

# ТЕХНИЧЕСКАЯ СПЕЦИФИКАЦИЯ (TRS)

**Проект:** BMV Eco System Market Making Bot

**Версия:** 2.6 (С комментариями от 9 января)

**Дата:** 09.01.2026

---

## CHANGELOG

### v2.5 → v2.6

Интеграция комментариев Михаила от 9 января

### v2.4 → v2.5

Добавление структуры модулей, ключевых ссылок, информации по токену, ручное редактирование текста, описание взаимодействий

### v2.2.1 → v2.4

Этот релиз восстанавливает структуру и полноту спецификации, а также учитывает новые требования, полученные 8 января 2026 года:

- **Pivot Point:** вернулся режим чистого VWAP - в расчёт средней цены не включаются комиссии и прочие расходы. Все затраты учётных операций фиксируются в собственной базе данных и со временем покрываются спредом.
- **Собственные записи:** для расчёта Pivot и анализа торговли используется локальная история сделок (PostgreSQL), чтобы не обращаться каждый раз к блокчейну.
- **Лимиты ордеров:** вместо жёстких 16 BUY / 16 SELL на кошельк, применяется общий лимит 32 лимитных ордеров на канал (BUY + SELL) для каждого кошелька; количество уровней на канале растёт вместе с числом кошельков. Ширина зон BUY и SELL изменяется инерционно вместе с рынком; отношение 30 % / 15 % - лишь настройка по умолчанию.
- **Rebalance Trigger:** перестройка сетки выполняется при отклонении текущей цены от центра канала более чем на заданный порог (по умолчанию 1 %); это позволяет снизить нагрузку на биржу и уменьшить риск проскальзывания. Каждая сделка больше не вызывает полный rebuild, но существует тайм-аут принудительной синхронизации (например, 1 час).
- **Fiat Manager:** концепция «Fiat Floor» заменена на динамический буфер ликвидности - бот регулирует долю SOL и USDC на верхней и нижней границах канала. В верхней части часть SOL постепенно конвертируется в USDC, а в нижней - USDC возвращается в SOL. Отношения 100/0 → 70/30 по умолчанию являются только ориентиром и регулируются в параметрах.
- **Target Control:** целевой процент контроля эмиссии (раньше 51 %) больше не жёсткая константа. Он вычисляется динамически: суммарная эмиссия минус заблокированные монеты минус токены на собственных кошельках и в открытых ордерах. При росте объёма монет у сторонних держателей бот увеличивает плотность BUY-зоны.

- **Flash Volume / Atomic Wash Trading:** для создания визуальной активности введён механизм атомарных встречных сделок. Два кошелька из пула совершают покупку и продажу одного и того же объёма SOL (по умолчанию 0.1 SOL) в одном Jito Bundle с минимальными чаевыми. Цена остаётся неизменной, а убыток равен только комиссии сети. Этот модуль заменяет прежний Market Noise и настраивается через параметры.

Дополнительно сохранены улучшения из предыдущих версий: Seeded Pivot, горизонтальное масштабирование Swarm, расширенные метрики наблюдаемости и безопасность.

## v2.1 → v2.2

Краткий список изменений между версиями 2.1 и 2.2 приведён для справки:

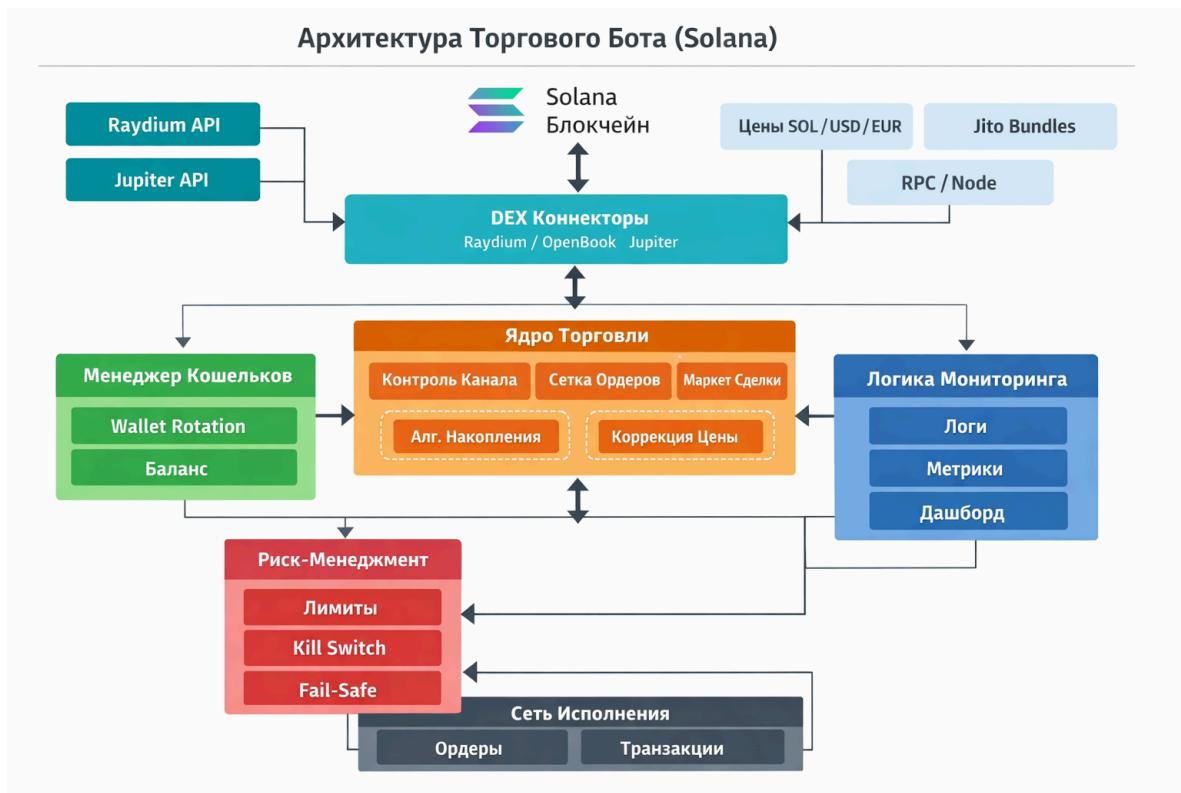
- введён режим *FULL COST BASIS* для Pivot;
- ребалансировка сделана event-driven (каждый Fill);
- контроль эмиссии вынесен в конфигурацию;
- фиксирована аксиома 32 ордера (16 BUY + 16 SELL) на кошелёк;
- улучшены разделы Observability и Swarm.

## v2.2 → v2.2.1

Основное новшество 2.2.1 – модуль Market Noise: при отсутствии активности бот выставляет чередующиеся рыночные ордера (buy/sell) через Jito Bundles. Кроме того, был добавлен Seeded Pivot (инициализация истинной средней с весом в 365 дней) и новые параметры `NOISE_ENABLED`, `NOISE_MODE`, `NOISE_ORDER_USD`, `NOISE_IDLE_INTERVAL_MIN`.

---

# 1. ВВЕДЕНИЕ И ЦЕЛИ



Система предназначена для автоматизированного маркет-мейкинга на блокчейне Solana и выполняет роль **Growth Engine**.

## Токен BMV

Проект BMV реализован на блокчейне Solana. Ниже представлены ключевые идентификаторы и ссылки, используемые для взаимодействия с токеном и рынком:

- **Название токена:** BMV Eco System
- **Адресс (Mint Address):** <8VMi93EEqSGCHGjRGNhEFBkGoNZTQxUFFFYWDEX2e5UF>
- **Идентификатор рынка OpenBook:**  
<B9coHrCxYv7xmPfSU7Z5VfugDqdTdZqZTpBGBdazq8AC>
- **Торговая пара:** BMV / SOL (с кросс-курсом к USD/EUR)
- **Solscan:** [Страница токена BMV Eco System](#)
- **Dexscreener:** [Основной график для отслеживания рыночного шума и общей цены.](#)
- **Birdeye:** [Ссылка на пару](#)
- **Jupiter (Aggregator):** [Ссылка](#)

## Перечень используемых данных:

Для функционирования бота используются следующие источники данных:

- **On-chain (Блокчейн):**
  - **Market ID (OpenBook):**  
<B9coHrCxYv7xmPfSU7Z5VfugDqdTdZqZTpBGBdazq8AC> – основная база для работы с ордербуком.

- Mint Address: 8VMi93EEqSGCHGjRGNhEFBkGoNzTQxUFFFYWDEX2e5UF .
- RPC URL: выделенный канал связи (Helius/QuickNode) .
- **Off-chain (Внешние):**
  - Кросс-курс SOL/USD – для расчета фиатного эквивалента и защиты ценового "поля".
- **История и Балансы (Локальная БД):**
  - Суммарный баланс токенов на всех подконтрольных кошельках.
  - Собственная история сделок бота и рынка (записывается в локальную базу данных) .

## Цели

- **Стабилизация цены:** поддержание цены токена BMV в восходящем канале и подавление резких просадок.
  - **Асимметрия ликвидности:** создание плотной BUY-зоны и разреженной SELL-зоны , позволяющих цене расти свободнее.
  - **Защита фиатной стоимости:** хранение части активов в USDC для демпфирования падений курса SOL.
  - **Скрытность и MEV-защита:** использование роя кошельков и Jito Bundles для предотвращения фронт-рингинга и защиты стратегии. нужен контроль размера чаевых, целесообразности.
  - **Аккумуляция эмиссии:** выкуп свободной эмиссии для контроля ликвидности.
- 

## 2. ТЕХНОЛОГИЧЕСКИЙ СТЕК

### 2.1 Core

- **Язык:** Rust (максимальная скорость) .

### 2.2 Инфраструктура

- **RPC:** выделенный узел (Helius/QuickNode) с поддержкой WebSocket.
- **Execution Layer:** Jito Block Engine (gRPC/HTTP) – обязательный шлюз для всех транзакций, включая рыночный шум и flash volume.
- **Хранилища:**
  - **Redis** (hot state) – текущий снэпшот ордербука и балансы;

- PostgreSQL (cold state) – история сделок, расчёт VWAP, хранение собственных записей для Pivot.
  - Логирование: `structlog` в JSON формате.
  - Метрики: `Prometheus`; `Grafana` для визуализации.
- 

## 3. МАТЕМАТИЧЕСКОЕ ЯДРО

### 3.1 Pivot Point – чистый VWAP (DEFAULT)

Для определения центра канала используется средневзвешенная цена по объёму (VWAP). Операционные издержки (комиссии, чаевые, `rent`) не входят в формулу и компенсируются спредом. Расчёт ведётся на основе накопленной истории сделок из локальной базы данных, куда периодически копируется история из блокчейна.

$$\text{Pivot} = \Sigma(\text{price} \times \text{volume}) / \Sigma \text{volume}$$

**Историческое окно** – скользящие 365 дней, по умолчанию, не аксиома. При запуске, когда история BMV мала, используется *Seeded Pivot*: текущая истинная средняя вычисляется и получает вес, эквивалентный 365 дням. Далее семенные данные постепенно вытесняются новыми.

### 3.2 Геометрия асимметричного канала

Канал состоит из двух зон относительно Pivot:

#### BUY Zone (Поддержка)

- Ширина по умолчанию – **15 % вниз от Pivot**. Параметр настраивается, а границы изменяются от инерционной Pivot, тогда как зона зависит от рыночной цены.
- Плотность ордеров высокая (экспоненциальная): чем ближе к нижней границе, тем крупнее объёмы. Решение: размер ордера и его число стабильное, а шаг переменный и это логичнее, меньше требования к размеру депозитов, проскальзываний.
- Масштабируется параметром `TARGET_CONTROL_%` – чем выше желаемый контроль эмиссии, тем плотнее BUY-тенденция.

## **SELL Zone (Рост)**

- Ширина по умолчанию – **30 % вверх от Pivot**. Параметр регулируется, а границы изменяются от инерционной Pivot, тогда как ширина зоны зависит от рыночной цены.
- Плотность низкая (разреженная), чтобы цена могла расти при минимальном сопротивлении.

Изначальные значения 30 %/15 % – настройки по умолчанию и могут быть изменены без изменения логики.

---

## **4. АРХИТЕКТУРА ИСПОЛНЕНИЯ**

### **4.1 Модульная структура BMV Bot**

<b>Модуль</b>	<b>Описание</b>
<b>Core Engine</b>	Центральный движок: расчёт Pivot, построение сетки, отправка ордеров.
<b>Swarm Orchestrator</b>	Координация мульти-кошельков, распределение ордеров по сегментам.
<b>Rebalancer</b>	Мониторинг отклонения цены, инициация перестройки сетки.
<b>Flash Volume Module</b>	Генерация атомарных wash-сделок для создания видимого объёма.
<b>Market Noise Module</b>	Чередующиеся рыночные ордера для поддержания активности (устаревший).
<b>Financial Manager</b>	Управление балансом SOL/USDC, конверсии через Jupiter DEX.
<b>Rent Recovery Service</b>	Возврат залоговой ренты с закрытых аккаунтов OpenBook.
<b>Key Manager</b>	Загрузка и безопасное хранение приватных ключей.
<b>Kill Switch</b>	Экстренная остановка всех операций.

## 4.2 Swarm Orchestrator

Для обхода ограничения OpenBook на количество открытых ордеров используется система мульти-кошельков:

- **Лимит ордеров:** один кошелёк может держать до 32 лимитных ордеров (BUY + SELL). Деление 16 BUY / 16 SELL является условным; фактическое распределение зависит от ширины BUY/SELL-зон и общего числа уровней.
- **Глобальная сетка:** количество уровней в канале (например, 100) распределяется между кошельками. Каждый кошелёк обслуживает свой сегмент из 32 ордеров.
- **Ротация:** Worker-кошельки действуют параллельно, создавая единый стакан. Балансы и истории синхронизируются через общий Redis и базу.

## 4.3 Rebalancer

Модуль Rebalancer отвечает за мониторинг рыночной цены и принятие решения о перестройке сетки.

### Взаимодействия Rebalancer

Rebalancer → Solana RPC : L2 orderbook scan (сканирование стакана)  
Rebalancer → Core Engine : rebuild trigger (сигнал на перестройку сетки)

Полный ребилд сетки выполняется, если:

- отклонение текущей цены от Pivot превышает REBALANCE\_THRESHOLD\_% (по умолчанию 1 %);
- или проходит принудительный интервал синхронизации (например, 1 час);
- или бот получает внешний сигнал (Kill Switch, изменение конфигурации).
- или при смене чужих блоков, перед которыми стояли собственные ордера."
- или при условии отсутствия снимаемых и выставляемых ордеров ближе чем 3% (по умолчанию) к рыночной цене."

Этот подход снижает нагрузку по сравнению с event-driven ребалансировкой. Для дополнительной защиты от снайперов бот сканирует L2-с т а к а н и размещает ордер на 1 тик ( $0.000001$  SOL) выгоднее крупного конкурирующего ордера.

#### 4.4 Flash Volume & Market Noise

**Flash Volume** (Atomic Wash Trading) используется для генерации видимого объёма при отсутствии сделок:

1. Бот выбирает два различных кошелька из пула.
2. Кошелёк A покупает заданный объём (по умолчанию эквивалент  $10\$$ ), кошелёк B одновременно продаёт тот же объём. Так как Solana волатильна, как и любая крипта, потому боту может быть отказано в размещении ордеров ниже этого порога, а если Solana вырастет, то шум будет из блоков, которые просверлят огромную дыру в бюджете.
3. Обе инструкции упаковываются в один Jito Bundle с минимальными чаевыми (например,  $0.001$  SOL) и исполняются атомарно.
4. На графике появляется пара сделок Buy/Sell, цена не меняется, а потери ограничены сетевой комиссией. Нужен контроль над целесообразностью чаевых, малые не дадут результата, большие не имеют смысла, может оказаться так, что рыночные окажутся выгоднее, или тормознуть вовсе, пока не снизится размер чаевых, который тоже волатилен.

Настройки:

- `FLASH_VOLUME_ENABLED` – включение режима.
- `FLASH_VOLUME_SIZE_SOL` – объём встречной сделки (default  $0.1$  SOL).
- `FLASH_VOLUME_INTERVAL_MIN` – минимальный интервал между атомарными сделками (по умолчанию 15 минут).

- `FLASH_VOLUME_TIP_SOL` – чаевые для валидатора (default 0.001 SOL).

Модуль Flash Volume является развитием идеи Market Noise. Чередующиеся рыночные ордера из версии 2.2.1 по-прежнему могут использоваться (режим `NOISE_MODE = alternated_market_cross`) – бот выставляет противоположные рыночные ордера объёмом `NOISE_ORDER_USD` через Jito Bundles, если в течение `NOISE_IDLE_INTERVAL_MIN` минут не было активностей. Отступ от рыночной цены не менее 3% по умолчанию.

---

## 5. ФИНАНСОВЫЙ МЕНЕДЖМЕНТ

### 5.1 Financial Manager

Модуль Financial Manager отвечает за управление балансом SOL/USDC и автоматические конверсии.

#### Взаимодействия Financial Manager

```

Financial Manager → Jupiter DEX : conversions (SOL ↔ USDC)

```

**Динамический буфер USDC/SOL.** Вместо жёсткой концепции «Fiat Floor» используется градиентное распределение активов с указанием минимального порога, иначе мелкими переливами, или попытками конвертации малой суммы будут потери на транзакциях или не исполнимые приказы (<10\$):

- В верхней части канала (SELL-зона) часть полученных от продаж SOL постепенно конвертируется в USDC. Соотношение SOL/USDC изменяется от **100/0 у Pivot** до **70/30 у верхней границы** по умолчанию.
- В нижней части (BUY-зона) накопленные USDC тратятся на покупку SOL, восстанавливая BUY-секцию и поддерживая цену токена. Соотношение BMV/USDC меняется аналогично от 100/0 до 70/30.
- Пороговые значения (70 %, 30 %) являются настройками и могут быть скорректированы.

- Цель: Использовать фиатный резерв для покупки подешевевшей Соланы, чтобы мгновенно усилить «стенки» (BUY-ордера) и откупить дно.

**Авто-инъекция:** если доля SOL в общем резерве падает ниже `MIN_SOL_RESERVE_%`, бот автоматически закупает SOL за USDC (через Jupiter) и распределяет их по торговым кошелькам.

## 5.2 Target Control

Целевой процент контроля эмиссии `TARGET_CONTROL_%` рассчитывается динамически:

$$\text{TARGET\_CONTROL\%} = 100\% - \text{LockedTokens} - \text{OwnedTokens}$$

Где `OwnedTokens` включает токены на кошельках бота и в выставленных лимитных ордерах. При увеличении объема токенов у сторонних держателей бот повышает плотность BUY-зона для выкупа эмиссии обратно и поддержания контроля. Для справки, эмиссия 10 000 000 токенов, заблокировано 50% на 4 года, с частичной разморозкой раз в 2 недели начиная с февраля 2026 года.

## 5.3 Rent Recovery Service

Закрытые аккаунты OpenBook освобождают залоговую ренту ( $\approx 0.023$  SOL). Периодический процесс (`RENT_RECOVERY_TIME`) сканирует закрытые ордера и вызывает инструкцию `CloseAccount` для возврата ренты.

### Взаимодействия Rent Recovery Service

Rent Recovery Service → Jito Block Engine : close accounts

...

## 6. КОНФИГУРАЦИЯ (Hot-Reload)

Параметры управляются из конфигурационного файла или переменных окружения и могут изменяться без перезапуска бота. Стандартные значения:

Вот отформатированная таблица конфигурационных параметров, основанная на предоставленном вами тексте:

Параметр	Default	Описание
HISTORY_WINDOW_DAYS	365	Длина окна для VWAP
BUY_CHANNEL_WIDTH_%	15	Ширина BUY-зоны
SELL_CHANNEL_WIDTH_%	30	Ширина SELL-зоны
REBALANCE_THRESHOLD_%	1	Порог отклонения цены для перестройки
ORDERS_PER_WALLET	32	Максимум лимитных ордеров на кошельк
TARGET_CONTROL_%	Dynamic	Целевой контроль эмиссии
UPPER_USDC_RATIO_MAX_%	30	Доля USDC у верхней границы канала
LOWER_USDC_RATIO_MAX_%	30	Доля USDC у нижней границы канала
MIN_SOL_RESERVE_%	70	Порог, при котором активируется USDC-->SOL инъекция
JITO_TIP_AMOUNT_SOL	0.005	Чаевые валидатору для лимитных транзакций
RENT_RECOVERY_TIME	1h	Интервал возврата ренты

<b>FLASH_VOLUME_ENABLED</b>	<code>true</code> ▾	Включение атомарных встречных сделок
<b>FLASH_VOLUME_SIZE_SCALE</b>	<code>0.1</code> ▾	Объём каждой стороны Flash Volume
<b>FLASH_VOLUME_INTERVAL_MIN</b>	<code>15</code> ▾	Интервал между Flash Volume
<b>FLASH_VOLUME_TIP_SIZE</b>	<code>0.001</code> ▾	Чаевые для валидатора в Flash Volume
<b>NOISE_MODE</b>	<code>alternated_market_crash</code> ▾	Режим рыночного шума (устаревший)
<b>NOISE_ORDER_USD</b>	<code>10</code> ▾	Размер рыночной сделки в режиме Noise
<b>NOISE_IDLE_INTERVAL_MIN</b>	<code>15</code> ▾	Пауза без активности для запуска Noise
<b>MIN_CONVERSION_BARRIER_USD</b>	<code>50</code> ▾	Порог конвертации (минимальная сумма перелива)
<b>NOISE_OFFSET_PERCENT</b>	<code>3</code> ▾	Отступ сменных лимитных ордеров от рыночной цены
<b>BLOCKCHAIN_INDEXING_INTERVAL_MIN</b>	<code>1</code> ▾	Периодичность индексации блокчейна

## 7. БЕЗОПАСНОСТЬ

- **Kill Switch:** глобальная переменная в Redis/Memory. При активации бот отменяет все ордера и завершает процессы.
  - **MEV Protection:** все write-транзакции отправляются в Jito Block Engine; публичный мемпул не используется; Atomic Bundles защищают от sandwich-атак.
  - **Exposure Limits:** жёсткое ограничение максимального размера ордера (в процентах от баланса кошелька), чтобы исключить ошибки округления и чрезмерный риск.
  - **Key Management:** приватные ключи загружаются из зашифрованного .env или внешнего Vault; ключи никогда не хранятся в коде.
- 

## 8. OBSERVABILITY & MARKET PRESENCE

Бот генерирует структурированные события и метрики для мониторинга и анализа стратегии:

### Обязательные события

```
GRID_REBUILD { pivot, trigger, wallet }
ORDER_FILL { price, volume }
FIAT_INJECTION { usdc, sol }
RENT_RECLAIM { sol }
FLASH_VOLUME { side, volume, tip }
```

Для устаревшего режима Noise также используется:

```
NOISE_TRADE { side, notional_usd, reason }
```

Эти события служат источником для внешнего мониторинга (Prometheus/Grafana) и публикации на витринах (например, DEXScreener).

### Дополнительные метрики

- **Synthetic Volume Rate** – доля искусственного объема (Flash Volume / Noise) относительно общего оборота.
  - **Trade Cadence** – среднее время между сделками.
  - **Market Noise Ratio** – отношение шумовых сделок к реальным.
  - **Wallet Exposure** – распределение активов и рисков по кошелькам.
- 

## 9. ПОВЕДЕНЧЕСКИЕ СЦЕНАРИИ

- **Рост SOL:** при росте цены SOL Pivot и SELL-зона смещаются вверх, BUY-зона подтягивается вслед; отношение SOL/USDC движется в сторону SOL.
- **Падение SOL:** при падении курса активируется демпфер USDC; бот конвертирует USDC в SOL и усиливает BUY-стену; Target Control увеличивает плотность заявок.
- **Флэт:** если рынок стабилен, бот может периодически создавать микро- сделки (Flash Volume или Noise) для поддержания видимости активности; зоны BUY/SELL остаются статичными до превышения порога перерасчета. Необходима оценка целесообразности и рисков применения рыночных ордеров, рыночных встречных, чередующихся ордеров через сервис Jito, или встречных.

## 10. ДИАГРАММА ВЗАИМОДЕЙСТВИЙ

Ниже приведена сводная таблица всех ключевых взаимодействий между модулями системы:

Источник	Назначение	Тип взаимодействия
Core Engine ▾	Price Oracle ▾	fetch price (получени... ▾
Core Engine ▾	Prometheus ▾	expose metrics ▾
Core Engine ▾	Storage (Redis/PG) ▾	read/write state ▾

Core Engine ▾	Jito Block Engine ▾	trade execution ▾
Swarm Orchestrator ▾	Storage (Redis/PG) ▾	coordination, sync ▾
Rebalancer ▾	Solana RPC ▾	L2 orderbook scan ▾
Rebalancer ▾	Core Engine ▾	rebuild trigger ▾
Flash Volume Module ▾	Jito Block Engine ▾	atomic wash trades ▾
Market Noise Module ▾	Jito Block Engine ▾	noise orders ▾
Financial Manager ▾	Jupiter DEX ▾	conversions (SOL ↔ ... ▾
Rent Recovery Service ▾	Jito Block Engine ▾	close accounts ▾
Kill Switch ▾	Redis ▾	stop operations ▾
Jito Block Engine ▾	OpenBook DEX ▾	trade execution ▾
Jito Block Engine ▾	Jupiter DEX ▾	trade execution ▾

---

**Статус документа:** в процессе обсуждения