placement, receipt by the OMS, transport to the exchange, execution, and the confirmation roundtrip happen instantly.

When an order is successfully placed, the API returns an `order_id`. The status of the order is not known at the moment of placing because of the aforementioned reasons.

In case of non-MARKET orders that may be open indefinitely during the course of a trading day, it is not practical to poll the order APIs continuously to know the status. For this, postbacks are ideal as they sent order updates asynchronously as they happen.

> **Note**
>
> Successful placement of an order via the API does not imply its successful execution. To know the true status of a placed order, you should scan the order history or retrieve the particular order's current details using its order_id.

Order varieties

You can place orders of different varieties—regular orders, after market orders, cover orders, iceberg orders etc. See the list of varieties here.

```
curl https://api.kite.trade/orders/regular \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "tradingsymbol=ACC" \
    -d "exchange=NSE" \
    -d "transaction_type=BUY" \
    -d "order_type=MARKET" \
    -d "quantity=1" \
    -d "product=MIS" \
    -d "validity=DAY"
```

```
{
    "status": "success",
    "data": {
      "order_id": "151220000000000"
    }
  }
```

Regular order parameters

These parameters are common across different order varieties.

parameter	
`tradingsymbol`	Tradingsymbol of the instrument ?
`exchange`	Name of the exchange (NSE, BSE, NFO, CDS, BCD, MCX)
`transaction_type`	BUY or SELL
`order_type`	Order type (MARKET, LIMIT etc.)
`quantity`	Quantity to transact
`product`	Margin product to use for the order (margins are blocked based on this) ?
`price`	The price to execute the order at (for LIMIT orders)
`trigger_price`	The price at which an order should be triggered (SL, SL-M)
`disclosed_quantity`	Quantity to disclose publicly (for equity trades)
`validity`	Order validity (DAY, IOC and TTL)
`validity_ttl`	Order life span in minutes for TTL validity orders
`iceberg_legs`	Total number of legs for iceberg order type (number of legs per Iceberg should be between 2 and 10)
`iceberg_quantity`	Split quantity for each iceberg leg order (quantity/iceberg_legs)
`auction_number` string	A unique identifier for a particular auction
`tag`	An optional tag to apply to an order to identify it (alphanumeric, max 20 chars)

Modifying orders

As long as on order is open or pending in the system, certain attributes of it may be modified. It is important to sent the right value for `:variety` in the URL.

```
curl --request PUT https://api.kite.trade/orders/regular/151220000000000 \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "order_type=MARKET" \
    -d "quantity=3" \
    -d "validity=DAY"
```

```
{
    "status": "success",
    "data": {
      "order_id": "151220000000000"
    }
}
```

Regular order parameters

parameter
`order_type`
`quantity`
`price`
`trigger_price`
`disclosed_quantity`
`validity`

Cover order (CO) parameters

parameter	
`order_id`	Unique order ID
`price`	The price to execute the order at
`trigger_price`	For `LIMIT` Cover orders

Cancelling orders

As long as on order is open or pending in the system, it can be cancelled.

```
curl --request DELETE \
    "https://api.kite.trade/orders/regular/151220000000000" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```

```
{
  "status": "success",
  "data": {
    "order_id": "151220000000000"
  }
}
```

Retrieving orders

The order history or the order book is transient as it only lives for a day in the system. When you retrieve orders, you get all the orders for the day including open, pending, and executed ones.

```
curl "https://api.kite.trade/orders" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```

```
{
  "status": "success",
  "data": [
    {
      "placed_by": "XXXXXX",
```

```
    "order_id": "100000000000000",
    "exchange_order_id": "200000000000000",
    "parent_order_id": null,
    "status": "CANCELLED",
    "status_message": null,
    "status_message_raw": null,
    "order_timestamp": "2021-05-31 09:18:57",
    "exchange_update_timestamp": "2021-05-31 09:18:58",
    "exchange_timestamp": "2021-05-31 09:15:38",
    "variety": "regular",
    "modified": false,
    "exchange": "CDS",
    "tradingsymbol": "USDINR21JUNFUT",
    "instrument_token": 412675,
    "order_type": "LIMIT",
    "transaction_type": "BUY",
    "validity": "DAY",
    "product": "NRML",
    "quantity": 1,
    "disclosed_quantity": 0,
    "price": 72,
    "trigger_price": 0,
    "average_price": 0,
    "filled_quantity": 0,
    "pending_quantity": 1,
    "cancelled_quantity": 1,
    "market_protection": 0,
    "meta": {},
    "tag": null,
    "guid": "XXXXX"
  },
  {
    "placed_by": "XXXXXX",
    "order_id": "300000000000000",
    "exchange_order_id": "400000000000000",
    "parent_order_id": null,
    "status": "COMPLETE",
    "status_message": null,
    "status_message_raw": null,
    "order_timestamp": "2021-05-31 15:20:28",
    "exchange_update_timestamp": "2021-05-31 15:20:28",
    "exchange_timestamp": "2021-05-31 15:20:28",
    "variety": "regular",
    "modified": false,
    "exchange": "NSE",
    "tradingsymbol": "IOC",
    "instrument_token": 415745,
    "order_type": "LIMIT",
    "transaction_type": "BUY",
    "validity": "DAY",
    "product": "CNC",
    "quantity": 1,
    "disclosed_quantity": 0,
```

```
      "price": 109.4,
      "trigger_price": 0,
      "average_price": 109.4,
      "filled_quantity": 1,
      "pending_quantity": 0,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {},
      "tag": null,
      "guid": "XXXXXX"
    },
    {
      "placed_by": "XXXXXX",
      "order_id": "500000000000000",
      "exchange_order_id": "600000000000000",
      "parent_order_id": null,
      "status": "COMPLETE",
      "status_message": null,
      "status_message_raw": null,
      "order_timestamp": "2021-05-31 15:20:51",
      "exchange_update_timestamp": "2021-05-31 15:20:52",
      "exchange_timestamp": "2021-05-31 15:20:52",
      "variety": "regular",
      "modified": false,
      "exchange": "NSE",
      "tradingsymbol": "IOC",
      "instrument_token": 415745,
      "order_type": "MARKET",
      "transaction_type": "SELL",
      "validity": "DAY",
      "product": "CNC",
      "quantity": 1,
      "disclosed_quantity": 0,
      "price": 0,
      "trigger_price": 0,
      "average_price": 109.35,
      "filled_quantity": 1,
      "pending_quantity": 0,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {},
      "tag": null,
      "guid": "XXXX"
    },
    {
      "placed_by": "XXXXXX",
      "order_id": "220524001859672",
      "exchange_order_id": null,
      "parent_order_id": null,
      "status": "REJECTED",
      "status_message": "Insufficient funds. Required margin is 95417.84 but
available margin is 74251.80. Check the orderbook for open orders.",
      "status_message_raw": "RMS:Margin Exceeds,Required:95417.84,
```

```
Available:74251.80 for entity account-XXXXX across exchange across segment across
product ",
      "order_timestamp": "2022-05-24 12:26:52",
      "exchange_update_timestamp": null,
      "exchange_timestamp": null,
      "variety": "iceberg",
      "modified": false,
      "exchange": "NSE",
      "tradingsymbol": "SBIN",
      "instrument_token": 779521,
      "order_type": "LIMIT",
      "transaction_type": "BUY",
      "validity": "TTL",
      "validity_ttl": 2,
      "product": "CNC",
      "quantity": 200,
      "disclosed_quantity": 0,
      "price": 463,
      "trigger_price": 0,
      "average_price": 0,
      "filled_quantity": 0,
      "pending_quantity": 0,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {
        "iceberg": {
          "leg": 1,
          "legs": 5,
          "leg_quantity": 200,
          "total_quantity": 1000,
          "remaining_quantity": 800
        }
      },
      "tag": "icebergord",
      "tags": [
        "icebergord"
      ],
      "guid": "XXXXXX"
    },
    {
      "placed_by": "XXXXXX",
      "order_id": "700000000000000",
      "exchange_order_id": "800000000000000",
      "parent_order_id": null,
      "status": "COMPLETE",
      "status_message": null,
      "status_message_raw": null,
      "order_timestamp": "2021-05-31 16:00:36",
      "exchange_update_timestamp": "2021-05-31 16:00:36",
      "exchange_timestamp": "2021-05-31 16:00:36",
      "variety": "regular",
      "modified": false,
      "exchange": "MCX",
```

```
      "tradingsymbol": "GOLDPETAL21JUNFUT",
      "instrument_token": 58424839,
      "order_type": "LIMIT",
      "transaction_type": "BUY",
      "validity": "DAY",
      "product": "NRML",
      "quantity": 1,
      "disclosed_quantity": 0,
      "price": 4854,
      "trigger_price": 0,
      "average_price": 4852,
      "filled_quantity": 1,
      "pending_quantity": 0,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {},
      "tag": "connect test order1",
      "tags": [
        "connect test order1"
      ],
      "guid": "XXXXXXX"
    },
    {
      "placed_by": "XXXXXX",
      "order_id": "9000000000000000",
      "exchange_order_id": "1000000000000000",
      "parent_order_id": null,
      "status": "COMPLETE",
      "status_message": null,
      "status_message_raw": null,
      "order_timestamp": "2021-05-31 16:08:40",
      "exchange_update_timestamp": "2021-05-31 16:08:41",
      "exchange_timestamp": "2021-05-31 16:08:41",
      "variety": "regular",
      "modified": false,
      "exchange": "MCX",
      "tradingsymbol": "GOLDPETAL21JUNFUT",
      "instrument_token": 58424839,
      "order_type": "LIMIT",
      "transaction_type": "BUY",
      "validity": "DAY",
      "product": "NRML",
      "quantity": 1,
      "disclosed_quantity": 0,
      "price": 4854,
      "trigger_price": 0,
      "average_price": 4852,
      "filled_quantity": 1,
      "pending_quantity": 0,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {},
      "tag": "connect test order2",
```

```
      "tags": [
        "connect test order2",
        "XXXXX"
      ],
      "guid": "XXXXXX"
    },
    {
      "placed_by": "XXXXXX",
      "order_id": "98000000000000000",
      "exchange_order_id": "67000000000000000",
      "parent_order_id": null,
      "status": "CANCELLED",
      "status_message": null,
      "status_message_raw": null,
      "order_timestamp": "2023-06-12 14:00:58",
      "exchange_update_timestamp": "2023-06-12 14:00:58",
      "exchange_timestamp": "2023-06-12 14:00:58",
      "variety": "auction",
      "modified": false,
      "exchange": "NSE",
      "tradingsymbol": "BHEL",
      "instrument_token": 112129,
      "order_type": "LIMIT",
      "transaction_type": "SELL",
      "validity": "DAY",
      "validity_ttl": 0,
      "product": "CNC",
      "quantity": 60,
      "disclosed_quantity": 0,
      "price": 85,
      "trigger_price": 0,
      "auction_number": "22",
      "average_price": 0,
      "filled_quantity": 0,
      "pending_quantity": 60,
      "cancelled_quantity": 0,
      "market_protection": 0,
      "meta": {},
      "tag": null,
      "guid": null
    }
  ]
}
```


Response attributes

attribute	
`order_id` string	Unique order ID
`parent_order_id` string	Order ID of the parent order (only applicable in case of multi-legged orders like CO)
`exchange_order_id` null, string	Exchange generated order ID. Orders that don't reach the exchange have null IDs
`modified` bool	Indicate that the order has been modified since placement by the user
`placed_by` string	ID of the user that placed the order. This may different from the user's ID for orders placed outside of Kite, for instance, by dealers at the brokerage using dealer terminals
`variety` string	Order variety (regular, amo, co etc.)
`status` string	Current status of the order. Most common values or COMPLETE, REJECTED, CANCELLED, and OPEN. There may be other values as well.
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`exchange` string	Exchange
`instrument_token` string	The numerical identifier issued by the exchange representing the instrument. Used for subscribing to live market data over WebSocket
`transaction_type` string	BUY or SELL

attribute	
`order_type` string	Order type (MARKET, LIMIT etc.)
`product` string>	Margin product to use for the order (margins are blocked based on this) ?
`validity` string	Order validity
`price` float64	Price at which the order was placed (LIMIT orders)
`quantity` int64	Quantity ordered
`trigger_price` float64	Trigger price (for SL, SL-M, CO orders)
`average_price` float64	Average price at which the order was executed (only for COMPLETE orders)
`pending_quantity` int64	Pending quantity to be filled
`filled_quantity` int64	Quantity that's been filled
`disclosed_quantity` int64	Quantity to be disclosed (may be different from actual quantity) to the public exchange orderbook. Only for equities
`order_timestamp` string	Timestamp at which the order was registered by the API
`exchange_timestamp` string	Timestamp at which the order was registered by the exchange. Orders that don't reach the exchange have null timestamps

attribute	
`exchange_update_timestamp` string	Timestamp at which an order's state changed at the exchange
`status_message` null, string	Textual description of the order's status. Failed orders come with human readable explanation
`status_message_raw` null, string	Raw textual description of the failed order's status, as received from the OMS
`cancelled_quantity` int64	Quantity that's cancelled
`auction_number` string	A unique identifier for a particular auction
`meta` {}, string	Map of arbitrary fields that the system may attach to an order.
`tag` null, string	An optional tag to apply to an order to identify it (alphanumeric, max 20 chars)
`guid` string	Unusable request id to avoid order duplication

Order statuses

The `status` field in the order response shows the current state of the order. The status values are largely self explanatory. The most common statuses are `OPEN`, `COMPLETE`, `CANCELLED`, and `REJECTED`.

An order can traverse through several interim and temporary statuses during its lifetime. For example, when an order is first placed or modified, it instantly passes through several stages before reaching its end state. Some of these are highlighted below.

status	
`PUT ORDER REQ RECEIVED`	Order request has been received by the backend
`VALIDATION PENDING`	Order pending validation by the RMS (Risk Management System)
`OPEN PENDING`	Order is pending registration at the exchange
`MODIFY VALIDATION PENDING`	Order's modification values are pending validation by the RMS
`MODIFY PENDING`	Order's modification values are pending registration at the exchange
`TRIGGER PENDING`	Order's placed but the fill is pending based on a trigger price.
`CANCEL PENDING`	Order's cancellation request is pending registration at the exchange
`AMO REQ RECEIVED`	Same as `PUT ORDER REQ RECEIVED`, but for AMOs

Tagging orders

Often, it may be necessary to tag and filter orders based on various criteria, for instance, to filter out all orders that came from a particular application or an `api_key` of yours. The `tag` field comes in handy here. It let's you send an arbitrary string while placing an order. This can be a unique ID, or something that indicates a particular type or context, for example. When the orderbook is retrieved, this value will be present in the `tag` field in the response.

Multi-legged orders (CO)

Cover orders are "multi-legged" orders, where, a single order you place (first-leg) may spawn new orders (second-leg) based on the conditions you set on the first-leg order. These orders have

special properties, where the first-leg order creates a position. The position is exited when the second-leg order is executed or cancelled.

These second-leg orders will have a `parent_order_id` field indicating the `order_id` of its parent order, that is, the first-leg order. This field may be required while modifying or cancelling an open second-leg order.

Retrieving an order's history

```
curl "https://api.kite.trade/orders/171229000724687" \
     -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "average_price": 0,
      "cancelled_quantity": 0,
      "disclosed_quantity": 0,
      "exchange": "NSE",
      "exchange_order_id": null,
      "exchange_timestamp": null,
      "filled_quantity": 0,
      "instrument_token": 1,
      "order_id": "171229000724687",
      "order_timestamp": "2017-12-29 11:06:52",
      "order_type": "LIMIT",
      "parent_order_id": null,
      "pending_quantity": 1,
      "placed_by": "DA0017",
      "price": 300,
      "product": "CNC",
      "quantity": 1,
      "status": "PUT ORDER REQ RECEIVED",
      "status_message": null,
      "tag": null,
      "tradingsymbol": "SBIN",
      "transaction_type": "BUY",
      "trigger_price": 0,
      "validity": "DAY",
      "variety": "regular",
      "modified": false
    },
    {
      "average_price": 0,
      "cancelled_quantity": 0,
      "disclosed_quantity": 0,
```

```
      "exchange": "NSE",
      "exchange_order_id": null,
      "exchange_timestamp": null,
      "filled_quantity": 0,
      "instrument_token": 779521,
      "order_id": "171229000724687",
      "order_timestamp": "2017-12-29 11:06:52",
      "order_type": "LIMIT",
      "parent_order_id": null,
      "pending_quantity": 1,
      "placed_by": "DA0017",
      "price": 300,
      "product": "CNC",
      "quantity": 1,
      "status": "VALIDATION PENDING",
      "status_message": null,
      "tag": null,
      "tradingsymbol": "SBIN",
      "transaction_type": "BUY",
      "trigger_price": 0,
      "validity": "DAY",
      "variety": "regular",
      "modified": false
    },
    {
      "average_price": 0,
      "cancelled_quantity": 0,
      "disclosed_quantity": 0,
      "exchange": "NSE",
      "exchange_order_id": null,
      "exchange_timestamp": null,
      "filled_quantity": 0,
      "instrument_token": 779521,
      "order_id": "171229000724687",
      "order_timestamp": "2017-12-29 11:06:52",
      "order_type": "LIMIT",
      "parent_order_id": null,
      "pending_quantity": 1,
      "placed_by": "DA0017",
      "price": 300,
      "product": "CNC",
      "quantity": 1,
      "status": "OPEN PENDING",
      "status_message": null,
      "tag": null,
      "tradingsymbol": "SBIN",
      "transaction_type": "BUY",
      "trigger_price": 0,
      "validity": "DAY",
      "variety": "regular",
      "modified": false
    },
    {
```

```
    "average_price": 0,
    "cancelled_quantity": 0,
    "disclosed_quantity": 0,
    "exchange": "NSE",
    "exchange_order_id": "1300000001887410",
    "exchange_timestamp": "2017-12-29 11:06:52",
    "filled_quantity": 0,
    "instrument_token": 779521,
    "order_id": "171229000724687",
    "order_timestamp": "2017-12-29 11:06:52",
    "order_type": "LIMIT",
    "parent_order_id": null,
    "pending_quantity": 1,
    "placed_by": "DA0017",
    "price": 300,
    "product": "CNC",
    "quantity": 1,
    "status": "OPEN",
    "status_message": null,
    "tag": null,
    "tradingsymbol": "SBIN",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "validity": "DAY",
    "variety": "regular",
    "modified": false
  },
  {
    "average_price": 0,
    "cancelled_quantity": 0,
    "disclosed_quantity": 0,
    "exchange": "NSE",
    "exchange_order_id": "1300000001887410",
    "exchange_timestamp": "2017-12-29 11:06:52",
    "filled_quantity": 0,
    "instrument_token": 779521,
    "order_id": "171229000724687",
    "order_timestamp": "2017-12-29 11:08:16",
    "order_type": "LIMIT",
    "parent_order_id": null,
    "pending_quantity": 1,
    "placed_by": "DA0017",
    "price": 300,
    "product": "CNC",
    "quantity": 1,
    "status": "MODIFY VALIDATION PENDING",
    "status_message": null,
    "tag": null,
    "tradingsymbol": "SBIN",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "validity": "DAY",
    "variety": "regular",
```

```
    "modified": false
  },
  {
    "average_price": 0,
    "cancelled_quantity": 0,
    "disclosed_quantity": 0,
    "exchange": "NSE",
    "exchange_order_id": "1300000001887410",
    "exchange_timestamp": "2017-12-29 11:06:52",
    "filled_quantity": 0,
    "instrument_token": 779521,
    "order_id": "171229000724687",
    "order_timestamp": "2017-12-29 11:08:16",
    "order_type": "LIMIT",
    "parent_order_id": null,
    "pending_quantity": 1,
    "placed_by": "DA0017",
    "price": 300,
    "product": "CNC",
    "quantity": 1,
    "status": "MODIFY PENDING",
    "status_message": null,
    "tag": null,
    "tradingsymbol": "SBIN",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "validity": "DAY",
    "variety": "regular",
    "modified": false
  },
  {
    "average_price": 0,
    "cancelled_quantity": 0,
    "disclosed_quantity": 0,
    "exchange": "NSE",
    "exchange_order_id": "1300000001887410",
    "exchange_timestamp": "2017-12-29 11:08:16",
    "filled_quantity": 0,
    "instrument_token": 779521,
    "order_id": "171229000724687",
    "order_timestamp": "2017-12-29 11:08:16",
    "order_type": "LIMIT",
    "parent_order_id": null,
    "pending_quantity": 1,
    "placed_by": "DA0017",
    "price": 300,
    "product": "CNC",
    "quantity": 1,
    "status": "MODIFIED",
    "status_message": null,
    "tag": null,
    "tradingsymbol": "SBIN",
    "transaction_type": "BUY",
```

```
      "trigger_price": 0,
      "validity": "DAY",
      "variety": "regular",
      "modified": false
    },
    {
      "average_price": 0,
      "cancelled_quantity": 0,
      "disclosed_quantity": 0,
      "exchange": "NSE",
      "exchange_order_id": "1300000001887410",
      "exchange_timestamp": "2017-12-29 11:08:16",
      "filled_quantity": 0,
      "instrument_token": 779521,
      "order_id": "171229000724687",
      "order_timestamp": "2017-12-29 11:08:16",
      "order_type": "LIMIT",
      "parent_order_id": null,
      "pending_quantity": 1,
      "placed_by": "DA0017",
      "price": 300.1,
      "product": "CNC",
      "quantity": 1,
      "status": "OPEN",
      "status_message": null,
      "tag": null,
      "tradingsymbol": "SBIN",
      "transaction_type": "BUY",
      "trigger_price": 0,
      "validity": "DAY",
      "variety": "regular",
      "modified": false
    }
  ]
}
```


Every order, right after being placed, goes through multiple stages internally in the OMS. Initial validation, RMS (Risk Management System) checks and so on before it goes to the exchange. In addition, an open order, when modified, again goes through these stages.

Retrieving all trades

While an orders is sent as a single entity, it may be executed in arbitrary chunks at the exchange depending on market conditions. For instance, an order for 10 quantity of an instrument can be executed in chunks of 5, 1, 1, 3 or any such combination. Each individual execution that fills an order partially is a trade. An order may have one or more trades.

This API returns a list of all trades generated by all executed orders for the day.

```
curl "https://api.kite.trade/trades" \
     -H "X-Kite-Version: 3" \
     -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "trade_id": "10000000",
      "order_id": "200000000000000",
      "exchange": "NSE",
      "tradingsymbol": "SBIN",
      "instrument_token": 779521,
      "product": "CNC",
      "average_price": 420.65,
      "quantity": 1,
      "exchange_order_id": "300000000000000",
      "transaction_type": "BUY",
      "fill_timestamp": "2021-05-31 09:16:39",
      "order_timestamp": "09:16:39",
      "exchange_timestamp": "2021-05-31 09:16:39"
    },
    {
      "trade_id": "40000000",
      "order_id": "500000000000000",
      "exchange": "CDS",
      "tradingsymbol": "USDINR21JUNFUT",
      "instrument_token": 412675,
      "product": "MIS",
      "average_price": 72.755,
      "quantity": 1,
      "exchange_order_id": "600000000000000",
      "transaction_type": "BUY",
      "fill_timestamp": "2021-05-31 11:18:27",
      "order_timestamp": "11:18:27",
      "exchange_timestamp": "2021-05-31 11:18:27"
    },
    {
      "trade_id": "70000000",
      "order_id": "800000000000000",
      "exchange": "MCX",
      "tradingsymbol": "GOLDPETAL21JUNFUT",
      "instrument_token": 58424839,
      "product": "NRML",
      "average_price": 4852,
      "quantity": 1,
      "exchange_order_id": "312115100078593",
      "transaction_type": "BUY",
      "fill_timestamp": "2021-05-31 16:00:36",
```

```
      "order_timestamp": "16:00:36",
      "exchange_timestamp": "2021-05-31 16:00:36"
    },
    {
      "trade_id": "90000000",
      "order_id": "1100000000000000",
      "exchange": "MCX",
      "tradingsymbol": "GOLDPETAL21JUNFUT",
      "instrument_token": 58424839,
      "product": "NRML",
      "average_price": 4852,
      "quantity": 1,
      "exchange_order_id": "1200000000000000",
      "transaction_type": "BUY",
      "fill_timestamp": "2021-05-31 16:08:41",
      "order_timestamp": "16:08:41",
      "exchange_timestamp": "2021-05-31 16:08:41"
    }
  ]
}
```


Response attributes

attribute	
`trade_id` string	Exchange generated trade ID
`order_id` string	Unique order ID
`exchange_order_id` null, string	Exchange generated order ID
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`exchange` string	Exchange
`instrument_token` string	The numerical identifier issued by the exchange representing the instrument. Used for subscribing to live market data over WebSocket

attribute	
`transaction_type` string	BUY or SELL
`product` string	Margin product to use for the order (margins are blocked based on this) ?
`average_price` float64	Price at which the quantity was filled
`filled` int64	Filled quantity
`fill_timestamp` string	Timestamp at which the trade was filled at the exchange
`order_timestamp` string	Timestamp at which the order was registered by the API
`exchange_timestamp` string	Timestamp at which the order was registered by the exchange

Retrieving an order's trades

This API returns the trades spawned and executed by a particular order.

```
curl "https://api.kite.trade/orders/200000000000000/trades" \
     -H "X-Kite-Version: 3" \
     -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "trade_id": "10000000",
      "order_id": "200000000000000",
      "exchange": "MCX",
      "tradingsymbol": "GOLDPETAL21JUNFUT",
      "instrument_token": 58424839,
```

```
      "product": "NRML",
      "average_price": 4852,
      "quantity": 1,
      "exchange_order_id": "300000000000000",
      "transaction_type": "BUY",
      "fill_timestamp": "2021-05-31 16:00:36",
      "order_timestamp": "16:00:36",
      "exchange_timestamp": "2021-05-31 16:00:36"
    }
  ]
}
```

GTT - Good Till Triggered orders

The GTT APIs allow you to place, modify and manage GTTs.

type	endpoint	
POST	/gtt/triggers	Place a GTT
GET	/gtt/triggers	Retrieve a list of all GTTs visible in GTT order book
GET	/gtt/triggers/:id	Retrieve an individual trigger
PUT	/gtt/triggers/:id	Modify an active GTT
DELETE	/gtt/triggers/:id	Delete an active GTT

Placing triggers

```
curl https://api.kite.trade/gtt/triggers \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
    -d 'type=single' \
    -d 'condition={"exchange":"NSE", "tradingsymbol":"INFY", "trigger_values":
[702.0], "last_price": 798.0}' \
    -d 'orders=[{"exchange":"NSE", "tradingsymbol": "INFY", "transaction_type":
"BUY", "quantity": 1, "order_type": "LIMIT","product": "CNC", "price": 702.5}]'
```

```
{"status":"success","data":{"trigger_id":123}}
```

Order parameters

parameter	
type	The type of GTT
condition	The condition parameters (json object)
orders	The orders to be placed (json array).

Condition parameters

The API expects the following parameters encoded as json in the condition post parameter.

parameter	
exchange	Name of the exchange
tradingsymbol	Trading symbol of the instrument
trigger_values	Trigger values (json array)
last_price	Last price of the instrument at the time of placement

Orders list

The API expects a list of orders encoded as a json array. The index of the order is used by API to determine which order is executed when a trigger value is reached.

The order object inside the list expects the following parameters.

parameter	
exchange	Name of the exchange
tradingsymbol	Trading symbol of the instrument
transaction_type	`BUY` or `SELL`

parameter	
quantity	Quantity to transact
order_type	`LIMIT`
product	Margin product to use for the order
price	The min or max price to execute the order at (for LIMIT orders)

Sample order:

```
{
  "exchange": "NSE",
  "tradingsymbol": "INFY",
  "transaction_type": "BUY",
  "quantity": 1,
  "order_type": "LIMIT",
  "product": "CNC",
  "price": 702.5
}
```

Type

The GTT API supports the following type of GTTs. The `type` in the post parameter is used to parse the `condition` passed to it. The following types are supported by GTT API.

`SINGLE`

The single leg trigger type expects a single trigger value, and executes the first order that is in the `orders` array when the trigger value is reached.

Sample `condition`:

```
{
  "exchange": "NSE",
  "tradingsymbol": "INFY",
  "trigger_values": [702.0],
  "last_price": 798.0
}
```

> **Note**
>
> This trigger type expects only one trigger value inside the `trigger_values`

Sample `orders`:

```
[
  {
    "exchange": "NSE",
    "tradingsymbol": "INFY",
    "transaction_type": "BUY",
    "quantity": 1,
    "order_type": "LIMIT",
    "product": "CNC",
    "price": 702.5
  }
]
```

`TWO-LEG`

The `two-leg` trigger implements the OCO (One Cancels Other) order. It expects two trigger values and executes the corresponding order in the `orders` array when either of the trigger value is reached, the other order is lain dormant.

Sample `condition`:

```
{
  "exchange": "NSE",
  "tradingsymbol": "INFY",
  "trigger_values": [702.0, 798.0],
  "last_price": 742.0
}
```

> **Note**
>
> This trigger type expects two trigger value inside the `trigger_values`

Sample `orders`:

```
[
  {
    "exchange": "NSE",
    "tradingsymbol": "INFY",
    "transaction_type": "SELL",
```

```
    "quantity": 1,
    "order_type": "LIMIT",
    "product": "CNC",
    "price": 702.5
  },
  {
    "exchange": "NSE",
    "tradingsymbol": "INFY",
    "transaction_type": "SELL",
    "quantity": 1,
    "order_type": "LIMIT",
    "product": "CNC",
    "price": 798.5
  }
]
```

Retrieving triggers

Active GTTs and GTTs in others states (previous 7 days) can be obtained by a GET API call to the `/triggers/gtt` endpoint.

```
curl  "https://api.kite.trade/gtt/triggers" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```

```
{
    "status": "success",
    "data": [
        {
            "id": 112127,
            "user_id": "XX0000",
            "parent_trigger": null,
            "type": "single",
            "created_at": "2019-09-12 13:25:16",
            "updated_at": "2019-09-12 13:25:16",
            "expires_at": "2020-09-12 13:25:16",
            "status": "active",
            "condition": {
                "exchange": "NSE",
                "last_price": 798,
                "tradingsymbol": "INFY",
                "trigger_values": [
                    702
                ],
                "instrument_token": 408065
            },
            "orders": [
```

```
            {
                "exchange": "NSE",
                "tradingsymbol": "INFY",
                "product": "CNC",
                "order_type": "LIMIT",
                "transaction_type": "BUY",
                "quantity": 1,
                "price": 702.5,
                "result": null
            }
        ],
        "meta": {}
    },
    {
        "id": 105099,
        "user_id": "XX0000",
        "parent_trigger": null,
        "type": "two-leg",
        "created_at": "2019-09-09 15:13:22",
        "updated_at": "2019-09-09 15:15:08",
        "expires_at": "2020-01-01 12:00:00",
        "status": "triggered",
        "condition": {
            "exchange": "NSE",
            "last_price": 102.6,
            "tradingsymbol": "RAIN",
            "trigger_values": [
                102.0,
                103.7
            ],
            "instrument_token": 3926273
        },
        "orders": [
            {
                "exchange": "NSE",
                "tradingsymbol": "RAIN",
                "product": "CNC",
                "order_type": "LIMIT",
                "transaction_type": "SELL",
                "quantity": 1,
                "price": 1,
                "result": null
            },
            {
                "exchange": "NSE",
                "tradingsymbol": "RAIN",
                "product": "CNC",
                "order_type": "LIMIT",
                "transaction_type": "SELL",
                "quantity": 1,
                "price": 1,
                "result": {
                    "account_id": "XX0000",
```

```
                        "exchange": "NSE",
                        "tradingsymbol": "RAIN",
                        "validity": "DAY",
                        "product": "CNC",
                        "order_type": "LIMIT",
                        "transaction_type": "SELL",
                        "quantity": 1,
                        "price": 1,
                        "meta": "{\"app_id\":12617,\"gtt\":105099}",
                        "timestamp": "2019-09-09 15:15:08",
                        "triggered_at": 103.7,
                        "order_result": {
                            "status": "failed",
                            "order_id": "",
                            "rejection_reason": "Your order price is lower than
the current lower circuit limit of 70.65. Place an order within the daily range."
                        }
                    }
                }
            ],
            "meta": null
        }
    ]
}
```


Retrieve trigger

Given a GTT ID, the GET API call to this endpoint will return details of the trigger irrespective of the age or status of the GTT.

```
curl https://api.kite.trade/gtt/triggers/123 \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token'
```


```
{
    "status": "success",
    "data": {
        "id": 123,
        "user_id": "XX0000",
        "parent_trigger": null,
        "type": "two-leg",
        "created_at": "2019-09-09 15:13:22",
        "updated_at": "2019-09-09 15:15:08",
        "expires_at": "2020-01-01 12:00:00",
        "status": "triggered",
        "condition": {
            "exchange": "NSE",
```

```
            "last_price": 102.6,
            "tradingsymbol": "RAIN",
            "trigger_values": [
                102.0,
                103.7
            ],
            "instrument_token": 3926273
        },
        "orders": [
            {
                "exchange": "NSE",
                "tradingsymbol": "RAIN",
                "product": "CNC",
                "order_type": "LIMIT",
                "transaction_type": "SELL",
                "quantity": 1,
                "price": 1,
                "result": null
            },
            {
                "exchange": "NSE",
                "tradingsymbol": "RAIN",
                "product": "CNC",
                "order_type": "LIMIT",
                "transaction_type": "SELL",
                "quantity": 1,
                "price": 1,
                "result": {
                    "account_id": "XX0000",
                    "exchange": "NSE",
                    "tradingsymbol": "RAIN",
                    "validity": "DAY",
                    "product": "CNC",
                    "order_type": "LIMIT",
                    "transaction_type": "SELL",
                    "quantity": 1,
                    "price": 1,
                    "meta": "{\"app_id\":12617,\"gtt\":105099}",
                    "timestamp": "2019-09-09 15:15:08",
                    "triggered_at": 103.7,
                    "order_result": {
                        "status": "failed",
                        "order_id": "",
                        "rejection_reason": "Your order price is lower than the
current lower circuit limit of 70.65. Place an order within the daily range."
                    }
                }
            }
        ],
        "meta": null
    }
}
```

Status

GTTs can be in the following state

status	
active	indicates that the trigger is active.
triggered	indicates that the trigger was triggered by core.
disabled	indicates that the trigger is disabled and action is expected from the user.
expired	indicates that the trigger has expired based on its expiry date.
cancelled	indicates that the trigger has been cancelled by our system.
rejected	indicates that the trigger has been rejected by our system.
deleted	indicates that the trigger was deleted by the user.

Modify trigger

To modify a GTT you need to send a PUT call with updated parameters.

```
curl --request PUT https://api.kite.trade/gtt/triggers/123 \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
    -d 'type=single' \
    -d 'condition={"exchange":"NSE", "tradingsymbol":"INFY", "trigger_values":
[701.0], "last_price": 798.0}' \
    -d 'orders=[{"exchange":"NSE", "tradingsymbol": "INFY", "transaction_type":
"BUY", "quantity": 1, "order_type": "LIMIT","product": "CNC", "price": 702.5}]'
```

```
{"status":"success","data":{"trigger_id":123}}
```

> **Note**
>
> It is recommended to fetch the trigger using trigger ID and modify the values and send that to the modify endpoint.

Delete trigger

```
curl --request DELETE https://api.kite.trade/gtt/triggers/123 \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
```


```
{"status":"success","data":{"trigger_id":123}}
```

Portfolio

A user's portfolio consists of long term equity holdings and short term positions. The portfolio APIs return instruments in a portfolio with up-to-date profit and loss computations.

type	endpoint	
GET	/portfolio/holdings	Retrieve the list of long term equity holdings
GET	/portfolio/positions	Retrieve the list of short term positions
PUT	/portfolio/positions	Convert the margin product of an open position
GET	/portfolio/holdings/auctions	Retrieve the list of auctions that are currently being held

---

Holdings

Holdings contain the user's portfolio of long term equity delivery stocks. An instrument in a holdings portfolio remains there indefinitely until its sold or is delisted or changed by the exchanges. Underneath it all, instruments in the holdings reside in the user's DEMAT account, as settled by exchanges and clearing institutions.

```
curl "https://api.kite.trade/portfolio/holdings" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "tradingsymbol": "GOLDBEES",
      "exchange": "BSE",
      "instrument_token": 151064324,
      "isin": "INF204KB17I5",
```

```
      "product": "CNC",
      "price": 0,
      "quantity": 2,
      "used_quantity": 0,
      "t1_quantity": 0,
      "realised_quantity": 2,
      "authorised_quantity": 0,
      "authorised_date": "2021-06-08 00:00:00",
      "opening_quantity": 2,
      "collateral_quantity": 0,
      "collateral_type": "",
      "discrepancy": false,
      "average_price": 40.67,
      "last_price": 42.47,
      "close_price": 42.28,
      "pnl": 3.5999999999999943,
      "day_change": 0.18999999999999773,
      "day_change_percentage": 0.44938505203405327
    },
    {
      "tradingsymbol": "IDEA",
      "exchange": "NSE",
      "instrument_token": 3677697,
      "isin": "INE669E01016",
      "product": "CNC",
      "price": 0,
      "quantity": 5,
      "used_quantity": 0,
      "t1_quantity": 0,
      "realised_quantity": 5,
      "authorised_quantity": 0,
      "authorised_date": "2021-06-08 00:00:00",
      "opening_quantity": 5,
      "collateral_quantity": 0,
      "collateral_type": "",
      "discrepancy": false,
      "average_price": 8.466,
      "last_price": 10,
      "close_price": 10.1,
      "pnl": 7.6700000000000035,
      "day_change": -0.09999999999999964,
      "day_change_percentage": -0.9900990099009866
    }
  ]
}
```


Response attributes

attribute	
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`exchange` string	Exchange
`instrument_token` uint32	Unique instrument identifier (used for WebSocket subscriptions)
`isin` string	The standard ISIN representing stocks listed on multiple exchanges
`t1_quantity` int64	Quantity on T+1 day after order execution. Stocks are usually delivered into DEMAT accounts on T+2 ?
`realised_quantity` int64	Quantity delivered to Demat
`quantity` int64	Net quantity (T+1 + realised)
`used_quantity` int64	Quantity sold from the net holding quantity
`authorised_quantity` int64	Quantity authorised at the depository for sale
`opening_quantity` int64	Quantity carried forward over night
`authorised_date` string	Date on which user can sell required holding stock
`price` float64	
`average_price` float64	Average price at which the net holding quantity was acquired

attribute	
`last_price` float64	Last traded market price of the instrument
`close_price` float64	Closing price of the instrument from the last trading day
`pnl` float64	Net returns on the stock; Profit and loss
`day_change` float64	Day's change in absolute value for the stock
`day_change_percentage` float64	Day's change in percentage for the stock
`product` string	Margin product applied to the holding
`collateral_quantity` int64	Quantity used as collateral
`collateral_type` null,string	Type of collateral
`discrepancy` bool	Indicates whether holding has any price discrepancy

Holdings auction list

This API returns a list of auctions that are currently being held, along with details about each auction such as the auction number, the security being auctioned, the last price of the security, and the quantity of the security being offered. Only the stocks that you hold in your demat account will be shown in the auctions list.

```
curl "https://api.kite.trade/portfolio/holdings/auctions" \
    -H "X-Kite-Version: 3" \
```

```
    -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "tradingsymbol": "ASHOKLEY",
      "exchange": "NSE",
      "instrument_token": 54282,
      "isin": "INE208A01029",
      "product": "CNC",
      "price": 0,
      "quantity": 1,
      "t1_quantity": 0,
      "realised_quantity": 1,
      "authorised_quantity": 0,
      "authorised_date": "2022-12-21 00:00:00",
      "opening_quantity": 1,
      "collateral_quantity": 0,
      "collateral_type": "",
      "discrepancy": false,
      "average_price": 131.95,
      "last_price": 142.5,
      "close_price": 145.1,
      "pnl": 10.550000000000011,
      "day_change": -2.5999999999999943,
      "day_change_percentage": -1.7918676774638143,
      "auction_number": "20"
    },
    {
      "tradingsymbol": "BHEL",
      "exchange": "NSE",
      "instrument_token": 112138,
      "isin": "INE257A01026",
      "product": "CNC",
      "price": 0,
      "quantity": 5,
      "t1_quantity": 0,
      "realised_quantity": 5,
      "authorised_quantity": 0,
      "authorised_date": "2022-12-21 00:00:00",
      "opening_quantity": 5,
      "collateral_quantity": 0,
      "collateral_type": "",
      "discrepancy": false,
      "average_price": 75.95,
      "last_price": 81.1,
      "close_price": 84,
      "pnl": 25.749999999999957,
      "day_change": -2.9000000000000057,
      "day_change_percentage": -3.4523809523809588,
      "auction_number": "34"
```

```
    },
    {
      "tradingsymbol": "SBIN",
      "exchange": "NSE",
      "instrument_token": 779530,
      "isin": "INE062A01020",
      "product": "CNC",
      "price": 0,
      "quantity": 3,
      "t1_quantity": 0,
      "realised_quantity": 3,
      "authorised_quantity": 0,
      "authorised_date": "2022-12-21 00:00:00",
      "opening_quantity": 3,
      "collateral_quantity": 0,
      "collateral_type": "",
      "discrepancy": false,
      "average_price": 573.4,
      "last_price": 593.75,
      "close_price": 604.6,
      "pnl": 61.05000000000007,
      "day_change": -10.850000000000023,
      "day_change_percentage": -1.794574925570629,
      "auction_number": "7529"
    }
  ]
}
```


Response attributes

attribute	
`auction_number` string	A unique identifier for a particular auction

Positions

Positions contain the user's portfolio of short to medium term derivatives (futures and options contracts) and intraday equity stocks. Instruments in the positions portfolio remain there until they're sold, or until expiry, which, for derivatives, is typically three months. Equity positions carried overnight move to the holdings portfolio the next day.

The positions API returns two sets of positions, `net` and `day`. `net` is the actual, current net position portfolio, while `day` is a snapshot of the buying and selling activity for that particular day. This is useful for computing intraday profits and losses for trading strategies.

```
curl "https://api.kite.trade/portfolio/positions" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```


```
{
    "status": "success",
    "data": {
        "net": [
            {
                "tradingsymbol": "LEADMINI17DECFUT",
                "exchange": "MCX",
                "instrument_token": 53496327,
                "product": "NRML",
                "quantity": 1,
                "overnight_quantity": 0,
                "multiplier": 1000,
                "average_price": 161.05,
                "close_price": 0,
                "last_price": 161.05,
                "value": -161050,
                "pnl": 0,
                "m2m": 0,
                "unrealised": 0,
                "realised": 0,
                "buy_quantity": 1,
                "buy_price": 161.05,
                "buy_value": 161050,
                "buy_m2m": 161050,
                "sell_quantity": 0,
                "sell_price": 0,
                "sell_value": 0,
                "sell_m2m": 0,
                "day_buy_quantity": 1,
                "day_buy_price": 161.05,
                "day_buy_value": 161050,
                "day_sell_quantity": 0,
                "day_sell_price": 0,
                "day_sell_value": 0
            },
            {
                "tradingsymbol": "GOLDGUINEA17DECFUT",
                "exchange": "MCX",
                "instrument_token": 53505799,
                "product": "NRML",
                "quantity": 0,
                "overnight_quantity": 3,
                "multiplier": 1,
```

```
            "average_price": 0,
            "close_price": 23232,
            "last_price": 23355,
            "value": 801,
            "pnl": 801,
            "m2m": 276,
            "unrealised": 801,
            "realised": 0,
            "buy_quantity": 4,
            "buy_price": 23139.75,
            "buy_value": 92559,
            "buy_m2m": 93084,
            "sell_quantity": 4,
            "sell_price": 23340,
            "sell_value": 93360,
            "sell_m2m": 93360,
            "day_buy_quantity": 1,
            "day_buy_price": 23388,
            "day_buy_value": 23388,
            "day_sell_quantity": 4,
            "day_sell_price": 23340,
            "day_sell_value": 93360
        },
        {
            "tradingsymbol": "SBIN",
            "exchange": "NSE",
            "instrument_token": 779521,
            "product": "CO",
            "quantity": 0,
            "overnight_quantity": 0,
            "multiplier": 1,
            "average_price": 0,
            "close_price": 0,
            "last_price": 308.4,
            "value": -2,
            "pnl": -2,
            "m2m": -2,
            "unrealised": -2,
            "realised": 0,
            "buy_quantity": 1,
            "buy_price": 311,
            "buy_value": 311,
            "buy_m2m": 311,
            "sell_quantity": 1,
            "sell_price": 309,
            "sell_value": 309,
            "sell_m2m": 309,
            "day_buy_quantity": 1,
            "day_buy_price": 311,
            "day_buy_value": 311,
            "day_sell_quantity": 1,
            "day_sell_price": 309,
            "day_sell_value": 309
```

```
            }
        ],
        "day": [
            {
                "tradingsymbol": "GOLDGUINEA17DECFUT",
                "exchange": "MCX",
                "instrument_token": 53505799,
                "product": "NRML",
                "quantity": -3,
                "overnight_quantity": 0,
                "multiplier": 1,
                "average_price": 23340,
                "close_price": 23232,
                "last_price": 23355,
                "value": 69972,
                "pnl": -93,
                "m2m": -93,
                "unrealised": -93,
                "realised": 0,
                "buy_quantity": 1,
                "buy_price": 23388,
                "buy_value": 23388,
                "buy_m2m": 23388,
                "sell_quantity": 4,
                "sell_price": 23340,
                "sell_value": 93360,
                "sell_m2m": 93360,
                "day_buy_quantity": 1,
                "day_buy_price": 23388,
                "day_buy_value": 23388,
                "day_sell_quantity": 4,
                "day_sell_price": 23340,
                "day_sell_value": 93360
            },
            {
                "tradingsymbol": "LEADMINI17DECFUT",
                "exchange": "MCX",
                "instrument_token": 53496327,
                "product": "NRML",
                "quantity": 1,
                "overnight_quantity": 0,
                "multiplier": 1000,
                "average_price": 161.05,
                "close_price": 0,
                "last_price": 161.05,
                "value": -161050,
                "pnl": 0,
                "m2m": 0,
                "unrealised": 0,
                "realised": 0,
                "buy_quantity": 1,
                "buy_price": 161.05,
                "buy_value": 161050,
```

```
                "buy_m2m": 161050,
                "sell_quantity": 0,
                "sell_price": 0,
                "sell_value": 0,
                "sell_m2m": 0,
                "day_buy_quantity": 1,
                "day_buy_price": 161.05,
                "day_buy_value": 161050,
                "day_sell_quantity": 0,
                "day_sell_price": 0,
                "day_sell_value": 0
            },
            {
                "tradingsymbol": "SBIN",
                "exchange": "NSE",
                "instrument_token": 779521,
                "product": "CO",
                "quantity": 0,
                "overnight_quantity": 0,
                "multiplier": 1,
                "average_price": 0,
                "close_price": 0,
                "last_price": 308.4,
                "value": -2,
                "pnl": -2,
                "m2m": -2,
                "unrealised": -2,
                "realised": 0,
                "buy_quantity": 1,
                "buy_price": 311,
                "buy_value": 311,
                "buy_m2m": 311,
                "sell_quantity": 1,
                "sell_price": 309,
                "sell_value": 309,
                "sell_m2m": 309,
                "day_buy_quantity": 1,
                "day_buy_price": 311,
                "day_buy_value": 311,
                "day_sell_quantity": 1,
                "day_sell_price": 309,
                "day_sell_value": 309
            }
        ]
    }
}
```


Response attributes

attribute	
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`exchange` string	Exchange
`instrument_token` uint32	The numerical identifier issued by the exchange representing the instrument. Used for subscribing to live market data over WebSocket
`product` string	Margin product applied to the position
`quantity` int64	Quantity held
`overnight_quantity` int64	Quantity held previously and carried forward over night
`multiplier` int64	The quantity/lot size multiplier used for calculating P&Ls.
`average_price` float64	Average price at which the net position quantity was acquired
`close_price` float64	Closing price of the instrument from the last trading day
`last_price` float64	Last traded market price of the instrument
`value` float64	Net value of the position
`pnl` float64	Net returns on the position; Profit and loss

attribute	
`m2m` float64	Mark to market returns (computed based on the last close and the last traded price)
`unrealised` float64	Unrealised intraday returns
`realised` float64	Realised intraday returns
`buy_quantity` int64	Quantity bought and added to the position
`buy_price` float64	Average price at which quantities were bought
`buy_value` float64	Net value of the bought quantities
`buy_m2m` float64	Mark to market returns on the bought quantities
`day_buy_quantity` int64	Quantity bought and added to the position during the day
`day_buy_price` float64	Average price at which quantities were bought during the day
`day_buy_value` float64	Net value of the quantities bought during the day
`sell_quantity` int64	Quantity sold off from the position
`sell_price` float64	Average price at which quantities were sold
`sell_value` float64	Net value of the sold quantities

attribute	
`sell_m2m` float64	Mark to market returns on the sold quantities
`day_sell_quantity` int64	Quantity sold off from the position during the day
`day_sell_price` float64	Average price at which quantities were sold during the day
`day_sell_value` float64	Net value of the quantities sold during the day

Position conversion

All positions held are of specific margin products such as NRML, MIS etc. A position can have one and only one margin product. These products affect how the user's margin usage and free cash values are computed, and a user may want to covert or change a position's margin product from time to time. More on margin policies.

```
curl --request PUT https://api.kite.trade/portfolio/positions
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "tradingsymbol=INFY" \
    -d "exchange=NSE" \
    -d "transaction_type=BUY" \
    -d "position_type=overnight" \
    -d "quantity=3" \
    -d "old_product=NRML" \
    -d "new_product=MIS"
```

```
{
    "status": "success",
    "data": true
}
```

Request parameters

parameter	
`tradingsymbol`	Tradingsymbol of the instrument
`exchange`	Name of the exchange
`transaction_type`	BUY or SELL
`position_type`	overnight or day
`quantity`	Quantity to convert
`old_product`	Existing margin product of the position
`new_product`	Margin product to convert to

Exiting holdings and positions

There are no special API calls for exiting instruments from holdings and positions portfolios. The way to do it is to place an opposite BUY or SELL order depending on whether the position is a long or a short (MARKET order for an immediate exit). It is important to note that the exit order should carry the same `product` as the existing position. If the exit order is of a different margin product, it may be treated as a new position in the portfolio.

Holdings authorisation

When executing sell transactions on equity holdings, where shares have to be debited from a user's demat account, a broker either requires a PoA (Power of Attorney) or an electronic authorisation at the depository from the user, to debit shares and settle the transactions.

Electronic authorisation happens centrally on the depostiory's portal (CDSL in Zerodha's case) when the user executing the sell transaction keys in their demat PIN, similar to a netbanking flow. The demat PIN is known only to the demat account holder and the depository, and not the broker.

In a single authorisation transaction, multiple shares with `n` quantities each, can be authorised. The authorisations are valid for a single trading session in a day (beginning of the day till 5:30 PM, after which, the authorisations are for the next trading day). The quantity in a sell transaction need not be the same as `n`, just that at any point, the total sell quantities, even over multiple days, should not exceed `n`. When it does, the order API throws an error asking for authorisation, at which point, the user has to be directed to the authorisation flow. To illustrate:

1. User has 50 quantity of INFY in their demat. There is no authorisation at this point.
2. User attempts to sell 10 quantity of INFY. The `POST /orders` API throws error "*10 quantity needs authorisation at depository.*" (`HTTP status 428`).
3. On encountering `428`, the authorisation flow is initiated and the user is redirected to the depository's portal. By default, Kite prompts the user to authorise the maximum quantities for every stock in their holding to avoid having to disrupt the sell transactions with the authorisation flow every time. In this case, 50 quantity of INFY is authorised.
4. User retries the transaction and 10 quantity is sold. 40 quantity remains authorised for the rest of the trading day (until 5:30 PM) and the user is not prompted for further authorisation until the remaining quantities have been sold.

Initiating authorisation

```
curl --request POST https://api.kite.trade/portfolio/holdings/authorise
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "isin=INE002A01018" -d "quantity=50" \
    -d "isin=INE009A01021" -d "quantity=50"
```

```
{
  "status": "success",
  "data": {
    "request_id": "na8QgCeQm05UHG6NL9sAGRzdfSF64UdB"
  }
}
```

The `isin` and `quantity` pairs here are optional. If they're provided, authorisation is sought only for those instruments and otherwise, the entire holdings is presented for authorisation. The `request_id` is then used to redirect the user to the following URL in a webivew or a popup.

`https://kite.zerodha.com/connect/portfolio/authorise/holdings/:api_key/:request_id`

After the user finishes the transaction, the webview is redirected to `/connect/portfolio/authorise/holdings/:api_key/:request_id/finish?status=success`.

Mobile applications can watch for this URL to detect the end of the transaction. `success` or `error` value in the `status` query param indicates the result.

Web applications can invoke this flow using the `authHoldings()` call in the Publisher Javascript plugin. It provides a callback event with the completion status along with additional metadata.

Market quotes and instruments

type	endpoint	
GET	/instruments	Retrieve the CSV dump of all tradable instruments
GET	/instruments/:exchange	Retrieve the CSV dump of instruments in the particular exchange
GET	/quote	Retrieve full market quotes for one or more instruments
GET	/quote/ohlc	Retrieve OHLC quotes for one or more instruments
GET	/quote/ltp	Retrieve LTP quotes for one or more instruments

---

Instruments

Between multiple exchanges and segments, there are tens of thousands of different kinds of instruments that trade. Any application that facilitates trading needs to have a master list of these instruments. The instruments API provides a consolidated, import-ready CSV list of instruments available for trading.

Retrieving the full instrument list

Unlike the rest of the calls that return JSON, the instrument list API returns a gzipped CSV dump of instruments across all exchanges that can be imported into a database. The dump is generated once everyday and hence last_price is not real time.

```
curl "https://api.kite.trade/instruments" \
  -H "X-Kite-Version: 3" \
  -H "Authorization: token api_key:access_token"
```

```
instrument_token, exchange_token, tradingsymbol, name, last_price, expiry, strike,
tick_size, lot_size, instrument_type, segment, exchange
408065,1594,INFY,INFOSYS,0,,,0.05,1,EQ,NSE,NSE
5720322,22345,NIFTY15DECFUT,,78.0,2015-12-31,,0.05,75,FUT,NFO-FUT,NFO
5720578,22346,NIFTY159500CE,,23.0,2015-12-31,9500,0.05,75,CE,NFO-OPT,NFO
645639,SILVER15DECFUT,,7800.0,2015-12-31,,1,1,FUT,MCX,MCX
```

CSV response columns

column	
`instrument_token` string	Numerical identifier used for subscribing to live market quotes with the WebSocket API.
`exchange_token` string	The numerical identifier issued by the exchange representing the instrument.
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`name` string	Name of the company (for equity instruments)
`last_price` float64	Last traded market price
`expiry` string	Expiry date (for derivatives)
`strike` float64	Strike (for options)
`tick_size` float64	Value of a single price tick
`lot_size` int64	Quantity of a single lot
`instrument_type` string	EQ, FUT, CE, PE

column	
`segment` string	Segment the instrument belongs to
`exchange` string	Exchange

Warning

The instrument list API returns large amounts of data. It's best to request it once a day (ideally at around 08:30 AM) and store in a database at your end.

Note

For storage, it is recommended to use a combination of exchange and tradingsymbol as the unique key, not the numeric instrument token. Exchanges may reuse instrument tokens for different derivative instruments after each expiry.

Market quotes

The market quotes APIs enable you to retrieve market data snapshots of various instruments. These are snapshots gathered from the exchanges at the time of the request. For realtime streaming market quotes, use the WebSocket API.

Retrieving full market quotes

This API returns the complete market data snapshot of up to **500** instruments in one go. It includes the quantity, OHLC, and Open Interest fields, and the complete bid/ask market depth amongst others.

Instruments are identified by the `exchange:tradingsymbol` combination and are passed as values to the query parameter `i` which is repeated for every instrument. If there is no data available for a

given key, the key will be absent from the response. The existence of all the instrument keys in the response map should be checked before to accessing them.

```
curl "https://api.kite.trade/quote?i=NSE:INFY" \
  -H "X-Kite-Version: 3" \
  -H "Authorization: token api_key:access_token"
```


```
{
    "status": "success",
    "data": {
      "NSE:INFY": {
        "instrument_token": 408065,
        "timestamp": "2021-06-08 15:45:56",
        "last_trade_time": "2021-06-08 15:45:52",
        "last_price": 1412.95,
        "last_quantity": 5,
        "buy_quantity": 0,
        "sell_quantity": 5191,
        "volume": 7360198,
        "average_price": 1412.47,
        "oi": 0,
        "oi_day_high": 0,
        "oi_day_low": 0,
        "net_change": 0,
        "lower_circuit_limit": 1250.7,
        "upper_circuit_limit": 1528.6,
        "ohlc": {
          "open": 1396,
          "high": 1421.75,
          "low": 1395.55,
          "close": 1389.65
        },
        "depth": {
          "buy": [
            {
              "price": 0,
              "quantity": 0,
              "orders": 0
            },
            {
              "price": 0,
              "quantity": 0,
              "orders": 0
            },
            {
              "price": 0,
              "quantity": 0,
              "orders": 0
            },
            {
              "price": 0,
```

```
            "quantity": 0,
            "orders": 0
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          }
        ],
        "sell": [
          {
            "price": 1412.95,
            "quantity": 5191,
            "orders": 13
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          },
          {
            "price": 0,
            "quantity": 0,
            "orders": 0
          }
        ]
      }
    }
  }
}
```


Response attributes

attribute	
`instrument_token` uint32	The numerical identifier issued by the exchange representing the instrument.

attribute	
`timestamp` string	The exchange timestamp of the quote packet
`last_trade_time` null, string	Last trade timestamp
`last_price` float64	Last traded market price
`volume` int64	Volume traded today
`average_price` float64	The volume weighted average price of a stock at a given time during the day?
`buy_quantity` int64	Total quantity of buy orders pending at the exchange
`sell_quantity` int64	Total quantity of sell orders pending at the exchange
`open_interest` float64	Total number of outstanding contracts held by market participants exchange-wide (only F&O)
`last_quantity` int64	Last traded quantity
`ohlc.open` float64	Price at market opening
`ohlc.high` float64	Highest price today
`ohlc.low` float64	Lowest price today
`ohlc.close` float64	Closing price of the instrument from the last trading day

attribute	
`net_change` float64	The absolute change from yesterday's close to last traded price
`lower_circuit_limit` float64	The current lower circuit limit
`upper_circuit_limit` float64	The current upper circuit limit
`oi` float64	The Open Interest for a futures or options contract ?
`oi_day_high` float64	The highest Open Interest recorded during the day
`oi_day_low` float64	The lowest Open Interest recorded during the day
`depth.buy[].price` float64	Price at which the depth stands
`depth.buy[].orders` int64	Number of open BUY (bid) orders at the price
`depth.buy[].quantity` int64	Net quantity from the pending orders
`depth.sell[].price` float64	Price at which the depth stands
`depth.sell[].orders` int64	Number of open SELL (ask) orders at the price
`depth.sell[].quantity` int64	Net quantity from the pending orders

Retrieving OHLC quotes

This API returns the OHLC + LTP snapshots of up to **1000** instruments in one go.

Instruments are identified by the `exchange:tradingsymbol` combination and are passed as values to the query parameter `i` which is repeated for every instrument. If there is no data available for a given key, the key will be absent from the response. The existence of all the instrument keys in the response map should be checked before to accessing them.

```
curl "https://api.kite.trade/quote/ohlc?i=NSE:INFY&i=BSE:SENSEX&i=NSE:NIFTY+50" \
    -H "X-Kite-Version: 3" \
  -H "Authorization: token api_key:access_token"
```


```
{
    "status": "success",
    "data": {
        "NSE:INFY": {
            "instrument_token": 408065,
            "last_price": 1075,
            "ohlc": {
                "open": 1085.8,
                "high": 1085.9,
                "low": 1070.9,
                "close": 1075.8
            }
        }
    }
}
```


Response attributes

attribute	
`instrument_token` uint32	The numerical identifier issued by the exchange representing the instrument.
`last_price` float64	Last traded market price
`ohlc.open` float64	Price at market opening

attribute	
`ohlc.high` float64	Highest price today
`ohlc.low` float64	Lowest price today
`ohlc.close` float64	Closing price of the instrument from the last trading day

Note

Always check for the existence of a particular key you've requested (eg: NSE:INFY) in the response. If there's no data for the particular instrument or if it has expired, the key will be missing from the response.

Retrieving LTP quotes

This API returns the LTPs of up to **1000** instruments in one go.

Instruments are identified by the `exchange:tradingsymbol` combination and are passed as values to the query parameter `i` which is repeated for every instrument. If there is no data available for a given key, the key will be absent from the response. The existence of all the instrument keys in the response map should be checked before to accessing them.

```
curl "https://api.kite.trade/quote/ltp?i=NSE:INFY&i=BSE:SENSEX&i=NSE:NIFTY+50" \
    -H "X-Kite-Version: 3" \
  -H "Authorization: token api_key:access_token"
```

```
{
    "status": "success",
    "data": {
        "NSE:INFY": {
            "instrument_token": 408065,
            "last_price": 1074.35
        }
    }
}
```

Response attributes

attribute	
`instrument_token` uint32	The numerical identifier issued by the exchange representing the instrument.
`last_price` float64	Last traded market price

> **Note**
>
> Always check for the existence of a particular key you've requested (eg: NSE:INFY) in the response. If there's no data for the particular instrument or if it has expired, the key will be absent from the response.

Limits

attribute	number of instruments
`/quote`	500
`/quote/ohlc`	1000
`/quote/ltp`	1000

WebSocket streaming

The WebSocket API is the most efficient (speed, latency, resource consumption, and bandwidth) way to receive quotes for instruments across all exchanges during live market hours. A quote consists of fields such as open, high, low, close, last traded price, 5 levels of bid/offer market depth data etc.

In addition, the text messages, alerts, and order updates (the same as the ones available as Postbacks) are also streamed. As the name suggests, the API uses WebSocket protocol to establish a single long standing TCP connection after an HTTP handshake to receive streaming quotes. To connect to the Kite WebSocket API, you will need a WebSocket client library in your choice of programming language.

You can subscribe for up to **3000** instruments on a single WebSocket connection and receive live quotes for them. Single API key can have upto 3 websocket connections.

> **Note**
>
> Implementing an asynchronous WebSocket client with a binary parser for the market data structure may be a complex task. We recommend using one of our pre-built client libraries.

Connecting to the WebSocket endpoint

```
// Javascript example.
var ws = new WebSocket("wss://ws.kite.trade?api_key=xxx&access_token=xxxx");
```

The WebSocket endpoint is `wss://ws.kite.trade`. To establish a connection, you have to pass two query parameters, `api_key` and `access_token`.

Request structure

```
// Subscribe to quotes for INFY (408065) and TATAMOTORS (884737)
var message = { a: "subscribe", v: [408065, 884737] };
ws.send(JSON.stringify(message));
```

Requests are simple JSON messages with two parameters, `a` (action) and `v` (value). Following are the available actions and possible values. Many values are arrays, for instance, array of `instrument_token` that can be passed to subscribe to multiple instruments at once.

a	v
`subscribe`	[`instrument_token` ...]
`unsubscribe`	[`instrument_token` ...]
`mode`	[`mode` , [`instrument_token` ...]]

```
// Set INFY (408065) to 'full' mode to
// receive market depth as well.
message = { a: "mode", v: ["full", [408065]] };
ws.send(JSON.stringify(message));

// Set TATAMOTORS (884737) to 'ltp' to only receive the LTP.
message = { a: "mode", v: ["ltp", [884737]] };
ws.send(JSON.stringify(message));
```

Modes

There are three different modes in which quote packets are streamed.

mode	
`ltp`	LTP. Packet contains only the last traded price (8 bytes).
`quote`	Quote. Packet contains several fields excluding market depth (44 bytes).
`full`	Full. Packet contains several fields including market depth (184 bytes).

Note

Always check the type of an incoming WebSocket messages. Market data is always binary and Postbacks and other updates are always text.

If there is no data to be streamed over an open WebSocket connection, the API will send a 1 byte "heartbeat" every couple seconds to keep the connection alive. This can be safely ignored.

Binary market data

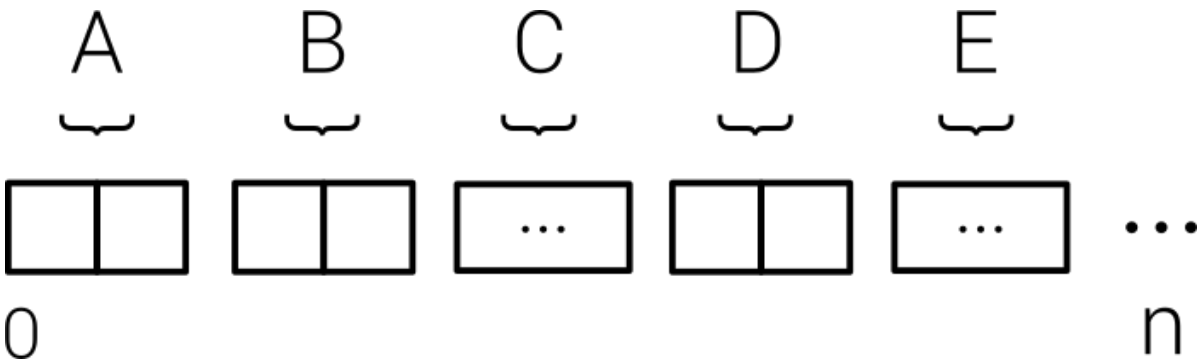
WebSocket supports two types of messages, binary and text.

Quotes delivered via the API are always binary messages. These have to be read as bytes and then type-casted into appropriate quote data structures. On the other hand, all requests you send to the API are JSON messages, and the API may also respond with non-quote, non-binary JSON messages, which are described in the next section.

For quote subscriptions, instruments are identified with their corresponding numerical `instrument_token` obtained from the instrument list API.

Message structure

Each binary message (array of `0` to `n` individual bytes)--or frame in WebSocket terminology--received via the WebSocket is a combination of one or more quote packets for one or more instruments. The message structure is as follows.



`A`	The first two bytes ([0 - 2] -- `SHORT` or `int16`) represent the number of packets in the message.
`B`	The next two bytes ([2 - 4] -- `SHORT` or `int16`) represent the length (number of bytes) of the first packet.
`C`	The next series of bytes ([4 - 4+B]) is the quote packet.
`D`	The next two bytes ([4+B - 4+B+2] -- `SHORT` or `int16`) represent the length (number of bytes) of the second packet.
`C`	The next series of bytes ([4+B+2 - 4+B+2+D]) is the next quote packet.

Quote packet structure

Each individual packet extracted from the message, based on the structure shown in the previous section, can be cast into a data structure as follows. All prices are in paise. For currencies, the `int32` price values should be divided by 10000000 to obtain four decimal plaes. For everything else, the price values should be divided by 100.

Bytes	Type	
0 - 4	`int32`	`instrument_token`
4 - 8	`int32`	Last traded price (*If mode is `ltp`, the packet ends here*)
8 - 12	`int32`	Last traded quantity
12 - 16	`int32`	Average traded price
16 - 20	`int32`	Volume traded for the day
20 - 24	`int32`	Total buy quantity

Bytes	Type	
24 - 28	`int32`	Total sell quantity
28 - 32	`int32`	Open price of the day
32 - 36	`int32`	High price of the day
36 - 40	`int32`	Low price of the day
40 - 44	`int32`	Close price (*If mode is `quote`, the packet ends here*)
44 - 48	`int32`	Last traded timestamp
48 - 52	`int32`	Open Interest
52 - 56	`int32`	Open Interest Day High
56 - 60	`int32`	Open Interest Day Low
60 - 64	`int32`	Exchange timestamp
64 - 184	`[]byte`	Market depth entries

Index packet structure

The packet structure for indices such as `NIFTY 50` and `SENSEX` differ from that of tradeable instruments. They have fewer fields.

Bytes	Type	
0 - 4	`int32`	Token
4 - 8	`int32`	Last traded price
8 - 12	`int32`	High of the day

Bytes	Type	
12 - 16	`int32`	Low of the day
16 - 20	`int32`	Open of the day
20 - 24	`int32`	Close of the day
24 - 28	`int32`	Price change (*If mode is `quote`, the packet ends here*)
28 - 32	`int32`	Exchange timestamp

Market depth structure

Each market depth entry is a combination of 3 fields, `quantity` (`int32`), `price` (`int32`), `orders` (`int16`) and there is a 2 byte padding at the end (which should be skipped) totalling to 12 bytes. There are ten entries in succession—five [`64 - 124`] bid entries and five [`124 - 184`] offer entries.

Postbacks and non-binary updates

Apart from binary market data, the WebSocket stream delivers postbacks and other updates in the text mode. These messages are JSON encoded and should be parsed on receipt. For order Postbacks, the payload is contained in the `data` key and has the same structure described in the Postbacks section.

> Message structure

```
{
  "type": "order",
  "data": {}
}
```

Message types

type	
`order`	Order Postback. The `data` field will contain the full order Postback payload
`error`	Error responses. The `data` field contain the error string
`message`	Messages and alerts from the broker. The `data` field will contain the message string

Historical candle data

The historical data API provides archived data (up to date as of the time of access) for instruments across various exchanges spanning back several years. A historical record is presented in the form of a *candle* (Timestamp, Open, High, Low, Close, Volume, OI), and the data is available in several intervals—minute, 3 minutes, 5 minutes, hourly ... daily.

type	endpoint	
GET	/instruments/historical/:instrument_token/:interval	Retrieve historical candle records for a given instrument.

URI parameters

parameter	
`:instrument_token`	Identifier for the instrument whose historical records you want to fetch. This is obtained with the instrument list API.
`:interval`	The candle record interval. Possible values are: · `minute` · `day` · `3minute` · `5minute` · `10minute` · `15minute` · `30minute` · `60minute`

Request parameters

parameter	
`from`	`yyyy-mm-dd hh:mm:ss` formatted date indicating the start date of records
`to`	`yyyy-mm-dd hh:mm:ss` formatted date indicating the end date of records
`continuous`	Accepts `0` or `1`. Pass `1` to get continuous data
`oi`	Accepts `0` or `1`. Pass `1` to get OI data

Response structure

The response is an array of records, where each record in turn is an array of the following values — `[timestamp, open, high, low, close, volume]`.

> **Note**
>
> It is possible to retrieve candles for small time intervals by making the `from` and `to` calls granular. For instance `from = 2017-01-01 09:15:00` and `to = 2017-01-01 09:30:00` to fetch candles for just 15 minutes between those timestamps.

Continuous data

It's important to note that the exchanges flush the `instrument_token` for futures and options contracts for every expiry. For instance, `NIFTYJAN18FUT` and `NIFTYFEB18FUT` will have different instrument tokens although their underlying contract is the same. The instrument master API only returns instrument_tokens for contracts that are live. It is not possible to retrieve instrument_tokens for expired contracts from the API, unless you regularly download and cache them.

This is where `continuous` API comes in which works for NFO and MCX futures contracts. Given a live contract's `instrument_token`, the API will return `day` candle records for the same instrument's expired contracts. For instance, assuming the current month is January and you pass `NIFTYJAN18FUT`'s `instrument_token` along with `continuous=1`, you can fetch day candles for December, November ... contracts by simply changing the `from` and `to` dates.

Examples

```
# Fetch minute candles for NSE-ACC.
# This will return several days of minute data ending today.
# The time of request is assumed to be to be 01:30 PM, 1 Jan 2016,
# which is reflected in the latest (last) record.

# The data has been truncated with ... in the example responses.

curl "https://api.kite.trade/instruments/historical/5633/minute?from=2017-12-
15+09:15:00&to=2017-12-15+09:20:00"
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```


```
{
  "status": "success",
  "data": {
    "candles": [
      [
        "2017-12-15T09:15:00+0530",
        1704.5,
        1705,
        1699.25,
        1702.8,
        2499
      ],
      [
        "2017-12-15T09:16:00+0530",
        1702,
        1702,
        1698.15,
        1698.15,
        1271
      ],
      [
        "2017-12-15T09:17:00+0530",
        1698.15,
        1700.25,
        1698,
        1699.25,
        831
      ],
      [
        "2017-12-15T09:18:00+0530",
        1700,
        1700,
        1698.3,
        1699,
        771
      ],
```

```
      [
        "2017-12-15T09:19:00+0530",
        1699,
        1700,
        1698.1,
        1699.8,
        543
      ],
      [
        "2017-12-15T09:20:00+0530",
        1699.8,
        1700,
        1696.55,
        1696.9,
        802
      ]
    ]
  }
}
```

OI Data

```
# Fetch minute candles for NIFTY19DECFUT for five minutes with OI data
curl "https://api.kite.trade/instruments/historical/12517890/minute?from=2019-12-04%2009:15:00&to=2019-12-04%2009:20:00&oi=1" \
     -H 'X-Kite-Version: 3' \
     -H 'Authorization: token api_key:access_token'
```

```
{
  "status": "success",
  "data": {
    "candles": [
      [
        "2019-12-04T09:15:00+0530",
        12009.9,
        12019.35,
        12001.25,
        12001.5,
        163275,
        13667775
      ],
      [
        "2019-12-04T09:16:00+0530",
        12001,
        12003,
        11998.25,
        12001,
        105750,
        13667775
      ],
```

```
      [
        "2019-12-04T09:17:00+0530",
        12001,
        12001,
        11995.1,
        11998.55,
        48450,
        13758000
      ],
      [
        "2019-12-04T09:18:00+0530",
        11997.8,
        12002,
        11996.25,
        12001.55,
        52875,
        13758000
      ],
      [
        "2019-12-04T09:19:00+0530",
        12002.35,
        12007,
        12001.45,
        12007,
        52200,
        13758000
      ],
      [
        "2019-12-04T09:20:00+0530",
        12006.95,
        12009.25,
        11999.6,
        11999.6,
        65325,
        13777050
      ]
    ]
  }
}
```

Postback (WebHooks)

The Postback API sends a `POST` request with a JSON payload to the registered `postback_url` of your app when an order's status changes. This enables you to get arbitrary updates to your orders reliably, irrespective of when they happen (`COMPLETE` , `CANCEL` , `REJECTED` , `UPDATE`). An `UPDATE` postback is triggered when an open order is modified or when there's a partial fill. This can be used to track trades.

> **Note**
>
> This Postback API is meant for platforms and public apps where a single api_key will place orders for multiple users. Only orders placed using the app's `api_key` are notified.
>
> For individual developers, Postbacks over WebSocket is recommended, where, orders placed for a particular user anywhere, for instance, web, mobile, or desktop platforms, are sent.

The JSON payload is posted as a raw HTTP POST body. You will have to read the raw body and then decode it.

> Sample payload


```
{
    "user_id": "AB1234",
    "unfilled_quantity": 0,
    "app_id": 1234,
    "checksum":
"2011845d9348bd6795151bf4258102a03431e3bb12a79c0df73fcb4b7fde4b5d",
    "placed_by": "AB1234",
    "order_id": "220303000308932",
    "exchange_order_id": "1000000001482421",
    "parent_order_id": null,
    "status": "COMPLETE",
    "status_message": null,
    "status_message_raw": null,
    "order_timestamp": "2022-03-03 09:24:25",
    "exchange_update_timestamp": "2022-03-03 09:24:25",
    "exchange_timestamp": "2022-03-03 09:24:25",
    "variety": "regular",
    "exchange": "NSE",
    "tradingsymbol": "SBIN",
    "instrument_token": 779521,
    "order_type": "MARKET",
```

```
    "transaction_type": "BUY",
    "validity": "DAY",
    "product": "CNC",
    "quantity": 1,
    "disclosed_quantity": 0,
    "price": 0,
    "trigger_price": 0,
    "average_price": 470,
    "filled_quantity": 1,
    "pending_quantity": 0,
    "cancelled_quantity": 0,
    "market_protection": 0,
    "meta": {},
    "tag": null,
    "guid": "XXXXXX"
}
```


Checksum

The JSON payload comes with a `checksum`, which is the SHA-256 hash of (`order_id` + `order_timestamp` + `api_secret`). For every Postback you receive, you should compute this checksum at your end and match it with the checksum in the payload. This is to ensure that the update is being POSTed by Kite Connect and not by an unauthorised entity, as only Kite Connect can generate a checksum that contains your `api_secret`.

Payload attributes

attribute	
`order_id` string	Unique order ID
`exchange_order_id` null, string	Exchange generated order id. Orders that don't reach the exchange have null ids
`parent_order_id` null, string	Order ID of the parent order (only applicable in case of multi-legged orders like CO)
`placed_by` string	ID of the user that placed the order. This may different from the user's id for orders placed outside of Kite, for instance, by dealers at the brokerage using dealer terminals.

attribute	
`app_id` int64	Your kiteconnect app ID
`status` null, string	Current status of the order. The possible values are COMPLETE, REJECTED, CANCELLED, and UPDATE.
`status_message` null, string	Textual description of the order's status. Failed orders come with human readable explanation
`status_message_raw` null, string	Raw textual description of the failed order's status, as received from the OMS
`tradingsymbol` string	Exchange tradingsymbol of the of the instrument
`instrument_token` uint32	The numerical identifier issued by the exchange representing the instrument
`exchange` string	Exchange
`order_type` string	Order type (MARKET, LIMIT etc.)
`transaction_type` string	BUY or SELL
`validity` string	Order validity
`variety` string	Order variety (regular, amo, co etc.)
`product` string	Margin product to use for the order

attribute	
`average_price` float64	Average price at which the order was executed (only for COMPLETE orders)
`disclosed_quantity` int64	Quantity to be disclosed (may be different from actual quantity) to the public exchange orderbook. Only for equities
`price` float64	Price at which the order was placed (LIMIT orders)
`quantity` int64	Quantity ordered
`filled_quantity` int64	Quantity that has been filled
`unfilled_quantity` int64	Quantity that has not filled
`pending_quantity` int64	Pending quantity for open order
`cancelled_quantity` int64	Quantity that had been cancelled
`trigger_price` float64	Trigger price (for SL, SL-M, CO orders)
`user_id` string	ID of the user for whom the order was placed.
`order_timestamp` string	Timestamp at which the order was registered by the API
`exchange_update_timestamp` string	Timestamp at which an order's state changed at the exchange

attribute	
`exchange_timestamp` string	Timestamp at which the order was registered by the exchange. Orders that don't reach the exchange have null timestamps
`checksum` string	SHA-256 hash of (order_id + timestamp + api_secret)
`meta` {}, string	Map of arbitrary fields that the system may attach to an order
`tag` null, string	An optional tag to apply to an order to identify it (alphanumeric, max 20 chars)

Note

Postback API works even when the user is not logged in. Just make sure you validate the checksum value to ensure that the update is indeed coming from Kite Connect.

Mutual funds

The mutual fund APIs allow buying, selling, and managing SIPs of mutual funds listed on Zerodha's Coin platform, where successful purchases are delivered to the buyer's DEMAT account. The APIs are built on top of the BSE STARMF platform. The mutual fund APIs have been grouped here under a separate section as they differ in structure, although not substantially, from the other trading APIs.

type	endpoint	
POST	/mf/orders	Place a buy or a sell order
DELETE	/mf/orders/:order_id	Cancel an open or pending order
GET	/mf/orders	Retrieve the list of all orders (open and executed) over the last 7 days
GET	/mf/orders/:order_id	Retrieve an individual order
POST	/mf/sips	Place a SIP order
PUT	/mf/sips/:order_id	Modify an open SIP order
DELETE	/mf/sips/:order_id	Cancel an open SIP order
GET	/mf/sips/	Retrieve the list of all open SIP orders
GET	/mf/sips/:order_id	Retrieve an individual SIP order
GET	/mf/holdings	Retrieve the list of mutual fund holdings available in the DEMAT
GET	/mf/instruments	Retrieve the master list of all mutual funds available on the platform

Note

Dividend reinvestment schemes are currently not supported.

Placing orders

Unlike stock APIs, mutual fund orders are not sent to the exchange immediately, but at a preset times, in batches, every day. More information can be found on Coin. A placed order can be cancelled and placed again any number of times before its cut-off time. To modify an existing order for a particular fund, you just have to place a new order, as it'll overwrite the existing order.

Note

Mutual fund orders are not sent to the exchange immediately. They're collected, batched, and sent to the exchange at 01:30 PM on market days.

```
curl "https://api.kite.trade/mf/orders" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "tradingsymbol=INF174K01LS2" \
    -d "transaction_type=BUY" \
    -d "amount=1000"
```

```
{
  "status": "success",
  "data": {
    "order_id": "3bb085d1-5038-450e-a807-6543fef6c9ae"
  }
}
```

Order parameters

parameter	
`tradingsymbol` string	Tradingsymbol (ISIN) of the fund

parameter	
`transaction_type` string	BUY or SELL
`quantity` float64	Quantity to SELL. Not applicable on BUYs. If the holding is less than `minimum_redemption_quantity`, all the units have to be sold
`amount` float64	Amount worth of units to purchase. Not applicable on SELLs
`tag` string	An optional tag to apply to an order to identify it (alphanumeric, max 8 chars)

Response attributes

attribute	
`order_id` string	Unique order id

Cancelling orders

As long as an order is open in the system, it can be cancelled.

```
curl --request DELETE \
    "https://api.kite.trade/mf/orders/123456" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```

```
{
  "status": "success",
  "data": {
    "order_id": "3bb085d1-5038-450e-a807-6543fef6c9ae"
  }
}
```

Response attributes

attribute	
`order_id` string	Unique order id

Retrieving orders

This API returns all orders placed in the last 7 days.

```
curl "https://api.kite.trade/mf/orders" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token"
```


```
{
  "status": "success",
  "data": [
    {
      "status": "REJECTED",
      "purchase_type": "FRESH",
      "folio": null,
      "order_timestamp": "2021-06-30 08:33:07",
      "average_price": 0.0,
      "exchange_order_id": "254657127",
      "last_price": 30.6800,
      "tradingsymbol": "INF179K01VY8",
      "settlement_id": "2122061",
      "transaction_type": "BUY",
      "order_id": "271989e0-a64e-4cf3-b4e4-afb8f38dd203",
      "amount": 1000.0,
      "tag": null,
      "placed_by": "ZV8062",
      "exchange_timestamp": "2021-06-30",
      "variety": "amc_sip",
      "last_price_date": "2021-06-29",
      "status_message": "AMC SIP: Insufficient balance.",
      "fund": "HDFC Balanced Advantage Fund - Direct Plan",
      "quantity": 0.0
    },
    {
      "status": "REJECTED",
      "purchase_type": "ADDITIONAL",
      "folio": null,
      "order_timestamp": "2021-06-30 01:30:02",
```

```
    "average_price": 0.0,
    "exchange_order_id": null,
    "last_price": 52.7980,
    "tradingsymbol": "INF174K01LS2",
    "settlement_id": null,
    "transaction_type": "BUY",
    "order_id": "ef7e696c-2fa6-400b-b180-eb25e6a04ccf",
    "amount": 2000.0,
    "tag": "coinandroidsip",
    "placed_by": "ZV8062",
    "exchange_timestamp": null,
    "variety": "sip",
    "last_price_date": "2021-06-29",
    "status_message": "SIP: Insufficient balance.",
    "fund": "Kotak Flexicap Fund - Direct Plan",
    "quantity": 0.0
  },
  {
    "status": "OPEN",
    "purchase_type": "FRESH",
    "folio": null,
    "order_timestamp": "2021-06-29 12:20:28",
    "average_price": 0.0,
    "exchange_order_id": null,
    "last_price": 10.4324,
    "tradingsymbol": "INF761K01EE1",
    "settlement_id": null,
    "transaction_type": "BUY",
    "order_id": "2b6ad4b7-c84e-4c76-b459-f3a8994184f1",
    "amount": 5000.0,
    "tag": null,
    "placed_by": "ZV8062",
    "exchange_timestamp": null,
    "variety": "regular",
    "last_price_date": "2021-06-29",
    "status_message": "Insufficient fund. 1/5",
    "fund": "BOI AXA Arbitrage Fund - Direct Plan",
    "quantity": 0.0
  },
  {
    "status": "REJECTED",
    "purchase_type": "FRESH",
    "folio": null,
    "order_timestamp": "2021-06-29 08:36:41",
    "average_price": 0.0,
    "exchange_order_id": "254447867",
    "last_price": 271.7500,
    "tradingsymbol": "INF179K01WA6",
    "settlement_id": "2122060",
    "transaction_type": "BUY",
    "order_id": "40410882-b1f8-4938-bb08-4bef2765cbfb",
    "amount": 1000.0,
    "tag": null,
```

```
      "placed_by": "ZV8062",
      "exchange_timestamp": "2021-06-29",
      "variety": "amc_sip",
      "last_price_date": "2021-06-29",
      "status_message": "AMC SIP: Insufficient balance.",
      "fund": "HDFC Balanced Advantage Fund - Direct Plan",
      "quantity": 0.0
    },
    {
      "status": "OPEN",
      "purchase_type": "FRESH",
      "folio": null,
      "order_timestamp": "2021-06-24 15:37:27",
      "average_price": 0.0,
      "exchange_order_id": null,
      "last_price": 11.5182,
      "tradingsymbol": "INF109K01V59",
      "settlement_id": null,
      "transaction_type": "BUY",
      "order_id": "e67b8741-5054-4fd5-a2da-8c672e1f494a",
      "amount": 5000.0,
      "tag": null,
      "placed_by": "ZV8062",
      "exchange_timestamp": null,
      "variety": "regular",
      "last_price_date": "2021-06-29",
      "status_message": "Insufficient fund. 3/5",
      "fund": "ICICI Prudential Bond Fund - Direct Plan",
      "quantity": 0.0
    }
  ]
}
```


Response attributes

attribute	
`order_id` string	Unique order id
`exchange_order_id` null, string	Exchange generated order id
`tradingsymbol` string	ISIN of the fund

attribute	
`status` null, string	Current status of the order. Most common values or COMPLETE, REJECTED, CANCELLED, and OPEN. There may be other values as well
`status_message` null, string	Textual description of the order's status. Failed orders come with human readable explanation
`folio` null, string	Folio number generated by AMC for the completed purchase order
`fund` string	Name of the fund
`order_timestamp` string	Timestamp at which the order was registered by the API
`exchange_timestamp` string	Date on which the order was registered by the exchange. Orders that don't reach the exchange have null timestamps
`settlement_id` string	Exchange settlement ID
`transaction_type` string	BUY or SELL
`amount` float64	Amount placed for purchase of units
`variety` string	Order variety (regular, sip)
`purchase_type` null, string	FRESH or ADDITIONAL (null incase of SELL order)
`quantity` float64	Number of units allotted or sold

attribute	
`price` float64	Buy or sell price
`last_price` float64	Last available NAV price of the fund
`average_price` float64	Allotted or sold NAV price
`placed_by` string	Id of the user that placed the order
`last_price_date` string	Date for which last NAV is available
`tag` string	Tag that was sent with an order to identify it (alphanumeric, max 8 chars)

Retrieving an individual order

While the orders list API returns orders within the last 7 days, given an order ID, this API will return the order details irrespective of its age.

```
curl  "https://api.kite.trade/mf/orders/123123"
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```


```
{
  "status": "success",
  "data": {
    "status": "OPEN",
    "purchase_type": "FRESH",
    "exchange_order_id": null,
    "last_price": 10.4324,
    "order_timestamp": "2021-06-29 12:20:28",
    "fund": "BOI AXA Arbitrage Fund - Direct Plan",
    "tradingsymbol": "INF761K01EE1",
    "tag": null,
```

```
    "placed_by": "ZV8062",
    "last_price_date": "2021-06-29",
    "folio": null,
    "variety": "regular",
    "exchange_timestamp": null,
    "average_price": 0.0,
    "settlement_id": null,
    "transaction_type": "BUY",
    "order_id": "2b6ad4b7-c84e-4c76-b459-f3a8994184f1",
    "amount": 5000.0,
    "status_message": "Insufficient fund. 1/5",
    "quantity": 0
  }
}
```

Placing SIP orders

SIP (Systematic Investment Plan) orders persist in the system once created and send orders to the exchange at recurring intervals. For instance, buying 5000 worth of units of a particular fund on the first of every month. It's important to note that prior to creating a SIP on an account, there has to have been at least one investment in the target fund. If not, an order worth `initial_amount` is created automatically along with the SIP.

```
curl "https://api.kite.trade/mf/sips" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
    -d "tradingsymbol=INF174K01LS2" \
    -d "frequency=monthly" \
    -d "instalment_day=1" \
    -d "instalments=-1" \
    -d "initial_amount=5000" \
    -d "amount=1000"
```


```
{
  "status": "success",
  "data": {
    "sip_id": "986124545877922"
  }
}
```

parameter	
`tradingsymbol` string	ISIN of the fund
`amount` float64	Amount worth of units to purchase. It should be equal to or greater than `minimum_additional_purchase_amount` and in multiple of `purchase_amount_multiplier` in the instrument master.
`initial_amount` float64	Amount worth of units to purchase before the SIP starts. Should be equal to or greater than `minimum_purchase_amount` and in multiple of `purchase_amount_multiplier`. This is only considered if there have been no prior investments in the target fund.
`frequency` string	weekly, monthly, or quarterly
`instalment_day` int64	If `frequency` is monthly, the day of the month (1, 5, 10, 15, 20, 25) to trigger the order on
`instalments` int64	Number of instalments to trigger. If set to -1, instalments are triggered at fixed intervals until the SIP is cancelled
`tag` string	An optional tag to apply to an order to identify it (alphanumeric, max 8 chars)

Response attributes

attribute	
`sip_id` string	Unique SIP id

Modifying SIPs

Unlike SIPs with a regular AMC, SIPs on Coin can be modified at any point of time. For instance, they can be created to invest INR 5000 every month, but can be modified to invest INR 1500 weekly.

```
curl --request PUT \
    "https://api.kite.trade/mf/sip/1234" \
      -H "X-Kite-Version: 3" \
      -H "Authorization: token api_key:access_token" \
      -d "frequency=monthly" \
      -d "instalments=10" \
      -d "amount=1000" \
      -d "status=paused" \
      -d "instalment_day=1"
```

```
{
  "status": "success",
  "data": {
    "sip_id": "986124545877922"
  }
}
```

parameter	
`amount` float64	Amount worth of units to purchase. It should be equal to or greater than `minimum_additional_purchase_amount` and in multiple of `purchase_amount_multiplier` in the instrument master.
`frequency` string	weekly, monthly, or quarterly
`instalment_day` int64	If `frequency` is monthly, the day of the month (1, 5, 10, 15, 20, 25) to trigger the order on
`instalments` int64	Number of instalments to trigger. If set to -1, instalments are triggered indefinitely until the SIP is cancelled
`status` string	Pause or unpause an SIP (`active` or `paused`)

Response attributes

attribute	
`order_id` string	Unique order id

Cancelling SIPs

A SIP order can be cancelled at any time, as long as it hasn't self-cancelled based on the `instalments` param.

> **Note**
>
> There has to have been at least one investment in a fund prior to starting an SIP. This can be handled with the `initial_amount` field.

```
curl --request DELETE \
    "https://api.kite.trade/mf/sips/123456" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```

```
{
  "status": "success",
  "data": {
    "sip_id": "986124545877922"
  }
}
```

Response attributes

attribute	
`sip_id` string	Unique SIP id

Retrieving all SIPs

This API returns the list of all active and paused SIPs

```
curl "https://api.kite.trade/mf/sips"
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```


```
{
"data": [
  {
    "status": "ACTIVE",
    "sip_reg_num": null,
    "created": "2021-05-05 05:56:27",
    "dividend_type": "idcw",
    "instalment_amount": 500.0,
    "fund": "Aditya Birla Sun Life Liquid Fund - Direct Plan",
    "instalments": -1,
    "next_instalment": "2021-05-12",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "step_up": {
      "05-05": 10
    },
    "tradingsymbol": "INF209K01VD7",
    "tag": "coiniossip",
    "frequency": "weekly",
    "last_instalment": "2021-05-05 05:56:27",
    "pending_instalments": -1,
    "instalment_day": 0,
    "sip_type": "sip",
    "completed_instalments": 0,
    "sip_id": "892741486820670"
  },
  {
    "status": "ACTIVE",
    "sip_reg_num": null,
    "created": "2021-05-25 10:55:09",
    "dividend_type": "idcw",
    "instalment_amount": 1000.0,
    "fund": "HDFC Balanced Advantage Fund - Direct Plan",
    "instalments": -1,
    "next_instalment": "2021-06-01",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "step_up": {
      "25-05": 10
    },
    "tradingsymbol": "INF179K01VY8",
```

```
    "tag": "coiniossip",
    "frequency": "weekly",
    "last_instalment": "2021-05-25 10:55:09",
    "pending_instalments": -1,
    "instalment_day": 0,
    "sip_type": "sip",
    "completed_instalments": 0,
    "sip_id": "109195857904698"
  },
  {
    "status": "ACTIVE",
    "sip_reg_num": "15158182",
    "created": "2021-05-22 10:45:29",
    "dividend_type": "idcw",
    "instalment_amount": 1000.0,
    "fund": "HDFC Balanced Advantage Fund - Direct Plan",
    "instalments": 9999,
    "next_instalment": "2021-07-12",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "step_up": {},
    "tradingsymbol": "INF179K01VY8",
    "tag": "coinandroidsip",
    "frequency": "monthly",
    "last_instalment": "2021-06-10 08:37:11",
    "pending_instalments": 9998,
    "instalment_day": 10,
    "sip_type": "amc_sip",
    "completed_instalments": 1,
    "sip_id": "846479755969168"
  },
  {
    "status": "ACTIVE",
    "sip_reg_num": "16055666",
    "created": "2021-06-18 03:56:46",
    "dividend_type": "idcw",
    "instalment_amount": 1000.0,
    "fund": "HDFC Balanced Advantage Fund - Direct Plan",
    "instalments": 9999,
    "next_instalment": "2021-07-30",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "step_up": {},
    "tradingsymbol": "INF179K01VY8",
    "tag": "coinandroidsip",
    "frequency": "monthly",
    "last_instalment": "2021-06-30 08:33:07",
    "pending_instalments": 9998,
    "instalment_day": 30,
    "sip_type": "amc_sip",
    "completed_instalments": 1,
    "sip_id": "749073272501476"
  },
```

```
  {
    "status": "ACTIVE",
    "sip_reg_num": null,
    "created": "2020-11-20 01:06:11",
    "dividend_type": "growth",
    "instalment_amount": 7427.0,
    "fund": "HDFC Hybrid Equity Fund - Direct Plan",
    "instalments": -1,
    "next_instalment": "2021-02-19",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "step_up": {
      "20-11": 30
    },
    "tradingsymbol": "INF179K01XZ1",
    "tag": "coinandroidsip",
    "frequency": "quarterly",
    "last_instalment": "2020-11-20 01:06:11",
    "pending_instalments": -1,
    "instalment_day": 0,
    "sip_type": "sip",
    "completed_instalments": 0,
    "sip_id": "576440634181776"
  }]
}
```


Response attributes

attribute	
`sip_id` string	Unique SIP id
`tradingsymbol` string	ISIN of the fund.
`fund` string	Name of the fund
`dividend_type` string	Dividend type (growth, payout)
`transaction_type` string	BUY or SELL

attribute	
`status` string	ACTIVE, PAUSED or CANCELLED
`created` string	Timestamp at which the SIP was registered by the API
`frequency` string	Frequency at which order is triggered (monthly, weekly, or quarterly)
`next_instalment` string	Upcoming instalment date
`instalment_amount` int64	Amount worth of units to purchase in each instalment
`instalments` int64	Number of instalments (-1 in case of SIPs active until cancelled)
`last_instalment` string	Timestamp at which the last instalment was triggered
`pending_instalments` int64	Number of instalments pending (-1 in case of SIPs active until cancelled)
`instalment_day` int64	Calendar day in a month on which SIP order to be triggered (valid only incase of frequency monthly, else 0)
`completed_instalments` int64	Total number of completed instalments from the start
`tag` string	Tag that was sent with an order to identify it (alphanumeric, max 8 chars)

Retrieving an individual SIP

Given a SIP id, this call returns its details. This also works for deleted SIPs that are not included in the SIP list.

```
curl "https://api.kite.trade/mf/sips/1234" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```


```
{
  "status": "success",
  "data": {
    "status": "ACTIVE",
    "sip_reg_num": null,
    "pending_instalments": -1,
    "created": "2022-02-15 13:39:44",
    "step_up": {
      "15-02": 10
    },
    "dividend_type": "growth",
    "instalment_amount": 5000.00,
    "tag": "coiniossip",
    "instalments": -1,
    "next_instalment": "2022-02-22",
    "transaction_type": "BUY",
    "trigger_price": 0,
    "fund": "Aditya Birla Sun Life Overnight Fund - Direct Plan",
    "tradingsymbol": "INF209KB1ZH2",
    "frequency": "weekly",
    "last_instalment": "2022-02-15 13:39:44",
    "instalment_day": 0,
    "sip_type": "sip",
    "fund_source": "pool",
    "completed_instalments": 0,
    "sip_id": "181635213661372"
  }
}
```

Holdings

Holdings contain the user's portfolio of allotted mutual fund units.

```
curl "https://api.kite.trade/mf/holdings" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```


```
{
  "status": "success",
```

```
  "data": [
    {
      "folio": "3108290884",
      "average_price": 78.43,
      "last_price": 84.86,
      "last_price_date": "",
      "pledged_quantity": 0,
      "fund": "INVESCO INDIA TAX PLAN - DIRECT PLAN",
      "tradingsymbol": "INF205K01NT8",
      "pnl": 0,
      "quantity": 382.488
    },
    {
      "folio": "5102495241",
      "average_price": 1874.101138,
      "last_price": 2081.4984,
      "last_price_date": "",
      "pledged_quantity": 0,
      "fund": "Indiabulls Liquid Fund - Direct Plan",
      "tradingsymbol": "INF666M01451",
      "pnl": 0,
      "quantity": 1.334
    },
    {
      "folio": "9104386836",
      "average_price": 116.7,
      "last_price": 101.13,
      "last_price_date": "",
      "pledged_quantity": 0,
      "fund": "BOI AXA TAX ADVANTAGE FUND - DIRECT PLAN",
      "tradingsymbol": "INF761K01884",
      "pnl": 0,
      "quantity": 257.057
    }
  ]
}
```


Response attributes

attribute	
`folio` null, string	Folio number generated by AMC for the completed purchase order (null incase of SELL order)
`fund` string	Name of the fund

attribute	
`tradingsymbol` string	ISIN of the fund.
`average_price` float64	Allotted NAV price for a completed BUY order; Selling NAV price for completed SELL order
`last_price` float64	Last available NAV price of the fund
`pnl` float64	Net returns of the holding. Based on the last available NAV price.
`last_price_date` string	Date for which last NAV is available
`quantity` float64	Quantity available in the client's holding for this ISIN.

Retrieving the full instrument list

Unlike the rest of the calls that return JSON, the instrument list API returns a Gzipped CSV dump of mutual funds supported by Zerodha's Coin platform.

```
curl "https://api.kite.trade/mf/instruments" \
    -H "X-Kite-Version: 3" \
    -H "Authorization: token api_key:access_token" \
```

```
tradingsymbol,amc,name,purchase_allowed,redemption_allowed,minimum_purchase_amount,p
INF846K01DP8,AXISMUTUALFUND_MF,Axis Equity Fund - Direct Plan -
Growth,1,1,5000.0,1.0,100.0,1.0,0.001,growth,equity,direct,T3,20.09,2016-11-11
INF846K01EW2,AXISMUTUALFUND_MF,Axis Long Term Equity Fund - Direct
Growth,1,1,500.0,500.0,500.0,1.0,0.001,growth,elss,direct,T3,33.0425,2016-11-11
INF174K01LS2,KOTAKMAHINDRAMF,Kotak Select Focus Fund- Direct Plan -
Growth,1,1,5000.0,1.0,1000.0,0.001,0.001,growth,equity,direct,T3,26.549,2016-11-11
INF174K01336,KOTAKMAHINDRAMF,Kotak Select Focus Fund-
Growth,1,1,5000.0,0.01,1000.0,0.001,0.001,growth,equity,regular,T3,25.635,2016-11-
11
```

Response columns

column	
`tradingsymbol` string	ISIN of the fund
`amc` string	AMC code as per the exchange
`name` string	Fund name
`purchase_allowed` string	`0` or `1`
`redemption_allowed` string	`0` or `1`
`minimum_purchase_amount` float64	Minimum purchase amount for the first BUY
`purchase_amount_multiplier` float64	Buy amount should be in multiple of this value
`minimum_additional_purchase_amount` float64	Minimum additional BUY amount
`minimum_redemption_quantity` float64	Minimum SELL quantity
`redemption_quantity_multiplier` float64	SELL quantity multiple
`dividend_type` string	`growth` or `payout`
`scheme_type` string	`equity`, `debt`, `elss`

column	
`plan` string	`direct` or `regular`
`settlement_type` string	Settlement type of the fund (`T1` , `T2` etc.)
`last_price` float64	Last available NAV price of the fund
`last_price_date` string	Last available NAV's date

Margin calculation

Margin calculation APIs lets you calculate `span`, `exposure`, `option premium`, `additional`, `bo`, `cash`, `var`, `pnl` values for a list of orders.

type	endpoint	
POST	/margins/orders	Calculates margins for each order considering the existing positions and open orders
POST	/margins/basket	Calculates margins for spread orders
POST	/charges/orders	Calculates order-wise charges for orderbook

Note

Requests to the above endpoints are JSON POST and it needs `application/json` header.

Order margins

Request order structure
Response margin structure

```
curl https://api.kite.trade/margins/orders \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
    -H 'Content-Type: application/json' \
    -d '[
    {
        "exchange": "NSE",
        "tradingsymbol": "INFY",
        "transaction_type": "BUY",
        "variety": "regular",
        "product": "CNC",
        "order_type": "MARKET",
        "quantity": 1,
```

```
        "price": 0,
        "trigger_price": 0
    }
]'
```

Query parameters are as follows.

parameter	
mode	`compact` - Compact mode will only give the total margins


```
{
  "status": "success",
  "data": [
    {
      "type": "equity",
      "tradingsymbol": "INFY",
      "exchange": "NSE",
      "span": 0,
      "exposure": 0,
      "option_premium": 0,
      "additional": 0,
      "bo": 0,
      "cash": 0,
      "var": 1498,
      "pnl": {
        "realised": 0,
        "unrealised": 0
      },
      "leverage": 1,
      "charges": {
        "transaction_tax": 1.498,
        "transaction_tax_type": "stt",
        "exchange_turnover_charge": 0.051681,
        "sebi_turnover_charge": 0.001498,
        "brokerage": 0.01,
        "stamp_duty": 0.22,
        "gst": {
          "igst": 0.011372219999999999,
          "cgst": 0,
          "sgst": 0,
          "total": 0.011372219999999999
        },
        "total": 1.79255122
      },
      "total": 1498
    }
  ]
}
```

Order structure

parameter	
exchange	Name of the exchange
transaction_type	`BUY` / `SELL`
variety	Order variety (regular, amo, co etc.)
product	Margin product to use for the order (margins are blocked based on this) ?
order_type	Order type (MARKET, LIMIT etc.)
quantity	Quantity of the order
price	Price at which the order is going to be placed (LIMIT orders)
trigger_price	Trigger price (for SL, SL-M, CO orders)

Margin structure

parameter	
type	`equity` / `commodity`
tradingsymbol	Trading symbol of the instrument
exchange	Name of the exchange
span	SPAN margins
exposure	Exposure margins
option_premium	Option premium

parameter	
additional	Additional margins
bo	BO margins
cash	Cash credit
var	VAR
pnl	Realised and unrealised profit and loss
leverage	Margin leverage allowed for the trade
charges	The breakdown of the various charges that will be applied to an order
total	Total margin block

Charges structure

Field	Definition
total	Total charges
transaction_tax	Tax levied for each transaction on the exchanges
transaction_tax_type	Type of transaction tax
exchange_turnover_charge	Charge levied by the exchange on the total turnover of the day
sebi_turnover_charge	Charge levied by SEBI on the total turnover of the day
brokerage	The brokerage charge for a particular trade
stamp_duty	Duty levied on the transaction value by Government of India

Field	Definition
gst.igst	Integrated Goods and Services Tax levied by the government
gst.cgst	Central Goods and Services Tax levied by the government
gst.sgst	State Goods and Services Tax levied by the government
gst.total	Total GST

Basket margins

```
curl https://api.kite.trade/margins/basket?consider_positions=true \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
    -H 'Content-Type: application/json' \
    -d '[
    {
        "exchange": "NFO",
        "tradingsymbol": "NIFTY23JUL20600CE",
        "transaction_type": "SELL",
        "variety": "regular",
        "product": "NRML",
        "order_type": "MARKET",
        "quantity": 75,
        "price": 0,
        "trigger_price": 0
    },
    {
        "exchange": "NFO",
        "tradingsymbol": "NIFTY23JUL20700CE",
        "transaction_type": "BUY",
        "variety": "regular",
        "product": "NRML",
        "order_type": "MARKET",
        "quantity": 75,
        "price": 0,
        "trigger_price": 0
    }
]'
```

Query parameters are as follows.

parameter	
consider_positions	Boolean to consider users positions
mode	`compact` - Compact mode will only give the total margins


```
{
  "status": "success",
  "data": {
    "initial": {
      "type": "",
      "tradingsymbol": "",
      "exchange": "",
      "span": 66832.5,
      "exposure": 29151.225000000002,
      "option_premium": 521.25,
      "additional": 0,
      "bo": 0,
      "cash": 0,
      "var": 0,
      "pnl": {
        "realised": 0,
        "unrealised": 0
      },
      "leverage": 0,
      "charges": {
        "transaction_tax": 0,
        "transaction_tax_type": "",
        "exchange_turnover_charge": 0,
        "sebi_turnover_charge": 0,
        "brokerage": 0,
        "stamp_duty": 0,
        "gst": {
          "igst": 0,
          "cgst": 0,
          "sgst": 0,
          "total": 0
        },
        "total": 0
      },
      "total": 96504.975
    },
    "final": {
      "type": "",
      "tradingsymbol": "",
      "exchange": "",
      "span": 7788.000000000007,
      "exposure": 29151.225000000002,
      "option_premium": -2152.5,
```

```
      "additional": 0,
      "bo": 0,
      "cash": 0,
      "var": 0,
      "pnl": {
        "realised": 0,
        "unrealised": 0
      },
      "leverage": 0,
      "charges": {
        "transaction_tax": 0,
        "transaction_tax_type": "",
        "exchange_turnover_charge": 0,
        "sebi_turnover_charge": 0,
        "brokerage": 0,
        "stamp_duty": 0,
        "gst": {
          "igst": 0,
          "cgst": 0,
          "sgst": 0,
          "total": 0
        },
        "total": 0
      },
      "total": 34786.725000000006
    },
    "orders": [
      {
        "type": "equity",
        "tradingsymbol": "NIFTY23JUL20600CE",
        "exchange": "NFO",
        "span": 66832.5,
        "exposure": 29151.225000000002,
        "option_premium": 0,
        "additional": 0,
        "bo": 0,
        "cash": 0,
        "var": 0,
        "pnl": {
          "realised": 0,
          "unrealised": 0
        },
        "leverage": 1,
        "charges": {
          "transaction_tax": 1.67109375,
          "transaction_tax_type": "stt",
          "exchange_turnover_charge": 1.336875,
          "sebi_turnover_charge": 0.00267375,
          "brokerage": 20,
          "stamp_duty": 0,
          "gst": {
            "igst": 3.8411187749999995,
            "cgst": 0,
```

```
            "sgst": 0,
            "total": 3.8411187749999995
          },
          "total": 26.851761274999998
        },
        "total": 95983.725
      },
      {
        "type": "equity",
        "tradingsymbol": "NIFTY23JUL20700CE",
        "exchange": "NFO",
        "span": 0,
        "exposure": 0,
        "option_premium": 521.25,
        "additional": 0,
        "bo": 0,
        "cash": 0,
        "var": 0,
        "pnl": {
          "realised": 0,
          "unrealised": 0
        },
        "leverage": 1,
        "charges": {
          "transaction_tax": 0,
          "transaction_tax_type": "stt",
          "exchange_turnover_charge": 0.260625,
          "sebi_turnover_charge": 0.00052125,
          "brokerage": 20,
          "stamp_duty": 0,
          "gst": {
            "igst": 3.6470063249999995,
            "cgst": 0,
            "sgst": 0,
            "total": 3.6470063249999995
          },
          "total": 23.908152575
        },
        "total": 521.25
      }
    ],
    "charges": {
      "transaction_tax": 0,
      "transaction_tax_type": "",
      "exchange_turnover_charge": 0,
      "sebi_turnover_charge": 0.003195,
      "brokerage": 40,
      "stamp_duty": 0,
      "gst": {
        "igst": 0,
        "cgst": 0,
        "sgst": 0,
        "total": 0
```

```
      },
      "total": 0
    }
  }
}
```

Response structure is as follows.

parameter	
initial	Total margins required to execute the orders
final	Total margins with the spread benefit
orders	Individual margins per order
charges	Final charges block

Note

The final charges block can be ignored as it may not include `transaction_tax` charges because baskets can contain both mcx and equity instruments, with different tax types (STT or CTT). Users can refer to the individual order charges response in the orders block.

Virtual contract note

A virtual contract provides detailed charges order-wise for brokerage, STT, stamp duty, exchange transaction charges, SEBI turnover charge, and GST.

```
curl https://api.kite.trade/charges/orders \
    -H 'X-Kite-Version: 3' \
    -H 'Authorization: token api_key:access_token' \
    -H 'Content-Type: application/json' \
    -d '[
  {
    "order_id": "111111111",
    "exchange": "NSE",
    "tradingsymbol": "SBIN",
    "transaction_type": "BUY",
    "variety": "regular",
```

```
    "product": "CNC",
    "order_type": "MARKET",
    "quantity": 1,
    "average_price": 560
  },
  {
    "order_id": "2222222222",
    "exchange": "MCX",
    "tradingsymbol": "GOLDPETAL23JULFUT",
    "transaction_type": "SELL",
    "variety": "regular",
    "product": "NRML",
    "order_type": "LIMIT",
    "quantity": 1,
    "average_price": 5862
  },
  {
    "order_id": "3333333333",
    "exchange": "NFO",
    "tradingsymbol": "NIFTY2371317900PE",
    "transaction_type": "BUY",
    "variety": "regular",
    "product": "NRML",
    "order_type": "LIMIT",
    "quantity": 100,
    "average_price": 1.5
  }
]'
```

Order structure

parameter	
`order_id` string	Unique order ID (It can be any random string to calculate charges for an imaginary order)
`exchange` string	Name of the exchange
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`transaction_type` string	`BUY` / `SELL`

parameter	
`variety` string	Order variety (regular, amo, co etc.)
`product` string	Margin product to use for the order (margins are blocked based on this) ?
`order_type` string	Order type (MARKET, LIMIT etc.)
`quantity` int64	Quantity of the order
`average_price` float64	Average price at which the order was executed (Note: Should be non-zero).


```
{
  "status": "success",
  "data": [
    {
      "transaction_type": "BUY",
      "tradingsymbol": "SBIN",
      "exchange": "NSE",
      "variety": "regular",
      "product": "CNC",
      "order_type": "MARKET",
      "quantity": 1,
      "price": 560,
      "charges": {
        "transaction_tax": 0.56,
        "transaction_tax_type": "stt",
        "exchange_turnover_charge": 0.01876,
        "sebi_turnover_charge": 0.00056,
        "brokerage": 0,
        "stamp_duty": 0,
        "gst": {
          "igst": 0.0033767999999999997,
          "cgst": 0,
          "sgst": 0,
          "total": 0.0033767999999999997
        },
        "total": 0.5826968
      }
    },
    {
```

```
      "transaction_type": "SELL",
      "tradingsymbol": "GOLDPETAL23JULFUT",
      "exchange": "MCX",
      "variety": "regular",
      "product": "NRML",
      "order_type": "LIMIT",
      "quantity": 1,
      "price": 5862,
      "charges": {
        "transaction_tax": 0.5862,
        "transaction_tax_type": "ctt",
        "exchange_turnover_charge": 0.152412,
        "sebi_turnover_charge": 0.005862,
        "brokerage": 1.7586,
        "stamp_duty": 0,
        "gst": {
          "igst": 0.34503732,
          "cgst": 0,
          "sgst": 0,
          "total": 0.34503732
        },
        "total": 2.84811132
      }
    },
    {
      "transaction_type": "BUY",
      "tradingsymbol": "NIFTY2371317900PE",
      "exchange": "NFO",
      "variety": "regular",
      "product": "NRML",
      "order_type": "LIMIT",
      "quantity": 100,
      "price": 1.5,
      "charges": {
        "transaction_tax": 0,
        "transaction_tax_type": "stt",
        "exchange_turnover_charge": 0.07575,
        "sebi_turnover_charge": 0.00015,
        "brokerage": 20,
        "stamp_duty": 0,
        "gst": {
          "igst": 3.613527,
          "cgst": 0,
          "sgst": 0,
          "total": 3.613527
        },
        "total": 23.689427000000002
      }
    }
  ]
}
```

Response attributes

attribute	
`transaction_type` string	Type of transaction being processed(BUY/SELL).
`tradingsymbol` string	Exchange tradingsymbol of the instrument
`exchange` string	Name of the exchange
`variety` string	Order variety (regular, amo, co etc.)
`product` string	Margin product to use for the order (margins are blocked based on this) ?
`order_type` string	Order type (MARKET, LIMIT etc.)
`quantity` int64	Quantity of the order
`price` float64	Price at which the order is completed
`charges` map	The breakdown of the various charges that will be applied to an order

Offsite order execution

The offsite order execution feature allows you to redirect your users to Kite's exchange approved order page where they place orders and come back to your application seamlessly, like a payment gateway. This way, you do not have to build, maintain, and get exchange approvals for order execution screens. The Kite Publisher program utilizes offsite order execution to provide embeddable Javascript+HTML trade buttons that do not require any API integrations.

Initiating orders

> Example JSON basket

```
[
  {
    "variety": "regular",
    "tradingsymbol": "INFY",
    "exchange": "NSE",
    "transaction_type": "BUY",
    "order_type": "MARKET",
    "quantity": 10,
    "readonly": false
  },
  {
    "variety": "regular",
    "tradingsymbol": "NIFTY15DECFUT",
    "exchange": "NFO",
    "transaction_type": "SELL",
    "order_type": "LIMIT",
    "price": 7845,
    "quantity": 1,
    "readonly": false
  },
  {
    "variety": "co",
    "tradingsymbol": "RELIANCE",
    "exchange": "NSE",
    "transaction_type": "BUY",
    "order_type": "LIMIT",
    "product": "MIS",
    "price": 915.15,
    "quantity": 1,
    "trigger_price": 910,
    "readonly": true
```

```
  }
]
```

> Posting the JSON basket

```
<form
  method="post"
  id="basket-form"
  action="https://kite.zerodha.com/connect/basket"
>
  <input type="hidden" name="api_key" value="xxx" />
  <input type="hidden" id="basket" name="data" value="" />
</form>

<script>
  document.getElementById("basket").value = your_basket;
  document.getElementById("basket-form").submit();
</script>
```

It is possible to send multiple orders which the user then confirms on a shopping basket like interface. You should prepare a JSON list of instruments to be traded with the required order parameters and `POST` it as a form field with the name `data` along with your `api_key` to **`https://kite.zerodha.com/connect/basket`**.

This is a browser / mobile (webview) request and has to happen at the user's end, although the basket preparation can happen in the backend. The easiest way to make the request is to create a hidden form, insert the JSON payload into it and submit it automatically using Javascript.

If you're preparing the basket client side on your web application, you can use the Kite Publisher javascript plugin to make things easier.

> **Note**
>
> You do not have to initiate a login using the login API to do offsite order execution. If a user is not already logged in, they'll be asked to login, otherwise, they'll be taken to the order basket directly. Either way, in the end, you will receive `status` and `request_token` at your `redirect_url` like in the login flow.

Linking offsite execution to Kite Connect

Offsite order execution follows the same flow as login, except for the order basket that appears after the login. Once the user is done placing orders, the basket will redirect back to your registered

`redirect_login` along with a `request_key`, exactly like login. You are free to use this key to create a new Kite Connect session for further API interactions, or disregard it altogether.

Request parameters

parameter	
`variety`	Order variety (`regular`, `amo`, `co`. Defaults to `regular`)
`tradingsymbol`	Tradingsymbol of the instrument
`exchange`	Name of the exchange
`transaction_type`	BUY or SELL
`order_type`	Order type (MARKET, LIMIT etc.)
`quantity`	Quantity to transact
`product`	Margin product to use for the order (margins are blocked based on this) ?
`price`	For LIMIT orders
`trigger_price`	For SL, SL-M etc.
`disclosed_quantity`	Quantity to disclose publicly (for equity trades)
`validity`	Order validity
`readonly`	Default is `false`. If set to true, the UI does not allow the user to edit values such as quantity, price etc., and they can only review and execute.
`tag`	An optional tag to apply to an order to identify it (alphanumeric, max 20 chars)

Publisher JS Plugin

> **Warning**
>
> The publisher JS plugin will not function in the iOS WebView due to Safari's new cookie policy, which blocks cross-domain cookie management in iframes. You will need to implement offsite order execution. We recommend using offsite order execution for all Kite Publisher use cases.

The Kite Publisher Javascript plugin lets you add one-click trade buttons to your webpage. It works like a basket combined with a payment gateway, where an inline popup opens on your webpage, guides the user through a trade, and lands the user back on your page. As described in the offsite order execution section, it is possible to capture the `request_token` from this flow to start a Kite Connect session as well.

You can add one or more stocks to the basket (maximum 10) dynamically using the Javascript plugin, or embed simple static buttons using plain HTML.

Getting started

```
<script src="https://kite.trade/publisher.js?v=3"></script>
```

Include Kite Publisher on your webpage by pasting the following script tag at the end of your webpage, just before the closing `</body>` tag. You only need to include this once to render any number of buttons on a page.

Branded HTML5 buttons

```
<!--  A link that initiates a buy (market) of the SBIN stock //-->
<kite-button
  href="#"
```

```
  data-kite="your_api_key"
  data-exchange="NSE"
  data-tradingsymbol="SBIN"
  data-transaction_type="BUY"
  data-quantity="1"
  data-order_type="MARKET"
  >Buy SBI stock</kite-button
>
```

You can use the custom `<kite-button>` HTML5 tag to render branded Kite buttons that initiate a trade with a single click. The branded buttons work in a similar fashion to social media buttons, and you can include as many as you want on a page.

Custom HTML5 buttons

```
<!--  A link that initiates a buy (market) of the SBIN stock //-->
<a
  href="#"
  data-kite="your_api_key"
  data-exchange="NSE"
  data-tradingsymbol="SBIN"
  data-transaction_type="BUY"
  data-quantity="1"
  data-order_type="MARKET"
  >Buy SBI stock</a
>

<!--  A button that initiates a sell (LIMIT) of the RELIANCE stock //-->
<button
  data-kite="your_api_key"
  data-exchange="NSE"
  data-tradingsymbol="RELIANCE"
  data-transaction_type="SELL"
  data-quantity="1"
  data-order_type="LIMIT"
  data-price="100"
>
  Buy RELIANCE
</button>

<!--  A button that initiates a CO of the RELIANCE stock //-->
<button
  data-kite="your_api_key"
  data-exchange="NSE"
  data-tradingsymbol="RELIANCE"
  data-transaction_type="BUY"
  data-quantity="1"
  data-order_type="LIMIT"
```

```
  data-variety="co"
  data-product="MIS"
  data-price="915"
  data-trigger_price="910"
>
  Buy RELIANCE (Cover Order)
</button>
```

You can use the HTML5 data attributes on any HTML element and turn it into a trade button which gets invoked with a single click. The examples on the right show a link and a button being turned into trade buttons.

Generating dynamic buttons with Javascript

```
<!-- A Kite button will be generated inside this container //-->
<p id="default-button"></p>

<!-- The basket will be linked to this element's onClick //-->
<button id="custom-button">Buy the basket</button>

<!-- Include the plugin //-->
<script src="https://kite.trade/publisher.js?v=3"></script>

<script>
  // Only run your custom code once KiteConnect has fully initialised.
  // Use KiteConnect.ready() to achieve this.
  KiteConnect.ready(function () {
    // Initialize a new Kite instance.
    // You can initialize multiple instances if you need.
    var kite = new KiteConnect("your_api_key");

    // Add a stock to the basket
    kite.add({
      exchange: "NSE",
      tradingsymbol: "INFY",
      quantity: 5,
      transaction_type: "BUY",
      order_type: "MARKET",
    });

    // Add another stock
    kite.add({
      exchange: "NSE",
      tradingsymbol: "SBIN",
      quantity: 1,
      order_type: "LIMIT",
      transaction_type: "SELL",
      price: 105,
```

```
    });

    // Add a Cover Order
    kite.add({
      tradingsymbol: "RELIANCE",
      exchange: "NSE",
      transaction_type: "BUY",
      order_type: "LIMIT",
      product: "MIS",
      price: 915.15,
      quantity: 1,
      variety: "co",
      trigger_price: 910,
      readonly: true,
    });

    // Register an (optional) callback.
    kite.finished(function (status, request_token) {
      alert("Finished. Status is " + status);
    });

    // Render the in-built button inside a given target
    kite.renderButton("#default-button");

    // OR, link the basket to any existing element you want
    kite.link("#custom-button");
  });
</script>
```

You can create a basket of stocks and get the plugin to render a Kite button that executes it, or link the basket to your own button (or any HTML element).

The plugin loads it's assets asynchronously, so it's important that you initialise your custom KiteConnect calls after it has fully loaded. You need to use the `KiteConnect.ready()` function to achieve this.

> **Note**
>
> Include your custom code after the publisher.js inclusion. Also, make sure to wrap your code in `KiteConnect.ready()`

Parameters

parameter	
`variety`	Order variety (`regular` , `amo` , `co` . Defaults to `regular`)
`tradingsymbol`	Tradingsymbol of the instrument
`exchange`	Name of the exchange
`transaction_type`	BUY or SELL
`order_type`	Order type (MARKET, LIMIT etc.)
`quantity`	Quantity to transact
`product`	Margin product to use for the order (margins are blocked based on this) ?
`price`	For LIMIT orders
`trigger_price`	For SL, SL-M etc.
`disclosed_quantity`	Quantity to disclose publicly (for equity trades)
`validity`	Order validity
`readonly`	Default is `false` . If set to true, the UI does not allow the user to edit values such as quantity, price etc., and they can only review and execute.
`tag`	An optional tag to apply to an order to identify it (alphanumeric, max 8 chars)

Methods

method	arguments	
`KiteConnect.ready()`	`function()`	Safe wrapper for all API calls that waits asynchronously for all assets to load.
`add()`	`entry`	Adds an object literal {} with the parameters mentioned in the previous section represeting a single trading entry to the basket.
`get()`		Returns an array[] of all added entries.
`count()`		Returns the number of added entries.
`setOption()`	`key, value`	Sets the value for certain supported keys.
	`redirect_url`	A redirect URL to override the registered Kite Connect redirect URL (using `setOption()`). The value can be a single '#', a 127.0.0.1 url for testing, or a fully qualified URL belonging to the same domain as the registered URL.
`renderButton()`	`element_selector`	Renders a branded Kite button in the given target element, which when clicked, will start the transaction in an overlay popup. `element_selector` is an HTML selector, for example, `#buy-button`, `.buttons` etc.
`link()`	`element_selector`	Links the basket to the given HTML element, which when clicked, will start the transaction in an overlay popup. `element_selector` is an HTML selector, for example, `#buy-button`, `.buttons` etc.
`html()`		Returns a serialized HTML form with the necessary hidden fields and the basket payload which can be written to the

method	arguments	
		document body and submitted to initiate the transaction.
`finished()`	`function(status, request_token)`	Register a callback which is triggered after order placement is finished.
`authHoldings()`	`request_id, callback(data)`	Request ID from the holdings authorisation call along with an optional callback function that triggers when the holdings authorisation flow finishes.

Mobile and Desktop apps

As described in the authentication section, the login flow ends with a redirect to your registered `redirect_url` with the `request_token` after a successful login. When this redirect end point is a web application, it is easy to get the token and exchange it for an `access_token`. When it's a desktop or a mobile application without a server backend, the approach is different.

Similar to how Google and Facebook authentication flows work on mobile apps, you will need to open a webview (browser view) component from within your application pointing to the login url. The entire login flow will happen within this webview. As it's an in-app component, you will have a certain level of control over it, including reading the current location (URL) of the component. It is then possible to monitor location changes using a change event or a poll timer to determine when the redirect happens, and extract the `request_token` from the URL.

> **Note**
>
> Don't forget to enable cookie (and 3rd party cookie) support in your webview or the login may not work.

In essence:

- Register a `redirect_url` with us when you apply for your API credentials. This can be a blank page even. Eg: `https://yoursite.com/kite-redirect`. For personal desktop apps, you can run a local web server and use `127.0.0.1` as the `redirect_url`'s' host.
- Start the login flow by opening an webview component within your desktop or mobile application `https://kite.zerodha.com/connect/login?api_key=xxx`
- From within your application, monitor changes in the URL of the component as the login happens
- When the URL changes to `https://yoursite.com/kite-redirect?request_token=yyy&status=zzz`, extract the `request_token` and `status_from` the URL and close the webview component.

If you intent on distributing your mobile or desktop application to the public, do not embed the api_secret in the application. Use a server backend to do the token exchange on behalf of your application.

Changelog

Kite Connect 3.1 — Changes

Orders

- Added `exchange_update_timestamp` to order response
- Added `meta` field to order response

GTT orders

- Added support placing, updating, and deleting for GTT orders

Quote Call

- Added Circuit Limits to quote call response

Historical Data

- Added OI to historical candle data response

Kite Connect 3.0 — Changes

The new APIs continue to be on the same route `api.kite.trade`, but to connect to the 3.0 backend, the header `X-Kite-Version: 3` should be sent with every HTTP request. This is illustrated throughout the documentation in the `curl` examples.

API endpoint changes

Endpoint	Status	
`/quote`	New	Retrieve complete market quotes

Endpoint	Status	
		(including depth) for up to 500 instruments in one go
`/quote/ohlc`	New	Retrieve OHLC quotes for up to 1000 instruments in one go
`/quote/ltp`	New	Retrieve LTP (`last_price`) quotes for up to 1000 instruments in one go
`/user/profile`	New	Retrieve user profile
`/market/instruments/:exchange/:tradingsymbol`	Removed	Replaced by the new `/quote` API

Login

For obtaining a 3.0 login session, the query param **v=3** should be added to the initial login URL.

`https://kite.zerodha.com/connect/login?v=3&api_key=xxx`

Authentication

In the previous version, request authentication was done by sending the `api_key` and `access_token` query parameters. In 3.0, this has been replaced by the `Authorization` header. This is evident in all the examples in this documentation.

All requests now have to be authenticated by sending the HTTP header:

```
Authorization: token api_key:access_token
```

WebSocket

- The new WebSocket address is **wss://ws.kite.trade**. This is already embedded into the updated Kite Connect 3.0 client libraries.
- The WebSocket API's binary quote protocol has changed to accommodate new fields. The new packet structure is described in the WebSocket section. New fields — Open Interest, Open Interest Day High, Open Interest Day Low, Last trade timestamp, and Exchange timestamp.
- The WebSocket API now also supports delivery of realtime order Postbacks and other types of messages.
- For connection authentication, the old combination of `api_key, user_id, public_token` is not supported anymore. It is now just `api_key` and `access_token` query parameters.
- The number of instruments that can be subscribed to has been increased from 200 to **1000**.

Changes to API responses

Date	Change

`/session/token`	New fields:
	`api_key`
	`refresh_token`
	
	Changed fields:
	`order_type -> order_types`
	`exchange -> exchanges`
	`product -> products`
	
	Removed fields:
	`password_reset`
	`member_id`

`/portfolio/positions`	New fields:
	`day_buy_qty`
	`day_buy_price`
	`day_buy_value`
	`day_sell_qty`
	`day_sell_price`
	`day_sell_value`

Date	Change

`/trades`
`/orders/:order_id/trades`
	New fields:
	`fill_timestamp`
	
	Removed fields:
	`order_timestamp`

Changelog

Date	Change
2019-12-10	Released minor feature updates - Kite Connect 3.1
2018-01-17	Major version upgrade to Kite Connect 3 from API v1.
2017-09-11	Introduced bulk quote APIs
2016-05-07	`/orders` call and bracket order modification and cancellation now involve the new `parent_order_id` parameter
2016-07-02	Several new fields added to the Webhooks payload (`exchange_timestamp`, `order_type`, `product`, `unfilled_quantity`, `validity`)
2017-09-11	Added support for granular timestamp `from` and `to` queries to historical data APIs
2017-09-12	Added bulk quote fetch APIs
2017-11-13	Added support for 'UPDATE' Postbacks