# GENERAL

## Main concepts

The solution provided to the propfirm is structured by the following components:

- User Dashboard: propfirm's customers can login on this portal in order to see their metrics, open orders/positions, trading report and platforms(if applicable)
- Admin Dashboard: propfirm's managers has access to this area and it can be used for the following purposes:
    - customer support: admins can access to all information about accounts, historical orders, historical trades and so on
    - management: creation of user, trading rule and so on
- APIs:
    - Rest API: they are used in order to manage different entities, like: User, Group Universes, Trading Rules, Subscriptions and Trading Accounts.
    - Webhooks: enabled on request and they allows the propfirm to get notified on real time when there are some changes/events on the following entities:
        - Accounts: Creation, Snapshot updates and status changes
        - Positions: Overnight
        - Subscriptions: Creation, Sign of the agreement (NO_PRO / PRO), changes of status
        - Report of a trade closed
- Important: all entities can be created, edited and deleted both from API and admin dashboard. Furthermore all the information available on user's dashboard are available also through API. It is care of the propfirm understanding what it is better to integrate on their side through API or in which scenarios it is better to use our portal.

## Architecture

We do not work with shared server/resources, so it means that each propfirm has its own dedicated main server with multiple nodes attached in order to ensure scalability and support unlimited growing number of users.

We provide to the propfirm two different environments:

- staging: used for testing and development purposes. Users and subscriptions created on this environment are free of charge
- production: used for real customers

Even if we take care of all the infrastructure, the provide a white label solution, which includes:

- custom logo on the dashboard/portal
- our servers can be pointed by a your domain name

## ENTITIES

**Users**

On the system there is a distinguish between normal users(customers) and propfirm's employess or managers. The second one can be divided into the following roles:

- viewer: readonly user
- manager: can see accounts details and eventually close orders and positions
- administrator: full control on symbols, users, rules and so on

When a new challenge is bought on propfirm's side, the first entity to create is the user, since subscription and trading accounts must be attached to a specific user.

The information provided on user creation are also used for pre-configure the onboarding data feed agreement for exchange reporting, so the following information should be provided:

- Name
- Surname
- Email (required field)
- Mobile phone
- Address
- Postal code
- City
- State
- Country
- Language (optional): ISO 639/1 code

Propfirm can force a password when the user is created or can leave an auto generated password on our side. The first approach is suggested in which way the user can have only one password for accessing both portals and trading platforms.

It is **suggested** to use a **random generated password**. Regarding safety you don't have to be worried, since passwords on our DB are stored with hashing and cryptography.

IMPORTANT: the users, in order to access to their trading platforms, have to use the same password of our dashboard/portal or provided by you through API.

**Subscriptions – market data**

The subscriptions can be managed with **two different approaches**:

1. The first one is when subscriptions are directly managed from the propfirm. In this case it is care of the propfirm enabling a subscription when an account is created or disabling a subscription when there are no any active accounts anymore.
   This approach has the pros that you can manage different market data exchange activation or active a particular market for a certain amount of time, but on the other side you have to be sure to not leave active subscriptions without any active accounts underground, otherwise you will pay unnecessary monthly fees
2. the second one is where subscriptions are transparent to the propfirm and they are internally managed by our backend. In this case a subscription **is always active until at least ONE trading account is ACTIVE**, while when there is **no any trading account active** anymore

for the user, it becomes **CANCELLED**. The subscription **can be resumed** creating a new trading account or reactivating the old account.
On account creation, it can be setted the market data entitlements, while other changes must be done directly from admin portal.
This approach has the the pros that you don't risk to leave active subscriptions without any active trading account, but it has the cons that you have less control over subscription, so you cannot for example manage through API the activation of the market data for a particular exchange for a limited time

IMPORTANT: Subscriptions can be cancelled and resumed multiple time in the same calendar month **without** paying any extra fees.

Based on the requirements of the propfirm, we configure on the solution different market data exchanges or products and the list/details will be directly provided to the propfirm via email.

Unless otherwise agreed, each subscription is unique for trading terminals/platforms, which means that when the subscription is created, it must also be setted which trading platform the user want to use. If the user wants to change the trading platform during the calendar month, propfirm's admins can change the platform from admin dashboard, but the monthly fees will be charged double.

Right now, trading platforms supported by the backend are:

- Volumetrica Trading: VolSys, VolBook
- ATAS
- Quantower

The relationship between user and subscriptions is 1-1, which means that user only has one subscription for whole its lifetime. Even if a subscription is cancelled/disabled, it must be activated again or resumed always the same subscription.

During the lifetime of a subscription, the subscriptions can have the following values:

- Disabled: it is when a subscription is created without activation or when it is disabled
- Scheduled: used on some specifics scenario. For example when it is required to schedule the start of the subscription on the begin of next calendar month. It can be done specifying a start date on the subscription level
- UserOnHold: the first time that a subscription is created, the user/customer must sign, through a link, dxFeed's data subscription agreement and certify himself as a no-professional or professional user. Subscription assumes this status until the user sign the agreement
- PropfirmOnHold: if the user certify himself as a PROFESSIONAL USER, it is required an extra confirmation by the propfirm that can be done both from API or admin dashboard. The reason is that the exchange fees applicable to pro users are extremely higher compared to no-pro users. This status is received through webhook in order to notify the propfirm
- Active: subscription is activated

**Trading Accounts**

Users can potentially have an unlimited number of accounts and it is not linked to the fees billed, since the fees are on the user/subscription level.

During its lifetime, the account can have the following status values:

- Initialized = Account is created, but not enabled yet. Account can persist this status if it has been setted a delayed start (not common used)
- Enabled = Account is enabled and ready for trading
- Challenge Success = if enabled on the risk/rule parameters, it means that the user has reach the run-up/profit target
- Challenge Failed = = if enabled on the risk/rule parameters, it means that user has breached one of the risk params like for example intraday drawdown, drawdown and so on
- Disabled = The account has been manually disabled by API or propfirm's admins

Risk parameters for the account are managed by the trading rule(we see it on details later).
When an account is created, it can be decided to set a dedicated trading rule(account level) with a custom drawdown and so on, or it can be used a trading rule globally defined. What it is best to use, it depends on case by case. One thing to note is that, if a global trading rule is used, **any changes to that rule will affect ALL the accounts associated** on real time.

## RISK PARAMETERS

**Symbols**

By default, the solution is provided by a preset of symbols available with default margins, commissions and maximum number of contract that a user can open.

For symbol management, propfirms have different possibilities:

- changes margins and commission for all account, editing them on symbol's admin page
- use custom margins and commissions for specific trading rule or limit the symbols available: this can be done using the "Group universe" function

Group Universes

Propfirm can create several different group universes, which overrides the list of symbol available for the accounts with custom margins and symbols.

Groups universes can be linked to many trading rules, so it means that the relationship is 1-N.

The use of group universes is not a requirement, since trading rule can also be created without any group universe associated.

Trading Rules

As written before, trading rules can be both global defined or defined on the account level. It is the entity which controls all the risk parameters for the accounts, in particular:

- Maximum Drawdown
- Run-up / Profit target
- Intraday Drawdown
- Intraday Runup
- Maximum position loss

- Maximum position gain
- Max portfolio loss
- Max portfolio gain
- Maximum daily number of trades
- Various: minimum session numbers, overnight and overweek allowed and so on

Each of the previous params can have different settings, both in term of enable or disable, values and anchor mode, but there are some common terms which have sense to explain:

- Balance: it is the closed balance of the account, so it means that it does NOT include floating/open PnL
- Equity: it is the floating balance of the account, which includes open positions

Actions available:

- None: the risk param is not enabled for the account
- Challenge fail: if triggered, the account is immediately disabled
- Flat: all open positions are flatted, but account keeps active
- Intraday disable: all open positions are flatted, the account is disabled for the CURRENT trading session. User will be able to trade again when next session opens

Values – it can be defined both or either a money amount or a % amount and it is possible to decide(selection) which of them should be used for risk evaluation

Trading rules are cyclically checked for each account with a very low frequency in order to disable the accounts that goes outside the defined risk params.

Currency rates

Right now, account can be created with USD or EUR base currency. As known, CME instruments are traded on USD, while EUREX on EUR, so it means that a conversion rate shall be applied.

Propfirm can manage this rates with different approaches:

- API: manual update of the currency rates through API
- Admin Dashboard: admins can change currency rates from dashboard.
- Automatic update: it can be decided a frequency of update, so rates are automatically updated once a day/week/month. It can also be setted a spread that should be applied.

## ADMIN DASHBOARD

Beyond the management of the entites, admins can access to all account information, which includes:

- realtime updates of accounts balance
- view of open orders and positions of specific account
- list of the historical trade for manual check of consistency or similar things
- list of the historical orders which includes the source IPs, platform and date/time. It is useful for tickets management

Furthermore, it is available a monitor page, where propfirm can see the list of all their accounts, filtered by specific trading rule, with realtime updates of balances, equities, open trades and so on.

# API

**Rest API**

Swagger Docs. It contains also parameters details:
https://dxfeed.volumetricatrading.com/swagger

All the request must be executed as follow and the result is on json format:

- Endpoint: host/api/v2/propsite/[controller]/[action?]
  E.g.: https://dxfeed.volumetricatrading.com/api/v2/propsite/user
- It must be included this header in the request:
  - x-api-key = *token provided by email*
- All responses message are wrapped in the following object:
  - success: *bool*
  - data: *object* - if successful (200 Status code) and expected for the API, it contains the object/entity
  - statusCode: *int* – sent only if unsuccessful (status code different to 200)
  - message: *string* - sent only if unsuccessful
  - details: list of *string* – sent only if unsuccessful, it provides extra details

On swagger there are descriptions of different endpoints available with also the description of the various params. If more details are required, feel free to reach us so we can help you and support the integration.

On this document, we analyze only the most important APIs available:

- User:
  - Creation - /user (POST):
    - It must be provided all the user details(name, surname and so on). They are needed for market data agreements as explained before
    - Parameters:
      - "encryptionMode" – if password is encrypted on API exchanges:
        - 0 = None encryption
        - 1 = AES_256 encryption – key is provided via mail
      - "forceNewPassword": it is setted at false by default. A password is ALWAYS generated if the user is new. It may be setted to true if it is required a new password on user update
    - Returns:
      - "userId:" it **must be stored** on the propfirm's DB because it is necessary for the next calls
      - password: it contains the password just created. It is encrypted with the same approach of "encryptionMode"
    - IMPORTANT: each propfirm can decide if they want to send the login credentials via email to the user or not. Two different approaches can be used:
      - Propfirm stores the users' passwords on its database and it shows them in its dashboard
      - Password is sent via email from our backend to each user and the propfirm doesn't need to store them.

- /user/loginurl (POST): it can be used to generate a OTP url, which redirects the user to our dashboard. In this case, the user will be automatically logged in. It can be useful for putting a button on propfirm's customer area

- Subscriptions – if it is used the subscription transparent approach, this endpoints are NOT available:
  - /subscription (GET):
    - it retrieves the user's subscription.
    - Parameters:
      - "subscriptionId": our internal id for the subscription
      - "userId": our internal id of the user
  - /subscription (POST):
    - it creates a new subscription for the user
  - /subscription (PUT) :
    - it updates an existing subscription. The subscription id must be sent on the query string
  - Common body for both POST and PUT request – important fields:
    - "userId": it is the id returned on the user creation
    - "enabled": if the subscription should be switch to enable status
    - "startDate", "durationMonths" and "durationDays" are optional
    - "dataFeedProducts" – list of integers: each value corresponds to a different exchange data entitlements. Values changes based on the propfirm requirements, but just to understand we make some examples:
      - 0 = CME L1
      - 1 = COMEX L1
      - 2 = CME L2
    - "platform":
      - 0 = VOLUMETRICA_TRADING:
        - if this platform is selected, user can chose which platform to use(VolSys or VolBook) from the portal or a default platform can be setted through "volumetricaPlatform" field
      - 1 = ATAS
      - 2 = QUANTOWER
  - IMPORTANT: "dataFeedProducts" **should not be changed** during the duration of the challenge. It can be changed, but based on the scenario, **you could be liable to EXTRA fees, so it should be changed as lowest as possible.**
  - /subscription (DELETE):
    - avoid the use of this endpoint, please just disable a subscription if not needed. It is obsolete and keeped only for backward compatibility
  - /subscription/active (POST):
    - active a subscription without changing market data products, platforms or linked params
  - /subscription/deactive (POST):
    - disable a subscription without changing market data products, platforms or linked params
  - /subscription/confirm (POST):

        - if the status of the subscription is PropfirmOnHold, which means as explained before that subscription must be confirmed, this is the endpoint to use
        - Parameters:
            - “subscriptionId”: id of the subscription
            - “confirmationId”: received through webhook or it can be retrieved by GET endpoint
    - Important fields available on the response or webhook body received:
        - Status available:
            - 0 = Disabled
            - 1 = Active
            - 2 = Scheduled
            - 3 = UserOnHold
            - 4 = PropfirmOnHold
            - 5 = dxFeed Error
        - dxFeedAgreementLink: when the status is “UserOnHold“, it contains the link that user must sign for accessing real time data. Propfirm may decide to send it also through email. Please let us know if you want to use this approach, since it is needed to verify the text of the email in order to follow the exchange compliance.
        - confirmationId: if status is “PropfirmOnHold”, it contains the confirmationId to use for confirm the activation of the subscription. Requred only for professional user
- Trading Accounts:
    - /tradingAccount (GET):
        - it returns the real time information for the trading account. Look at swagger for field available, they should be intuitive
    - /tradingAccount (POST):
        - it creates a new trading account.
        - Params that need an extra explanation:
            - “userId”: it is the id returned on the user creation
            - “currency”:
                - 0 = EUR
                - 1 = USD (we suggest to use this)
            - “header”: default is header is something like: DFX-CUM_NUMBER. Setting this field, it can be overridden.
            - “description”: custom description that it will be shown on the dashboard
            - “disableOtherAccountsEnabled”: if true – it disables all the other account enabled
            - Account expirations:
                - “expirationMode”:
                    - 0 = never expires
                    - 1 = use “endDate” sent
                    - 2 = use “expirationDays” and it adds them from activation date/time

- 3 = use “expirationDays” and it adds them from the first order inserted date/time
- 4 = use “expirationDays” and it adds them from the first order executed date/time

- Trading rules:
  - Global level: it must be setted the “accountRuleId” param
  - Account level: it must me populated “accountCustomRule”, setting all the parameters
- “subscriptionDetail”:
  - if you directly manage the subscription, do NOT use it. It is used only on transparent subscription management
  - “dataFeedProducts” – list of integers: each value corresponds to a different exchange data entitlements. Values changes based on the propfirm requirements, but just to understand we make some examples:
    - 0 = CME L1
    - 1 = COMEX L1
    - 2 = CME L2
  - “platform”:
    - 0 = VOLUMETRICA_TRADING:
      - if this platform is selected, user can chose which platform to use(VolSys or VolBook) from the portal or a default platform can be setted through “volumetricaPlatform” field
    - 1 = ATAS
    - 2 = QUANTOWER

- Important fields returned:
  - “accountId” returned must be stored for next calls or for having reference on webhook notifications
  - “tradingRuleId”: if it has been setted the trading rule on the account level, it is generated a new id and it may be used later for changing params

- /tradingAccount/Enable (POST):
  - it enables a trading account
- /tradingAccount/Disable (POST):
  - it disables a trading account. If you want to force the disable, set “forceClose” to true, otherwise if it has open positions and the market is closed, an exception can be thrown
  - send also a “reason”, so the user can see it on the dashboard. It may avoid unnecessary tickets
- /tradingAccount/ChangeStatus (POST):
  - if a disable is requested and you want to force the disable, set “forceClose” to true, otherwise if it has open positions and the market is closed, an exception can be thrown
  - send also a “reason”, so the user can see it on the dashboard. It may avoid unnecessary tickets
  - “status”: new status for the account (list below)
- /tradingAccount/changeTradingRule (POST):

        - change the trading rule for the account
        - IMPORTANT: it cannot be used if the rule used by the account is on the account level
    - /tradingAccount (DELETE):
        - it cannot be used on active accounts
    - Important fields available on the response or webhook body received:
        - “accountId:” internal id of the account. It must be used for next calls
        - “status” – see description on entity section
            - 0 = Initialized
            - 1 = Enabled
            - 2 = ChallengeSuccess
            - 4 = ChallengeFailed
            - 8 = Disabled
        - “reason”: if account is disabled, it provides the reason of the disable. Possible values are:
            - "TRADING_RULE_BALANCE";
            - "TRADING_RULE_RUNUP";
            - "TRADING_RULE_MAX_DD";
            - "TRADING_RULE_MAX_INTRADAY_DD";
            - "TRADING_RULE_MAX_INTRADAY_RUNUP";
            - "TRADING_RULE_MAX_POS_OPEN_LOSS";
            - "TRADING_RULE_MAX_POS_OPEN_GAIN";
            - "TRADING_RULE_MAX_OPEN_LOSS";
            - "TRADING_RULE_MAX_OPEN_GAIN";
            - "TRADING_RULE_MAX_NUMBER_TRADES";
            - "TRADING_RULE_OVERNIGHT";
            - "TRADING_RULE_OVERWEEK";
- Trading rules:
    - they can be created or edited also through API
    - /tradingRule (GET):
        - it returns the trading rule params
    - /tradingRule (POST):
        - create new trading rule globally defined
        - update an existing trading rule: it can be both global or account level
    - /tradingRule/validate (POST):
        - it checks if the params are correctly setted. If necessary, it must be call before the creation or update of the trading rule
    - /tradingRule/changeGroupUniverse (POST):
        - it changes the group universe associated to a trading rule

## Webhooks

We strictly suggest the integration of webhooks technology, because it both helps propfirm to get notified on realtime when something happens and our resources to have a lower overhead compared to polling all the accounts to get changes.

The architecture is structured to be consistent, which means that failed webhooks are sent until we do not receive an HTTP 200 status code, so couple of considerations:

- you do not have any risk of loosing information during update or downtime
- they are sent sequentially in order to rebuild the changes of status in a chronological order
- if you have revieved the request, please always answer 200 OK, even if some exceptions are thrown, otherwise all other webhooks are blocked

Propfirm should provide an url endpoint where they can receive the HTTP requests. Requests will be sent using the following token:

- x-api-key: token provided via email. Usually it is the same of token used for Rest API requests

At the moment, they are available the following categories of notifications:

- 0 = Accounts:
    - Events available:
        - 0 = Created
        - 1 = Update – it can be configured to receive the following updates:
            - account status changes: it is enabled by default. It is sent a notification for example when an account switches from “Enable” to “Challenge failed”
            - balance changes: it can be enabled on request, an event it is sent each time that a transaction is closed on the account
        - 2 = Delete
- 1 = Positions
    - Events available:
        - 3 = Overnight: it notifies overnight positions for the account. Some propfirm do not want to disable account as default on overnight, but managed it on their side
- 2 = Subscriptions:
    - Events available:
        - 0 = Created
        - 1 = Update:
            - it is important to handle this event, because you are notified when the user sign the data feed agreements and if he has certified himself as pro or no pro
        - 2 = Delete
- 3 = Trade Report – it notifies an close transaction with PnL informations
    - it is enabled on request
    - Events available:
        - 0 = Created

For development or debug, propfirm's admins can see webhook generated on the admin portal with also the JSON body generated.

If you need some examples for specific situations, please directly ask us and we will provide JSON payloads example.

Last thing: on swagger there is an endpoint which allows you to see what it is received through webhook, in particular: /webhook/GetModel