ООО «Дольмен»

ПРОГРАММА ДЛЯ ЭВМ
«LiteBot»

Документация, содержащая информацию,
необходимую для установки и эксплуатации
экземпляра программного обеспечения

Версия 0.2.0

Санкт-Петербург, 2026

Содержание

1. Общие положения.....3
1.1. Общие сведения о программе.....3
1.2. Информация, необходимая для установки и настройки.....3
1.3. Описание структуры.....4
2. Подготовка к работе.....9
3. Описание работы.....11
3.1. Установка и настройка ПО.....11
3.2. Установка обновлений.....12
3.3. Штатное функционирование.....12
3.4. Резервное копирование и восстановление данных.....13
3.5. Проведение диагностики.....14

1. Общие положения

1.1. Общие сведения о программе

Программа для ЭВМ «LiteBot» (далее — программа) предназначена для автоматизированной торговли на ПАО Московская Биржа (далее — биржа): приёма рыночных данных, исполнения торговых алгоритмов (стратегий), выставления и сопровождения заявок, учёта позиций и финансового результата, контроля рисков. Программа применяется профессиональными участниками рынка ценных бумаг, их клиентами и иными лицами, допущенными к торгам на бирже, на фондовом, валютном и срочном рынках биржи.

Программа устанавливается на сервер пользователя под управлением ОС Linux и работает как системная служба. Графического интерфейса программа не имеет: оператор и администратор работают с ней через программный интерфейс управления по протоколу HTTP (JSON), командную строку и файлы журналов.

Правообладатель — ООО «Дольмен». Документация программы размещена на сайте правообладателя: dolmenfintech.ru.

1.2. Информация, необходимая для установки и настройки

Установку и настройку программы выполняет администратор пользователя по настоящему документу; при необходимости правообладатель оказывает консультации (раздел 3.5). Для установки необходимы:

- комплект поставки программы (предоставляется правообладателем по договору);
- сервер, соответствующий требованиям раздела 2;
- доступы к бирже: подключение к сервисам рыночных данных и приёма заявок, логины и пароли шлюзов, торговый счёт и код клиента (предоставляются биржей или брокером);
- для подключения к боевым шлюзам — пройденная сертификация программно-технических средств в порядке, установленном биржей.

Состав комплекта поставки:

Состав	Назначение
`litebot`	Исполняемый файл программы (Linux x86_64 или aarch64, статически скомпонован)

Состав	Назначение
`twime-mock`	Имитатор биржевого шлюза приёма заявок для проверки установки и репетиции сертификации
`resources/moex/`	Схемы протоколов биржи (публикуются биржей, загружаются с сайта биржи при сборке комплекта)
`configs/`	Примеры конфигурации: боевой запуск, проверка, сценарии сертификации
`deploy/ansible/`	Сценарии автоматизированной установки и сопровождения
`bench/hot_path`	Средство замера времени обработки на сервере пользователя (запускается вне торговых часов, сценарий `deploy/ansible/bench.yml`)
`VERSION`	Версия и идентификатор сборки
`LICENSE`	Условия использования программы
`THIRD-PARTY-NOTICES.md`	Лицензии сторонних компонентов

Комплект предоставляется правообладателем по договору. Версию установленного файла показывает `litebot --version`; она совпадает с файлом `VERSION`.

Комплект разработчика (SDK) поставляется отдельно, для каждой архитектуры:

Состав	Назначение
`lib/liblitebot_engine.a`	Программа в виде статической библиотеки
`litebot-sdk/`	Программный интерфейс стратегий (исходный текст)
`examples/`	Учебные примеры: лесенка заявок, маркет-мейкер Авельянеды–Стойкова, граф признаков
`vendor/`, `docker/`	Зависимости интерфейса и примеров, образ сборки: сборка без сети
`bin/twime-mock`, `deploy/`, `scripts/`, `docs/`	Имитатор шлюза, сценарии установки, загрузка схем биржи, документация
`MANIFEST.sha256`, `VERSION`	Контрольные суммы и версия комплекта

1.3. Описание структуры

Программа обеспечивает выполнение следующих функций:

1) Приём рыночных данных биржи, построение книг заявок, ведение справочника инструментов.
2) Выставление, изменение и снятие заявок на бирже, ведение сессий со шлюзами приёма заявок.

3) Исполнение торговых стратегий в портфелях; изменение параметров стратегий во время торгов.
4) Учёт позиций и финансового результата, сохранение и восстановление позиций при перезапуске.
5) Риск-контроль: проверка заявок до отправки, ограничения счёта и портфелей, стоп-лоссы, аварийное снятие заявок.
6) Ручная торговля оператора.
7) Управление и контроль состояния через программный интерфейс.
8) Журналирование обмена с биржей и событий программы, метрики мониторинга.
9) Режимы проверки: демонстрационный режим, проигрывание записей рынка, имитатор шлюза биржи, проверка конфигурации, сценарии сертификации.
10) Подключение собственных стратегий пользователя через комплект разработчика.

Основные понятия

Понятие	Значение
Площадка	Рынок биржи: фондовый и валютный (ASTS) или срочный (SPECTRA)
Инструмент	Ценная бумага, валютная пара или контракт. Записывается режимом торгов и кодом (`TQBR:SBER`), кодом контракта (`Si-12.26`) или правилом выбора контракта (`Si.IMM1` — ближайший квартальный)
Стратегия	Торговый алгоритм, подключённый к программе. В поставку входит демонстрационная стратегия `ladder`
Портфель	Запущенная стратегия со своими инструментами, параметрами, позициями и ограничениями. У каждого портфеля своё имя
Деск	Встроенный портфель для заявок оператора
Токен	Секретная строка, которой оператор подтверждает право на запрос к интерфейсу управления
Признак (фича)	Величина, которую стратегия вычисляет из рыночных данных по выражению из конфигурации, например сглаженный дисбаланс объёмов лучших цен

Командная строка

Команда	Назначение
`litebot run --config <файл>`	Торговля: подключение к бирже, запуск портфелей из конфигурации, интерфейс управления
`litebot check --config <файл>`	Проверка конфигурации, схем, адресов и портфелей без подключения к бирже

Команда	Назначение
`litebot demo --config <файл> --market <файл>`	Демонстрационный режим: синтетический рынок, имитатор шлюза, интерфейс управления. Подключение к бирже не нужно
`litebot replay --config <файл> --pcap <запись>`	Проигрывание записанного рыночного трафика; заявки исполняет имитатор
`litebot cert --config <файл> --scenario <сценарий>`	Сценарий сертификации на тестовом полигоне биржи или на имитаторе шлюза
`litebot logs parquet --dir <каталог журналов>`	Перевод журналов событий, заявок и трейсов циклов в формат Parquet
`litebot logs latency --dir <каталог журналов>`	Отчёт о времени обработки за торговый день: перцентили по стадиям, по часам и по инструментам
`litebot --version`, `litebot --help`	Версия программы, справка по командам и параметрам
`twime-mock`	Имитатор шлюза приёма заявок биржи, запускается отдельно

Код возврата 0 означает успешное завершение. Иной код означает ошибку, её описание выводится в стандартный поток ошибок.

Конфигурация

Конфигурация хранится в файле формата TOML. Неизвестный параметр считается ошибкой: программа не запускается и называет его. Образец с комментариями — `configs/live-example.toml`.

Раздел	Содержание
`[engine]`	Ядра процессора, приоритет торгового потока, служебные периоды, время штатной остановки, уровень журнала
`[asts]`, `[spectra]`	Площадки: включение, схемы протоколов, адреса рыночных данных, сетевой интерфейс, порог отсутствия данных, торговые сессии
`[asts.twime]`, `[spectra.twime]`	Шлюзы приёма заявок: адреса, учётные данные, счёт, код клиента, снятие заявок при запуске, восстановлении связи и остановке
`[risk]`, `[risk.limits]`, `[risk.guard]`	Риск-ограничения счёта и портфелей, стоп-лоссы
`[secdef]`	Действие при расхождении справочника биржи с ожидаемыми параметрами инструмента
`[control]`	Адрес интерфейса управления и токены
`[log]`, `[persist]`	Каталог и состав журналов, формат журнала событий, файл позиций
`[prometheus]`, `[kafka]`	Экспорт метрик и выгрузка событий. По умолчанию выключены
`[[instruments]]`, `[[portfolios]]`	Инструменты и портфели, которые запускаются при старте

Адреса рыночных данных и шлюзов берутся из действующих документов подключения биржи. Пароли и токены хранятся вне системы контроля версий; при установке через Ansible — в зашифрованном хранилище.

Для торговли на боевых шлюзах программа требует:

– ограничения риска на счёт;
– токены длиной не менее 16 символов, если интерфейс управления доступен не только с локального адреса.

Интерфейс управления

Интерфейс работает по HTTP, запросы и ответы передаются в формате JSON. Каждый запрос, кроме проверок работоспособности, содержит заголовок `Authorization: <токен>` или `Authorization: Bearer <токен>`. Операции, которые снимают защитные ограничения, требуют токена администратора, если он задан.

По умолчанию интерфейс доступен только с локального адреса сервера. Удалённый доступ — через SSH-туннель или через порт, открытый межсетевым экраном для доверенной сети.

Торговый запрос возвращает ответ после того, как программа его исполнила. С параметром `wait_ms` ответ возвращается после ответа биржи. Ошибка возвращается с кодом 4xx и полем `detail`, в котором указана причина.

Портфели

Запрос	Действие
`POST /start`	Запустить портфель: вид стратегии, имя, инструменты, параметры
`POST /stop?portfolio=<имя>`	Остановить портфель и снять его заявки
`POST /portfolios/stop`	Остановить несколько или все портфели
`POST /pause?portfolio=`, `POST /resume?portfolio=`	Приостановить и возобновить портфель. Возобновление требует токена администратора, если он задан
`POST /control`	Передать стратегии сообщение оператора, например новые уровни
`GET /portfolios/status`, `GET /portfolio?portfolio=`	Состояние портфелей, позиции, результат, рабочие заявки
`GET /results?portfolio=`	Итоговые показатели стратегии

Запрос	Действие
`GET /strategies`	Виды стратегий, доступные в исполняемом файле

Пример запуска демонстрационной стратегии:

```
curl -s -H "Authorization: $TOKEN" -X POST localhost:3000/start -d
'{"kind":"ladder","name":"SBER","instruments":["TQBR:SBER"],"config":
{"instrument":0,"precision":2,"step_num":3,"step_size":0.5,"qty_buy":1,"qty_sell":1,
"lim_buy":300,"lim_sell":305}}'
```

Состояния портфеля: запускается, работает, на паузе, останавливается, остановлен, ошибка. Причина паузы или ошибки указана в состоянии портфеля.

Ручная торговля

Запрос	Действие
`POST /orders`	Заявка оператора: инструмент, сторона, тип (лимитная, рыночная, «исполнить немедленно или отклонить», только пассивная), цена, объём. Заявку можно подать от имени портфеля
`POST /orders/modify`	Изменить цену или объём заявки оператора
`POST /orders/cancel`	Снять заявки по номерам или по фильтрам: портфель, инструмент, сторона
`POST /positions/close/request`, `POST /positions/close`	Сформировать заявки на закрытие позиции портфеля и исполнить их после проверки оператором
`GET /orders`, `GET /positions`	Заявки и позиции

Ручные заявки проходят те же проверки и риск-ограничения, что и заявки стратегий.

Риск

Запрос	Действие
`GET /risk`	Состояние ограничений счёта и портфелей, сработавшие правила
`POST /risk/reset`	Снять сработавший стоп-лосс. Требует токена администратора
`POST /cancel_all`	Аварийно снять все заявки и приостановить все портфели
`POST /positions/reset`	Обнулить учётные позиции. Требует токена администратора

Состояние и мониторинг

Запрос	Ответ
`GET /health`	Процесс работает (200) или нет (503)
`GET /health/ready`	Программа готова к торговле

Запрос	Ответ
`GET /status`	Полное состояние: портфели, инструменты, заявки, позиции, сессии
`GET /pnl`, `GET /exposure`	Результат и экспозиция по портфелям и инструментам
`GET /book?instrument=<инструмент>&depth=<уровней>`	Книга заявок инструмента: ценовые уровни с объёмом и числом заявок. Полная книга — для инструментов, по которым она ведётся, для остальных — лучшие цены
`GET /metrics`, `GET /metrics/prometheus`, `GET /latency`	Счётчики, время обработки, метрики в формате Prometheus
`GET /host`	Результаты проверки сервера при запуске
`GET /logs?n=`	Последние строки журнала

Отдельный порт метрик (раздел `[prometheus]`) защищается своим токеном. Система мониторинга передаёт его в заголовке `Authorization: Bearer <токен>`.

Журналы

Файл	Содержание
`orders-<дата>.log`	Журнал заявок: каждый запрос на биржу, каждый отказ программы с причиной, каждый ответ биржи, сессии, срабатывания риск-контроля. Новый файл каждые сутки
`engine.log`	Журнал работы программы в читаемом виде
`events.jsonl`	События программы, по одному объекту JSON в строке
`events-<дата>T<час>.parquet`	События программы в колоночном формате Parquet, файл на час. Пишется, если в разделе `[log]` параметр `events` равен `"parquet"` или `"both"`
`results-<портфель>.json`	Итоговые показатели стратегии при остановке портфеля, если задан каталог результатов
`positions.json`	Позиции портфелей для восстановления при запуске
журнал обмена со шлюзом	Кадры обмена со шлюзом приёма заявок в формате, установленном биржей

Каталог журналов задаётся в разделе `[log]`. Срок хранения журнала заявок пользователь устанавливает по своим внутренним правилам и требованиям регулятора.

Демонстрационная стратегия ladder

Стратегия выставляет до `step_num` заявок на покупку ниже уровня `lim_buy` и до `step_num` заявок на продажу выше уровня `lim_sell` с шагом `step_size`, объёмом `qty_buy` и `qty_sell` лотов. Оператор меняет уровни во

время торгов сообщением `POST /control` с полями `lim_buy` и `lim_sell`; поле `active: false` прекращает выставление новых заявок. Модели цены в стратегии нет: она служит для проверки программы и обучения операторов.

Собственные стратегии

Собственные стратегии пишутся на языке Rust с помощью комплекта разработчика (SDK), который предоставляется по договору с правообладателем. Комплект содержит программу в виде статической библиотеки, программный интерфейс стратегий в исходном тексте, библиотеку признаков (выражения над рыночными данными, пересчитываемые при каждом событии), учебные примеры, образ сборки и руководство разработчика `docs/sdk.md`.

Исполняемый файл со стратегиями пользователя собирается командой `make build` (в образе сборки — `make docker-build`) и имеет ту же командную строку, конфигурацию, интерфейс управления и журналы, что `litebot`; в `--version`, первой строке `engine.log` и ответе `GET /health` он выводит версию программы и сборку пользователя. Команда `make dist EXAMPLE=<пример>` готовит каталог поставки, который устанавливается так же, как `litebot` (раздел 3.1). Для сборки нужен Docker (образ сборки из комплекта) или Rust не ниже 1.85; сеть нужна только для загрузки схем биржи. Учебные примеры не являются инвестиционными рекомендациями.

2. Подготовка к работе

Требования к серверу

Параметр	Требование
Операционная система	Linux с systemd: RHEL 9 и совместимые (Rocky Linux, AlmaLinux, Oracle Linux), а также российские дистрибутивы Linux того же поколения
Архитектура	x86_64 или aarch64
Процессор	Не менее 4 ядер; для торговли рекомендуется выделить одно ядро торговому потоку
Память	Не менее 8 ГБ; разрешение на блокировку страниц в памяти
Диск	Не менее 50 ГБ под журналы (объём зависит от срока хранения)
Сеть	Подключение к программно-техническим средствам биржи: приём многоадресной рассылки рыночных данных и TCP-соединения со шлюзами приёма заявок
Время	Синхронизация часов: chrony или NTP; для измерения задержек — PTP

Рабочее место администратора

Для автоматизированной установки: ansible-core 2.14 или новее, коллекция `ansible.posix`, доступ по SSH к серверу с правами sudo.

Среда разработки и сборки программы

Программа разработана на языке Rust. Сборка выполняется в изолированной среде на ОС Linux (образ сборки Docker с закреплённой версией компилятора) для целевых платформ x86_64 и aarch64; результат — статически скомпонованный исполняемый файл, не зависящий от системных библиотек.

Проверка комплекта поставки

Перед установкой рекомендуется проверить комплект без подключения к бирже:

```
./litebot --version
./litebot --help
```

Проверка без подключения к бирже — репетиция сценариев сертификации против встроенного имитатора шлюза (рыночные данные не нужны):

```
./litebot cert --config configs/cert-spectra.toml --scenario
configs/cert/spectra.toml --oms mock --time-scale 0.02 --no-control --set
instruments.futures.security_id=5215853 --set params.futures_price=73940 --set
instruments.option.security_id=240193 --set params.option_price=500
```

```
./litebot cert --config configs/cert-asts.toml --scenario configs/cert/asts.toml --
oms mock --time-scale 0.02 --no-control --set params.price=223.74 --set
params.new_password=Qwerty12
```

Команды выполняются из корня комплекта. Программа проходит шаги сценария, печатает итог по каждому шагу и пишет отчёт в `out/`. Параметры `--set` задают инструменты и цены тестового сценария. Ожидаемый итог:

– срочный рынок — 19 шагов пройдено, 0 с ошибкой, 2 шага без проверяемого результата;
– фондовый рынок — 16 шагов пройдено, 0 с ошибкой.

Демонстрационный режим запускает программу целиком на синтетическом рынке, с имитатором шлюза и интерфейсом управления на `127.0.0.1:3000`. Токены заданы в `configs/demo.toml`:

```
./litebot demo --config configs/demo.toml --market configs/demo/market.toml
```

Служба для постоянного стенда — `deploy/demo/litebot-demo.service`.

3. Описание работы

3.1. Установка и настройка ПО

Автоматизированная установка

Шаг 1. Распаковать комплект на рабочем месте администратора и перейти в `deploy/ansible`. Сценарии берут исполняемые файлы и схемы из корня комплекта.

Шаг 2. Установить коллекцию: `ansible-galaxy collection install ansible.posix`.

Шаг 3. Перечислить серверы в `inventory.ini` (образец — `inventory.example.ini`).

Шаг 4. Заполнить параметры в `group_vars/litebot.yml` (образец — `group_vars/litebot.example.yml`): площадки, адреса рыночных данных и шлюзов из документов подключения биржи, счёт, риск-ограничения, портфели.

Шаг 5. Создать зашифрованное хранилище паролей: `ansible-vault create group_vars/vault.yml` — пароли шлюзов и токены программного интерфейса управления.

Шаг 6. Установить без запуска:

```
ansible-playbook site.yml --ask-vault-pass -e litebot_start=false
```

Сценарий создаёт системного пользователя без командной оболочки, каталоги программы (`/opt/litebot`), конфигурации (`/etc/litebot`), данных и журналов (`/var/lib/litebot`, `/var/log/litebot`), настраивает параметры ядра и сети для приёма рыночных данных, ротацию журналов, межсетевой экран, устанавливает файлы, формирует и проверяет конфигурацию и регистрирует службу systemd `litebot`.

Шаг 7. Если в параметрах задана изоляция ядер процессора, перезагрузить сервер.

Шаг 8. Запустить и проверить:

```
ansible-playbook start.yml
ansible-playbook status.yml
```

Установка считается успешной, когда проверка готовности (`GET /health/ready`) возвращает код 200: сессии со шлюзами установлены, рыночные данные поступают.

Установка вручную

Шаг 1. Создать пользователя `litebot` и каталоги `/opt/litebot/bin`, `/opt/litebot/resources`, `/etc/litebot`, `/var/lib/litebot/out`.

Шаг 2. Скопировать `litebot` и `twime-mock` в `/opt/litebot/bin`, `resources/` в `/opt/litebot/resources`.

Шаг 3. Составить конфигурацию `/etc/litebot/litebot.toml` по образцу `configs/live-example.toml` (права 0640, владелец root, группа litebot).

Шаг 4. Проверить конфигурацию:

```
/opt/litebot/bin/litebot check --config /etc/litebot/litebot.toml
```

Шаг 5. Создать службу systemd, запускающую `litebot run --config /etc/litebot/litebot.toml` от пользователя `litebot` с правами `CAP_SYS_NICE`, `CAP_IPC_LOCK` и без ограничения блокируемой памяти; перед запуском — `litebot check`.

Шаг 6. Включить и запустить службу: `systemctl enable --now litebot`.

3.2. Установка обновлений

Обновления не устанавливаются автоматически: пользователь сам решает, когда устанавливать новую версию. Программа не обращается к серверам правообладателя и не может управляться ими.

```
ansible-playbook deploy.yml --ask-vault-pass
```

Новый исполняемый файл проверяется на установленной конфигурации, предыдущий сохраняется, служба перезапускается. При

перезапуске программа снимает заявки счёта и восстанавливает позиции портфелей. Обновление рекомендуется выполнять вне торговой сессии.

Откат к предыдущей версии:

```
ansible-playbook rollback.yml --ask-vault-pass
```

Возвращает предыдущую версию исполняемого файла и перезапускает службу.

Удаление программы:

1) Остановить службу: `systemctl disable --now litebot`.
2) Удалить файл службы, каталоги `/opt/litebot`, `/etc/litebot`; журналы `/var/lib/litebot`, `/var/log/litebot` — после истечения срока их хранения по внутренним правилам пользователя.
3) Удалить пользователя `litebot`.

3.3. Штатное функционирование

Запуск: `systemctl start litebot` или `litebot run --config <файл>`. Программа:

1) печатает версию;
2) проверяет сервер и конфигурацию;
3) восстанавливает позиции;
4) устанавливает сессии со шлюзами и снимает заявки счёта, оставшиеся с прошлого запуска;
5) запускает портфели, когда получен справочник по их инструментам.

Готовность: `GET /health/ready` возвращает 200, когда сессии установлены, данные поступают и ни один портфель не находится в состоянии ошибки.

Остановка: `systemctl stop litebot`, сигнал SIGTERM или запрос `POST /shutdown`. Портфели снимают свои заявки, шлюзы выполняют страховочную массовую отмену и закрывают сессии, журналы и позиции сохраняются. Время остановки ограничено; повторный сигнал завершает её немедленно.

Позиции при остановке не закрываются. При следующем запуске они восстанавливаются.

Оператор управляет программой через интерфейс управления (раздел 1.3). Рекомендуемые оповещения системы мониторинга:

- проверка работоспособности не проходит;
- растёт число отказов по токену (`litebot_control_auth_failures_total`);
- сессия шлюза разорвана в торговое время;
- рыночные данные не поступают;
- сработал стоп-лосс или запрет торговли;
- есть потери записей журнала.

Сертификация на бирже

Подключение к боевым шлюзам биржи допускается после сертификации программно-технических средств участника. Программа содержит сценарии сертификации для обеих площадок. Команда `litebot cert` выполняет шаги сценария на тестовом полигоне биржи, записывает обмен со шлюзом в формате, требуемом биржей, и позволяет повторить отдельные шаги вручную. До выхода на полигон сценарий проверяется на имитаторе шлюза (`--oms mock`).

3.4. Резервное копирование и восстановление данных

Программа не использует базу данных. Для восстановления работы достаточно сохранять:

- файл конфигурации (`/etc/litebot/litebot.toml`) и хранилище паролей и токенов;
- файл позиций портфелей (`positions.json` в каталоге данных программы);
- журналы заявок и событий — в течение срока хранения, установленного внутренними правилами пользователя и требованиями регулятора.

Резервное копирование выполняется средствами, принятыми у пользователя. При обновлении предыдущая версия исполняемого файла сохраняется автоматически; возврат к ней — сценарием отката (раздел 3.2). Позиции портфелей восстанавливаются при каждом запуске из файла позиций; для восстановления на новом сервере достаточно установить программу (раздел 3.1) и вернуть конфигурацию и файл позиций.

3.5. Проведение диагностики

Диагностика выполняется по журналам программы, журналу службы (`journalctl -u litebot`), запросам состояния интерфейса управления (`GET /status`, `GET /host` — результаты проверки сервера при запуске) и метрикам. Удалённого доступа правообладателя к программе нет; диагностика выполняется администратором пользователя, при необходимости — с консультацией правообладателя по журналам.

Ситуация	**Действие**
Программа не запускается	Выполнить `litebot check` и исправить конфигурацию по сообщению об ошибке; проверить журнал службы (`journalctl -u litebot`)
Портфель не стартует	Причина указана в `GET /portfolios/status`: ожидание справочника, расхождение параметров инструмента, ошибка параметров стратегии или выражения признака
Портфель на паузе	Причина указана в состоянии портфеля: нет свежих данных, торги инструментом приостановлены, смена торговой сессии, команда оператора. Паузы по причинам биржи снимаются автоматически
Сработал стоп-лосс	Разобраться в причине по журналу заявок, затем выполнить `POST /risk/reset`
Разрыв сессии со шлюзом	Программа сама восстанавливает сессию, снимает заявки счёта и продолжает работу
Позиция расходится с отчётом брокера	Остановить портфель, обнулить позицию (`POST /positions/reset`), учесть расхождение вручную

Обращения принимает служба поддержки правообладателя — ООО «Дольмен»: support@dolmenfintech.ru, сайт dolmenfintech.ru. Обращение содержит версию программы (`litebot --version`), описание ситуации, время и, по возможности, фрагменты журналов. Журналы не содержат паролей и токенов.