Руководство администратора ПО

«Сервис лимитов на транзакции»

1. ВВЕДЕНИЕ

1.1. Область применения

Настоящий документ предназначен для администраторов, обеспечивающих установку, настройку и эксплуатацию программного обеспечения «Сервис лимитов на транзакции» (далее — Сервис).

Документ описывает порядок развертывания Сервиса в контейнеризованной среде, требования к инфраструктуре, параметры конфигурации, порядок администрирования, мониторинга и взаимодействия с API.

1.2. Перечень выполняемых функций администратора/оператора

Администратор Сервиса обеспечивает полный жизненный цикл эксплуатации системы. В его обязанности входит развертывание Сервиса в контейнерной среде (Docker, Kubernetes или иной оркестратор), настройка параметров подключения к базе данных PostgreSQL, конфигурация переменных окружения и параметров безопасности, включая ключ доступа к API.

Администратор контролирует работоспособность Сервиса, выполняет мониторинг состояния приложения и базы данных, анализирует журналы работы, организует сбор и хранение метрик, а также обеспечивает резервное копирование и восстановление данных.

В задачи администратора входит управление конфигурацией лимитов через административные методы API, контроль корректности их работы, а также устранение инцидентов, связанных с превышением лимитов, ошибками конфигурации или отказами инфраструктуры.

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен уметь пользоваться и настраивать среду функционирования контейнеров или систему оркестрации, используемую на предприятии.

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

Администраторы Системы должны пройти обязательную общую и специальную подготовку для работы с Системой.

Общая подготовка должна включать в себя получение знаний и навыков работы с Системой в качестве администратора.

Специальная подготовка должна включать в себя получение знаний и навыков в объеме, необходимом для выполнения своих должностных обязанностей.

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- Функциональная спецификация системы
- Техническое задание на разработку системы
- Настоящее руководство администратора

2. УСТАНОВКА СИСТЕМЫ

2.1. Системные требования

2.1.1. Аппаратные требования

Сервис предназначен для запуска в контейнеризованной среде. Ниже приведены минимальные рекомендуемые требования для одного экземпляра приложения (одного pod/контейнера API):

- CPU: не менее 2 логических ядер;
- Оперативная память: не менее 2 ГБ (рекомендуется 4 ГБ при высокой нагрузке);
- Дисковое пространство: от 10 ГБ (без учета резервных копий).

Фактические требования зависят от интенсивности входящего потока операций и объема запросов к базе данных.

При горизонтальном масштабировании требования применяются к каждому экземпляру отдельно.

2.1.2. Программные требования

Сервис разворачивается в контейнерной среде и требует:

- Kubernetes, Docker Swarm или другой оркестратор;
- PostgreSQL версии 16+;
- Prometheus (опционально, для мониторинга);
- Grafana (опционально, для визуализации метрик).

Операционная система хоста должна поддерживать запуск контейнеров (Linux-дистрибутив серверного класса).

2.1.3. Язык программирования

Сервис реализован на языке Go (Golang).

Для динамического извлечения ключей и значений лимитов используется встроенный JavaScript-интерпретатор (для исполнения пользовательских скриптов).

2.2. Порядок установки

Развертывание осуществляется в контейнерной среде.

Общий порядок действий:

- Подготовить инфраструктуру контейнеризации (Docker или Kubernetes).
- Подготовить экземпляр PostgreSQL.
- Создать базу данных для Сервиса.
- Настроить переменные окружения контейнера (см. раздел 3.2).
- Развернуть контейнер API.
- Проверить доступность публичного метода ping.
- Подключить мониторинг (при необходимости).

3. НАСТРОЙКА СИСТЕМЫ

3.1. Общие сведения

В данном документе приводятся примеры настройки Системы с использованием среды Docker Compose. Настройка операционной системы, СУБД, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

3.2. Конфигурируемые параметры

Конфигурация осуществляется через переменные окружения контейнера.

Сетевые параметры

- HOST — адрес и порт API (по умолчанию :80)
- METRICS_PORT — порт для служебных эндпоинтов и health-check (по умолчанию :3000)
- Параметры безопасности
- ACCESS_KEY — ключ доступа к административным методам API (обязательный параметр)
  Передается клиентом в заголовке:
  Authorization: Bearer <token>

Параметры логирования и метрик

- RPC_LOGGING_ENABLED — включение логирования JSON-RPC вызовов (по умолчанию false)
- RPC_METRICS_ENABLED — включение метрик JSON-RPC (по умолчанию false)

Параметры подключения к PostgreSQL

- DB_HOST — адрес сервера БД (по умолчанию db)
- DB_PORT — порт БД (по умолчанию 5432)
- DB_USER — пользователь БД (обязательно)
- DB_PASSWORD — пароль пользователя БД (обязательно)
- DB_NAME — имя базы данных (по умолчанию ratelimiter_db)

Параметры пула соединений

- DB_MAX_OPEN_CONNECTIONS — максимальное количество открытых соединений (по умолчанию 30)
- DB_MAX_IDLE_CONNECTIONS — максимальное количество неактивных соединений (по умолчанию 30)
- DB_CONN_MAX_TIME — максимальное время жизни соединения (по умолчанию 3m)

3.3. Безопасность

Ключ ACCESS_KEY должен храниться в защищенном хранилище секретов.

Рекомендуется ограничить доступ к административному API на уровне сети.

Рекомендуется использовать TLS-терминацию на уровне ingress/reverse-proxy.

3.4. Мониторинг

Сервис предоставляет эндпоинт:

GET /metrics

Метрики формируются в формате Prometheus.

При включенном RPC_METRICS_ENABLED доступны:

- jsonrpc_requests_total{method,status}
- jsonrpc_request_duration_seconds_sum{method}
- jsonrpc_request_duration_seconds_count{method}

Рекомендуется настроить алерты на:

- рост количества ошибок;
- увеличение времени ответа;
- недоступность сервиса.

3.5. Логирование и аудит

Логи приложения выводятся в stdout в формате JSON.

При включенном RPC_LOGGING_ENABLED логируются:

- метод;
- статус выполнения;
- длительность обработки.

В таблице operation_log сохраняются:

- параметры вызова;
- решение (allowed/denied);
- причина отказа;
- значения лимитов.

3.6. Структура базы данных

Основные таблицы:

- limits — конфигурация лимитов;
- limit_counters — счетчики потребления;
- operation_log — аудит операций.

Рекомендуется:

- регулярно выполнять резервное копирование;
- контролировать рост таблицы operation_log;

4. ОПИСАНИЕ API

4.1. Общие сведения

Система предоставляет JSON-RPC API версии 2.0 по HTTP.

Все приватные методы доступны по адресу /

Публичный метод ping доступен по адресу /public

Доступ к приватным методам защищен с помощью ключа доступа, передаваемого в заголовке Authorization в формате Bearer token

Заголовок Content-Type должен быть application/json

4.2. Методы API

4.2.1. ping

Проверка доступности сервиса.

Запрос:

```
{
   "jsonrpc": "2.0",
   "method": "ping",
   "id": 1
}
```

Ответ:

```
{
   "jsonrpc": "2.0",
   "result": "pong",
   "id": 1
}
```

4.2.2. limits.check

Метод предназначен для проверки возможности выполнения операции без изменения счётчиков лимитов.

Метод анализирует применимые активные лимиты, рассчитывает текущее потребление и определяет, приведёт ли выполнение операции к превышению порога.

Формат запроса

```
{
  "jsonrpc": "2.0",
  "method": "limits.check",
  "id": 2,
  "params": {
    "user_id": "user_001",
    "operation_type": "bet",
    "attributes": {
      "amount": 500,
      "currency": "RUB",
      "game_id": "slot-1"
    }
  }
}
```

Параметры метода

- user_id (string, обязательный) - Идентификатор пользователя, для которого выполняется проверка.
- operation_type (string, обязательный) - Тип операции (например: bet, deposit, withdrawal). Используется при выборе лимитов, настроенных на конкретный тип операции.
- attributes (object, обязательный) - Объект произвольных атрибутов операции.

Может содержать:

- amount (number) - сумма операции. Используется при metric_type = amount.
- sum (number) - альтернативное имя поля суммы.
- game_id (string) - идентификатор игры (если лимит учитывает игру).
- иные параметры - используются пользовательскими JavaScript-скриптами (script_extract_key, script_extract_value).

Формат успешного ответа (операция разрешена)


```
{
 "jsonrpc": "2.0",
 "result": {
  "allowed": true
 },
 "id": 2
}
```


allowed (boolean) - Признак разрешения операции.

Формат ответа при превышении лимита


```
{
 "jsonrpc": "2.0",
 "result": {
  "allowed": false,
  "reason": "Limit exceeded",
  "limit_id": 1,
  "current": 9800,
  "threshold": 10000,
  "description": "Максимальная сумма ставок пользователя за день"
 },
 "id": 2
}
```

Поля результата:

- allowed (boolean) - всегда false.
- reason (string) - Техническое описание причины отказа.
- limit_id (integer) - Идентификатор лимита, который был превышен.
- current (number) - Текущее значение потребления лимита.
- threshold (number) - Пороговое значение лимита.
- description (string) - Описание лимита.

4.2.3. limits.commit

Метод фиксирует успешно выполненную операцию и увеличивает счётчики потребления.

Формат запроса

```
{
 "jsonrpc": "2.0",
 "method": "limits.commit",
 "id": 3,
 "params": {
  "user_id": "user_001",
  "operation_type": "bet",
  "attributes": {
   "amount": 500,
   "currency": "RUB",
   "game_id": "slot-1"
  }
 }
}
```

Параметры метода

Параметры полностью аналогичны методу limits.check:

- user_id (string, обязательный)
- operation_type (string, обязательный)
- attributes (object, обязательный)

Формат ответа

```
{
 "jsonrpc": "2.0",
 "result": true,
 "id": 3
}
```

result (boolean) - true — операция успешно зафиксирована.

При ошибке возвращается стандартный JSON-RPC error-объект.

4.2.4. limits.create

Метод предназначен для создания нового лимита. Требует авторизации.

Формат запроса

```
{
 "jsonrpc": "2.0",
 "method": "limits.create",
 "id": 4,
 "params": {
  "name": "Дневной лимит ставок",
  "description": "Максимальная сумма ставок пользователя за день",
  "period_type": "daily",
  "metric_type": "amount",
  "threshold_value": 10000,
  "scope_type": "user",
  "priority": 10,
  "level": "global",
  "is_active": true
 }
}
```

Параметры метода

- name (string, обязательный) - Название лимита.
- description (string, необязательный) - Текстовое описание лимита.
- period_type (string, обязательный) - Тип периода действия лимита:
    - once,
    - daily,
    - weekly,
    - monthly,
    - custom.
- period_seconds (integer, обязательный при custom) - Длительность периода в секундах (скользящее окно).
- metric_type (string, обязательный) - Тип метрики:
    - amount — суммирование значения операции;
    - count — подсчёт количества операций.

- threshold_value (number, обязательный) - Пороговое значение лимита.
- scope_type (string, обязательный) - Область применения лимита:
    - user,
    - operation_type,
    - game,
    - user_operation,
    - user_game,
    - operation_game,
    - user_operation_game.
- scope_user_id (string, необязательный) — идентификатор пользователя, для которого применяется данный лимит. Если параметр указан, лимит распространяется только на операции этого пользователя.
- scope_operation_type (string, необязательный) — тип операции, к которому применяется лимит. Если параметр указан, лимит учитывает только операции указанного типа.
- scope_game_id (string, необязательный) — идентификатор игры, к которой применяется лимит. Если параметр указан, лимит распространяется только на операции, связанные с указанной игрой.
- priority (integer, необязательный) - Приоритет проверки (большее значение — выше приоритет).
- level (string, необязательный) - global, user, temporary.
- script_extract_key (string, необязательный) - JavaScript-код для формирования ключа лимита.
- script_extract_value (string, необязательный) - JavaScript-код для извлечения значения операции.
- is_active (boolean, необязательный) - Признак активности лимита.

Формат ответа

Возвращается созданный объект лимита:


```
{
  "jsonrpc": "2.0",
  "result": {
    "id": 1,
    "name": "...",
    "description": "...",
    "period_type": "daily",
    "metric_type": "amount",
    "threshold_value": 10000,
    "scope_type": "user",
```

```
    "priority": 10,
    "level": "global",
    "is_active": true,
    "created_at": "2026-02-23T10:00:00Z",
    "updated_at": "2026-02-23T10:00:00Z"
  },
  "id": 4
}
```


4.2.5. limits.get

Возвращает лимит по идентификатору.

Формат запроса


```
{
  "jsonrpc": "2.0",
  "method": "limits.get",
  "id": 5,
  "params": {
    "id": 1
  }
}
```


Параметры

- id (integer, обязательный) - Идентификатор лимита.

Формат ответа

Возвращается объект лимита в полном виде (аналогично limits.create).

4.2.6. limits.update

Обновляет существующий лимит.

Формат запроса


```
{
  "jsonrpc": "2.0",
  "method": "limits.update",
  "id": 6,
  "params": {
    "id": 1,
    "threshold_value": 15000,
    "is_active": true
  }
```

```
}
```

Параметры

- id (integer, обязательный) - Идентификатор лимита.
- Остальные параметры — аналогичны limits.create.

Передаются только поля, подлежащие изменению.

Формат ответа

Возвращается обновлённый объект лимита.

4.2.7. limits.delete

Удаляет лимит.

Формат запроса

```
{
 "jsonrpc": "2.0",
 "method": "limits.delete",
 "id": 7,
 "params": {
  "id": 1
 }
}
```

Параметры

- id (integer, обязательный) - Идентификатор лимита.

Формат ответа

```
{
 "jsonrpc": "2.0",
 "result": true,
 "id": 7
}
```

4.2.8. limits.list

Возвращает список лимитов с фильтрацией и постраничной навигацией.

Формат запроса

```
{
 "jsonrpc": "2.0",
 "method": "limits.list",
 "id": 8,
 "params": {
  "level": "global",
  "is_active": true,
  "limit": 20,
  "offset": 0
 }
}
```

Параметры

- level (string, необязательный) - Фильтр по уровню: global, user, temporary.
- is_active (boolean, необязательный) - Фильтр по активности.
- limit (integer, необязательный) - Количество записей (по умолчанию 100).
- offset (integer, необязательный) - Смещение выборки (по умолчанию 0).

Формат ответа

```
{
 "jsonrpc": "2.0",
 "result": [
  {
   "id": 1,
   "name": "...",
   "period_type": "daily",
   "threshold_value": 10000,
   "is_active": true
  }
 ],
 "id": 8
}
```

Результат представляет собой массив объектов лимитов, отсортированных по: priority по убыванию, id по возрастанию