VOCSET API Documentation

Version History

Version	Date	Changes
1.2	2026-07-21	Added programmatic API key rotation endpoint (`POST /api/auth/apiKey/rotate`)
1.1.1	2026-02-01	Simplified mult-leg trade format.
1.1.0	2026-01-16	Updated for multi-leg trade support
1.0.1	2025-12-29	Various small updates
1.0.0	2024-11-19	Initial release - single-leg trades

This document outlines the VOCSET API used to upload trade data directly into the application. Currently a very restricted sub-set of the application features are available from the API - just those operations to handle trade data.

Authentication & Authorisation

All endpoints are secured using an API key. Setting up a Key / Secret can be managed from the VOCSET application, in your user profile.

When calling the API, the api key and api secret should be placed in the named headers in all requests.

```
X-API-KEY: "api_key"
X-API-SECRET: "api_secret"
```

Handling Trades

Each endpoint manipulates or displays information related to the User whose Token is provided with the request:

- Show Trades : `GET /api/trade`
- Upload Trades : `POST /api/trade`

Managing API Keys

- Rotate API Key : `POST /api/auth/apiKey/rotate`

Upload Active Trades

Allow the user with valid API Key to upload their trades into the daily blotter, and submit these to their clients.

URL : `/api/trade`

Method : `POST`

Auth required : YES (API Key)

Permissions required : None

Data constraints (Single-Leg Trades)

```
{
  "tradeID": "<MANDATORY>",
  "traceID": "<OPTIONAL>",
  "tradeDate": "YYYY-MM-DD <MANDATORY> cannot be in future",
  "side": "Buy|Sell <MANDATORY>",
  "quantity": "676 <MANDATORY> integer - limited to 32 digits - no decimals",
  "price": "2009 <MANDATORY> decimal - limited to 24 digits + 8 dps",
  "instrumentCode": "String <MANDATORY> contract code only, no maturity information",
  "maturity": "YYYY-MM-DD <MANDATORY>>",
  "strike": "Decimal <MANDATORY when assetClass=Option>",
  "optionType": "Call|Put <MANDATORY when assetClass=Option>",
  "mic": "String <MANDATORY>",
  "client": "String <MANDATORY>",
  "executingAccount": "String <OPTIONAL>",
  "executingBroker": "String <OPTIONAL> only necessary when firms need to use multiple
  "productDescription": "String <OPTIONAL>>",
  "clearingAccount": "String <MANDATORY>",
  "clearingBroker": "String <MANDATORY>",
  "carryBroker": "String <OPTIONAL> only use when requested. Can be defaulted",
  "executionTime": "DateTime(ISO 8601) <MANDATORY> example: 2024-12-01T14:00:04-05:00",
  "giveupTime": "DateTime(ISO 8601) <OPTIONAL> example: 2024-12-01T14:01:00-05:00",
  "assetClass": "Future|Option <MANDATORY>"
}
```

Data constraints (Multi-Leg Trades)

Multi-leg trades use a parent-child structure where the parent trade contains an array of leg trades.

```
{
  "tradeID": "<MANDATORY> unique identifier for parent trade",
  "strategyName": "CalendarSpread|CalendarStrip|VerticalSpread|Straddle|Strangle|Butter
  "tradeDate": "YYYY-MM-DD <MANDATORY> cannot be in future",
  "side": "Buy|Sell <MANDATORY>",
  "quantity": "<MANDATORY>",
  "price": "<MANDATORY> can be 0 for spread trades",
  "instrumentCode": "String <MANDATORY>",
  "maturity": "YYYY-MM-DD <MANDATORY>",
  "mic": "String <MANDATORY>",
  "client": "String <MANDATORY>",
  "clearingAccount": "String <MANDATORY>",
  "clearingBroker": "String <MANDATORY>",
  "executionTime": "DateTime(ISO 8601) <MANDATORY>",
  "assetClass": "Future|Option <MANDATORY>",
  "legs": [
    {
      "tradeID": "<MANDATORY> unique identifier for leg",
      "side": "Buy|Sell <MANDATORY>",
      "quantity": "<MANDATORY>",
      "price": "<MANDATORY>",
      "instrumentCode": "String <MANDATORY>",
      "maturity": "YYYY-MM-DD <MANDATORY>",
      "mic": "String <MANDATORY>",
      "executionTime": "DateTime(ISO 8601) <MANDATORY>",
      "assetClass": "Future|Option <MANDATORY>"
    }
  ]
}
```

Strategy Types and Minimum Leg Requirements

Strategy	Min Legs	Description
CalendarSpread	2	Different expiration dates, same strike
CalendarStrip	2	Buy/Sell Consecutive expiries
VerticalSpread	2	Same expiration, different strikes

Strategy	Min Legs	Description
Straddle	2	Buy/Sell both call and put at same strike
Strangle	2	Buy/Sell call and put at different strikes
Butterfly	3	Three strikes with defined wings
Condor	4	Four different strikes
IronButterfly	4	Short butterfly with protective wings
IronCondor	4	Short strangle with protective collars
Strip	3	One call + two puts
Strap	3	Two calls + one put
Custom	2	User-defined structure

Multi-Leg Validation Rules

- Legs inherit `client`, `tradeDate`, `clearingBroker`, `clearingAccount`, `executingBroker`, and `executingAccount` from the parent trade
- Multi-leg trades must have at least 2 legs (or more depending on strategy)
- If any leg fails validation, the entire parent trade is rejected

Note for `Client, executingBroker, clearingBroker, executingAccount, clearingAccount` please refer to the company codes visible in VOCSET gui.

Data examples

```
[
  {
    "tradeID": "20241119-001",
    "tradeDate": "2024-11-19",
    "side": "Buy",
    "quantity": "676",
    "price": "2009",
    "instrumentCode": "CL",
    "maturity": "2024-12-01",
    "mic": "XNYM",
    "client": "CTCINC",
    "productDescription": "Brent Crude",
    "clearingAccount": "GC123",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Future"
  },
  {
    "tradeID": "20241119-002",
    "tradeDate": "2024-11-19",
    "side": "Buy",
    "quantity": "47",
    "price": "1979",
    "instrumentCode": "NG",
    "maturity": "2024-12-01",
    "mic": "XNYM",
    "client": "CTCINC",
    "productDescription": "Natural Gas",
    "clearingAccount": "GC123",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Future"
  },
  {
    "tradeID": "20241119-003",
    "tradeDate": "2024-11-19",
    "side": "Sell",
    "quantity": "1",
    "price": "77",
    "instrumentCode": "C",
    "maturity": "2025-03-01",
    "mic": "NDEX",
```

```
    "client": "CTCINC",
    "productDescription": "EUA Futures",
    "clearingAccount": "GC456",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Future"
  },
  {
    "tradeID": "20241119-004",
    "tradeDate": "2024-11-19",
    "side": "Buy",
    "quantity": "50",
    "price": "50",
    "instrumentCode": "T",
    "maturity": "2025-03-01",
    "mic": "IFEU",
    "client": "CTCINC",
    "productDescription": "WTI Crude",
    "clearingAccount": "GC456",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Future"
  },
  {
    "tradeID": "20241119-005",
    "tradeDate": "2024-11-19",
    "side": "Sell",
    "quantity": "1",
    "price": "100",
    "instrumentCode": "B",
    "maturity": "2025-06-01",
    "mic": "IFEU",
    "client": "CTCINC",
    "productDescription": "Brent Crude ICE",
    "clearingAccount": "GC123",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Future"
  },
  {
    "tradeID": "20241119-006",
    "tradeDate": "2024-11-19",
```

```
    "side": "Sell",
    "quantity": "1",
    "price": "100",
    "instrumentCode": "B",
    "maturity": "2025-06-01",
    "strike": 100.1,
    "optionType": "Call",
    "mic": "IFEU",
    "client": "CTCINC",
    "productDescription": "Brent Crude ICE",
    "clearingAccount": "GC123",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "giveupTime": "2024-11-19T14:01:00-05:00",
    "assetClass": "Option"
  }
]
```

Multi-Leg Trade Example (Calendar Spread)

```
[
  {
    "tradeID": "ML-20241119-001",
    "strategyName": "CalendarSpread",
    "tradeDate": "2024-11-19",
    "side": "Buy",
    "quantity": "10",
    "price": "0",
    "instrumentCode": "CL",
    "maturity": "2025-01-01",
    "mic": "XNYM",
    "client": "CTCINC",
    "productDescription": "Crude Oil Calendar Spread",
    "clearingAccount": "GC123",
    "clearingBroker": "DBAG",
    "executionTime": "2024-11-19T14:00:04-05:00",
    "assetClass": "Future",
    "legs": [
      {
        "tradeID": "ML-20241119-001-L1",
        "side": "Buy",
        "quantity": "10",
        "price": "70.50",
        "instrumentCode": "CL",
        "maturity": "2025-01-01",
        "mic": "XNYM",
        "executionTime": "2024-11-19T14:00:04-05:00",
        "assetClass": "Future"
      },
      {
        "tradeID": "ML-20241119-001-L2",
        "side": "Sell",
        "quantity": "10",
        "price": "71.25",
        "instrumentCode": "CL",
        "maturity": "2025-02-01",
        "mic": "XNYM",
        "executionTime": "2024-11-19T14:00:04-05:00",
        "assetClass": "Future"
      }
    ]
  }
]
```

Success Responses

Condition : Data provided is valid and User is Authenticated.

Code : `200 OK` (response payload need to be examined for individual trade errors)

Content example : Response will reflect back the updated information.

```
{
  "timestamp": "2024-11-21T17:16:12.424011385Z",
  "status": 200,
  "result": [
    {
      "id": "20241119-001",
      "status": "OK",
      "message": ""
    },
    {
      "id": "20241119-002",
      "status": "OK",
      "message": ""
    },
    {
      "id": "20241119-003",
      "status": "OK",
      "message": ""
    },
    {
      "id": "20241119-004",
      "status": "OK",
      "message": ""
    },
    {
      "id": "20241119-005",
      "status": "OK",
      "message": ""
    },
    {
      "id": "20241119-006",
      "status": "OK",
      "message": ""
    }
  ],
  "path": "/api/trade"
}
```


Success Response (Partial)

Condition : Data provided is correctly formatted but some trades may have invalid data

Code : `200 OK`

Content example :


```
{
"timestamp": "2024-11-21T17:16:12.424011385Z",
"status": 200,
"result": [
    {
        "id": "20241119-001",
        "status": "ERROR",
        "message": "Trade ID [20241119-001] is not unique",
        "field": "tradeId",
        "value": "20241119-001"
    },
    {
        "id": "20241119-002",
        "status": "ERROR",
        "message": "Trade ID [20241119-002] is not unique",
        "field": "tradeId",
        "value": "20241119-002"
    },
    {
        "id": "20241119-006",
        "status": "OK",
        "message": ""
    }
  ]
}
```


Notes

- Because some trades in the upload were valid, but the payload was formatted correctly, but some trades in the upload were correct, a partially successful upload is possible.
- In these instances, the return code will be 200, but the payload will indicate individual trade errors on the status field of each result element as indicated above.

Error Response

Condition : Bad data has been supplied in the payload.

Code : `400 BAD REQUEST`

Error Response

Condition : VOCSET encountered an unexpected / unhandled error while processing.

Code : `500 INTERNAL SERVER ERROR`

Show Active Trades

Gets an `Array` of `Trade` objects representing the currently active trades in the blotter for the API key owner's company.

URL : `/api/trades`

Method : `GET`

Auth required : YES (API Key)

Permissions required : None

Success Response

Code : `200 OK`

Content examples

Details of a `Futures` trade on `IFEU` exchange.

```
[{
  "tradeID": "20241119-005",
  "userID": "jj@java2go.com",
  "side": "Sell",
  "quantity": 1,
  "price": 100.00000000,
  "tradeDate": "2024-11-19",
  "currency": "USD",
  "mic": "IFEU",
  "assetClass": "Future",
  "instrumentCode": "B",
  "instrumentCodeType": "Ticker",
  "optionType": "None",
  "maturity": "2025-06-01",
  "client": "CTCINC",
  "executingBroker": "JJFUTLTD",
  "executingAccount": "",
  "clearingBroker": "DBAG",
  "clearingAccount": "GC123",
  "executionTime": "2024-11-19T23:33:00.895319Z",
  "reportedTime": "2024-11-19T23:33:01.492583Z"
}]
```


Details of an `Option` trade on `XNYM` exchange.

```
[{
  "tradeID": "119934867",
  "userID": "jj@java2go.com",
  "side": "Sell",
  "quantity": 200,
  "price": 0.09600000,
  "tradeDate": "2024-10-27",
  "currency": "USD",
  "mic": "XNYM",
  "assetClass": "Option",
  "instrumentCode": "ON",
  "instrumentCodeType": "Ticker",
  "strike": 3.20,
  "optionType": "Call",
  "maturity": "2024-12-01",
  "client": "CTCINC",
  "executingBroker": "JJFUTLTD",
  "executingAccount": "",
  "clearingBroker": "JPMLLC",
  "clearingAccount": "CTC123",
  "executionTime": "2024-10-27T13:27:33Z",
  "giveupTime": "2024-10-28T13:32:01Z",
  "reportedTime": "2024-10-31T14:54:07.630565Z"
}]
```

Details of a `CalendarSpread` multi-leg trade on `IFEU` exchange.

```
[{
  "tradeID": "ML-20250116-001",
  "userID": "jj@java2go.com",
  "strategyName": "CalendarSpread",
  "side": "Buy",
  "quantity": 10,
  "price": 0.00000000,
  "tradeDate": "2025-01-16",
  "currency": "USD",
  "mic": "IFEU",
  "assetClass": "Future",
  "instrumentCode": "B",
  "instrumentCodeType": "Ticker",
  "optionType": "None",
  "maturity": "2025-03-01",
  "client": "CTCLTD",
  "executingBroker": "DBAG",
  "executingAccount": "",
  "clearingBroker": "DBAG",
  "clearingAccount": "GC123",
  "executionTime": "2025-01-16T10:00:00Z",
  "reportedTime": "2025-01-16T10:00:01.123456Z",
  "legs": [
    {
      "tradeID": "ML-20250116-001-L1",
      "side": "Buy",
      "quantity": 10,
      "price": 76.00000000,
      "mic": "IFEU",
      "assetClass": "Future",
      "instrumentCode": "B",
      "maturity": "2025-03-01",
      "executionTime": "2025-01-16T10:00:00Z"
    },
    {
      "tradeID": "ML-20250116-001-L2",
      "side": "Sell",
      "quantity": 10,
      "price": 75.50000000,
      "mic": "IFEU",
      "assetClass": "Future",
      "instrumentCode": "B",
      "maturity": "2025-04-01",
      "executionTime": "2025-01-16T10:00:00Z"
```

```
    }
  ]
}]
```

Details of a `Butterfly` options multi-leg trade on `IFEU` exchange.

```
[{
  "tradeID": "ML-20250116-003",
  "userID": "jj@java2go.com",
  "strategyName": "Butterfly",
  "side": "Buy",
  "quantity": 10,
  "price": 0.00000000,
  "tradeDate": "2025-01-16",
  "currency": "USD",
  "mic": "IFEU",
  "assetClass": "Option",
  "instrumentCode": "B",
  "instrumentCodeType": "Ticker",
  "strike": 75.00,
  "optionType": "Call",
  "maturity": "2025-06-01",
  "client": "CTCLTD",
  "executingBroker": "DBAG",
  "executingAccount": "",
  "clearingBroker": "DBAG",
  "clearingAccount": "GC123",
  "executionTime": "2025-01-16T10:00:00Z",
  "reportedTime": "2025-01-16T10:00:01.123456Z",
  "legs": [
    {
      "tradeID": "ML-20250116-003-L1",
      "side": "Buy",
      "quantity": 10,
      "price": 5.00000000,
      "mic": "IFEU",
      "assetClass": "Option",
      "instrumentCode": "B",
      "maturity": "2025-06-01",
      "strike": 70.00,
      "optionType": "Call",
      "executionTime": "2025-01-16T10:00:00Z"
    },
    {
      "tradeID": "ML-20250116-003-L2",
      "side": "Sell",
      "quantity": 20,
      "price": 3.00000000,
      "mic": "IFEU",
      "assetClass": "Option",
```

```
      "instrumentCode": "B",
      "maturity": "2025-06-01",
      "strike": 75.00,
      "optionType": "Call",
      "executionTime": "2025-01-16T10:00:00Z"
    },
    {
      "tradeID": "ML-20250116-003-L3",
      "side": "Buy",
      "quantity": 10,
      "price": 1.50000000,
      "mic": "IFEU",
      "assetClass": "Option",
      "instrumentCode": "B",
      "maturity": "2025-06-01",
      "strike": 80.00,
      "optionType": "Call",
      "executionTime": "2025-01-16T10:00:00Z"
    }
  ]
}]
```


Notes

- The fields `strike` and `optionType` are only returned for `Option` trades.
- Multi-leg trades include a `strategyName` field and a `legs` array containing individual leg trades.
- Legs inherit `client`, `tradeDate`, `clearingBroker`, `clearingAccount`, `executingBroker`, and `executingAccount` from the parent trade.

Rotate API Key

Allow a client holding a **valid** API key + secret to programmatically mint a **new** key + secret without logging into the VOCSET GUI or entering a TOTP code. The current credential is used to authenticate the request; on success the old key/secret is **burned immediately** and the newly minted pair is returned in the response.

This enables headless / server-to-server clients to renew their own credentials before the 60-day expiry, replacing the previous GUI-only flow.

URL : `/api/auth/apiKey/rotate`

Method : `POST`

Auth required : YES — the *current* API key + secret (presented in the standard headers). No JWT, session, or TOTP is required.

Permissions required : None (the credential authorises rotating **only itself**).

Request headers

```
X-API-KEY: "<current_api_key>"
X-API-SECRET: "<current_api_secret>"
```

Request body : None.

Behaviour

- The presented key + secret are validated. If either is wrong the request is rejected with a generic authentication error (the response does **not** reveal which of the two was invalid).
- A new key and secret are generated using a secure random source:
- `apiKey` — prefixed `vk_` (a non-secret handle)
- `apiSecret` — prefixed `vs_` (128-bit key / 256-bit secret respectively)
- The `apiLabel` is preserved; the key creation date is reset to the time of rotation, restarting the 60-day validity period.
- The old credential stops authenticating **the instant the call commits** — there is no overlap window.
- A rotation notification email is sent to the key owner, and the rotation is recorded in the audit log (never the secret itself).

Expiry grace period

A key may be rotated up to **7 days past** its 60-day expiry (configurable). Beyond this grace window, rotation is refused and the key must be re-issued via the GUI (session + TOTP) flow.

Restrictions

- **Service accounts** cannot hold API keys and are refused rotation.
- **Inactive users** are refused rotation.

Success Response

Condition : The presented key + secret are valid and within the (expiry + grace) window.

Code : `200 OK`

Headers : `Cache-Control: no-store` is set so the secret is not cached by intermediaries.

Content example :

```
{
  "timestamp": "2026-07-21T17:16:12.424011385Z",
  "status": 200,
  "result": {
    "apiKey": "vk_8Qw3nR1tZ7yKpL0aB2cD4e",
    "apiSecret": "vs_Nf9Xk2Lp7Qw3nR1tZ7yKpL0aB2cD4eG6hJ8mN0oP2q",
    "apiLabel": "cron-uploader",
    "apiKeyCreationDate": "2026-07-21T17:16:12.424011385Z"
  },
  "path": "/api/auth/apiKey/rotate"
}
```

> **Important** : The response is the **only** copy of the new secret — it is returned once and cannot be retrieved again. Persist it before making any further calls. The old key/secret is invalid immediately, so the client must switch to the new pair for all subsequent requests.

Error Response

Condition : Missing headers, or the presented key/secret is invalid (including a credential that has already been burned by a previous rotation).

Code : `401 UNAUTHORIZED`

Content example :

```
{
  "timestamp": "2026-07-21T17:16:12.424011385Z",
  "status": 401,
  "reason": "AUTHENTICATION_FAILED",
  "message": "Invalid API key or secret",
  "path": "/api/auth/apiKey/rotate"
}
```

Error Response

Condition : The credential belongs to a service account or an inactive user.

Code : `403 FORBIDDEN`

Content example :

```
{
  "timestamp": "2026-07-21T17:16:12.424011385Z",
  "status": 403,
  "reason": "ACTION_UNAUTHORIZED",
  "message": "Service accounts cannot hold API keys",
  "path": "/api/auth/apiKey/rotate"
}
```

Error Response

Condition : The key has no creation date, or is expired beyond the rotation grace window. The key must be re-issued via the GUI.

Code : `400 BAD REQUEST`

Content example :

```
{
  "timestamp": "2026-07-21T17:16:12.424011385Z",
  "status": 400,
  "reason": "API_KEY_EXPIRED",
  "message": "API key expired beyond rotation grace",
  "path": "/api/auth/apiKey/rotate"
}
```

Error Response

Condition : VOCSET encountered an unexpected / unhandled error while processing.

Code : `500 INTERNAL SERVER ERROR`

Example

```
# Rotate -> 200 with new key/secret/creationDate
curl -sS -X POST https://api.vocset.net/api/auth/apiKey/rotate \
  -H "X-API-KEY: $OLD_KEY" \
  -H "X-API-SECRET: $OLD_SECRET"

# Old creds are now rejected on protected endpoints -> 401
curl -sS -o /dev/null -w "%{http_code}\n" https://api.vocset.net/api/trade \
  -H "X-API-KEY: $OLD_KEY" -H "X-API-SECRET: $OLD_SECRET"

# New creds authenticate -> 200
curl -sS -o /dev/null -w "%{http_code}\n" https://api.vocset.net/api/trade \
  -H "X-API-KEY: $NEW_KEY" -H "X-API-SECRET: $NEW_SECRET"
```

Notes

- Rotation is **self-scoped**: a credential can only rotate itself and the new key inherits the same user, role, and privileges. There is no way to rotate another user's key.
- Concurrent rotations of the same key are **last-write-wins**; the losing caller is left holding a dead secret. Serialise rotations for a given key.
- If a client crashes after the server commits but before it stores the new secret, the old key is already burned and the new secret is lost — recover by re-issuing the key via the GUI (session + TOTP).

- Browser clients should not use API-key authentication; API keys are intended for server-to-server use.