Общее руководство программного обеспечения «Sparrow Ads»

О программном обеспечении

Программное обеспечение "Sparrow Ads" - это платформа размещения ставок на аукционы в системах programmatic-рекламы. ПО обрабатывает запросы на аукционы в связке SSP (Supply-Side Platform) и DSP (Demand-Side Platform), распределяет рекламные предложения по ставкам, отслеживает состояние аукционов и рекламных кампаний.
ПО «Sparrow Ads» позволяет обрабатывать OpenRTB запросы в рамках программатик рекламы. На запросы ПО «Sparrow Ads» формирует аукцион размещая ставки информационных и рекламных объявлений (далее «креативы») по предложениям со стороны запроса в формате NativeAd. Состояние аукционов отслеживается по событиям формируя записи в OLAP базе данных для анализа эффективности рекламных кампаний, предложений и ставок.

ПО «Sparrow Ads» поставляется в виде .zip архива, содержащего исходный код ПО, примеры настроек dockerfile и docker compose, для развертывания на серверах пользователя (клиента/заказчика).

ПО «Sparrow Ads» взаимодействует с внешними сервисами/источниками данных:

- Источник OpenRTB запросов (SSP платформа, далее также именуется “сеть”)
- Источник креативов и рекламных кампаний (SVK платформа, каталог тизеров, баннеров и т.д.)
- Рекомендательным сервисом (SVK платформа, общие рекомендации объявлений к ресурсам)
- Аналитическими сервисами для ставок и коррекции (далее именуется “модуль”)

ПО «Sparrow Ads» выступает в роли брокера между креативами, предложениями о размещения креативов (imp), рекомендательным сервисом и модулями корректировки.

Главным процессом ПО является формирование ответов аукциона в формате OpenRTB 2.5 и NativeAd 1.1 на запрос от источников SSP.

Задачи выполняемые ПО

ПО «Sparrow Ads» выполняет следующие задачи:
- Обработка запросов и формирование ответов по протоколу OpenRTB
- Поддержка формата NativeAd
- Фильтрация входящих запросов
- Подбор ставок по рекламным кампаниям и фильтрам
- Настройка для соответствия требованиям со стороны SSP
- Обработка событий состояния аукционов
- Сбор и обработка статистики

- Интеграция с платформами рекламодателей.

А также осуществляет:
1. Проверку входящих OpenRTB запросов по адресу: *.dsp.sparrow.ru/request/openrtb
2. Корректировку запроса по настройкам сети.
3. Поиск рекомендаций по схожим запросам.
4. Применение фильтров рекламных кампаний в отношении характеристик запроса и предложений (imp).
5. Поиск подходящих цен ставок по параметрам запроса и найденых/отфильтрованных выше рекомендаций.
6. Применение корректирующих коэффициентов.
7. Выборка креатива и сопутствующей ставки исходя из веса предложения.
8. Генерация ставки (bid) и разметки в формате NativeAd для ответа в формате OpenRTB.
9. Возвращение ответа.
10. Запись полученного аукциона в базу данных (Redis), предложений и ставок в базу данных (ClickHouse). Передача данных в цикличную коллекцию в базу данных (MongoDB) для отладки.

Также ПО «Sparrow Ads» принимает события для изменения статусов ставок.

Внутренний механизм допускает Out-of-Order на всех этапах, т.е. сам аукцион и события могут быть получены в любом порядке. Повторные запросы игнорируются. Очередь ожидания и распределения маркирует состояния прежде чем аукцион будет считаться завершенным.

Поступающие ставки из аукционов хранятся временно, изначально с минимальным сроком жизни. События notice и bill продлевают срок жизни ставки аукциона в ожидании возможного click события.

Ставка аукциона считается завершенной, если по ней были получены либо click событие, либо lose событие, или срок жизни ставки был исчерпан.

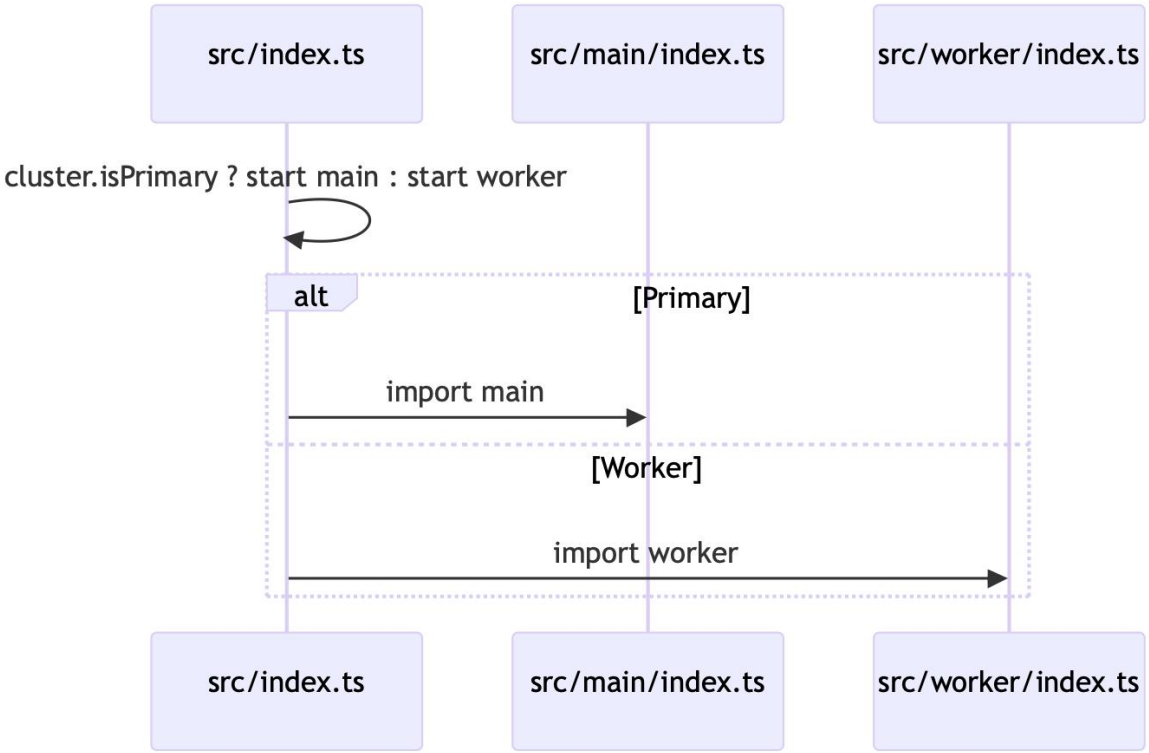
! Аукцион считается завершенным, если все ставки в нем были завершены.

Приоритетом обработки является скорость и время на формирование ответа аукциона — многие SSP платформы имеют строгие лимиты и ожидают ответ в узкий промежуток времени, например менее 80 миллисекунд. В ПО «Sparrow Ads» это может быть скорректировано настройками буферов кэш данных, позволяя выбирать оптимальный вариант между сроком жизни и скоростью ответа для каждой конкретной сети.

Каждый процесс обработки запросов содержит локальную копию буферов для рекомендаций, объявлений, рекламных кампаний и настроек ставок. Синхронизация данных выполняется через систему оповещения сообщениями pub/sub механизма в базе данных (Redis).

Процессы обработки запросов объединяются в узлы используя механизм кластеризации в среде NodeJS. В свою очередь узлы могут также группироваться в рамках одной сети для повышения пропускной способности и отказоустойчивости.

Описание процесса запуска ПО



Основной (main) процесс (src/main/index.ts):

- Инициализация журнала.
- Запуск кластера (node:cluster).
- IPC маршрутизация сообщений с worker процессами.
- Запуск синхронизации с платформой SVK для получения рекламных кампаний, фильтров и объявлений.
- Запуск синхронизации между инстансами ПО посредством Redis (pub/sub).
- Запуск синхронизации настроек для SSP из MongoDB.
- Запуск HTTP сервера (Fastify) на {MASTER_PORT} для API ПО «Sparrow Ads».
- Ретрансляция данных синхронизации от Redis к worker процессам.

Worker процесс (src/worker/index.ts):

- Создаются автоматически основным процессом.
  - В случае аварийного завершения процессы перезапускаются (перезапуск осуществляется только когда основной процесс переходит в режим готовности).
- IPC маршрутизация сообщений с родительским процессом.
- HTTP сервер (Fastify) на {WORKER_PORT} (общий для всех worker процессов одного инстанса ПО «Sparrow Ads») для открытого API и взаимодействия с платформой SSP партнера.

Синхронизация данных из внешних систем:

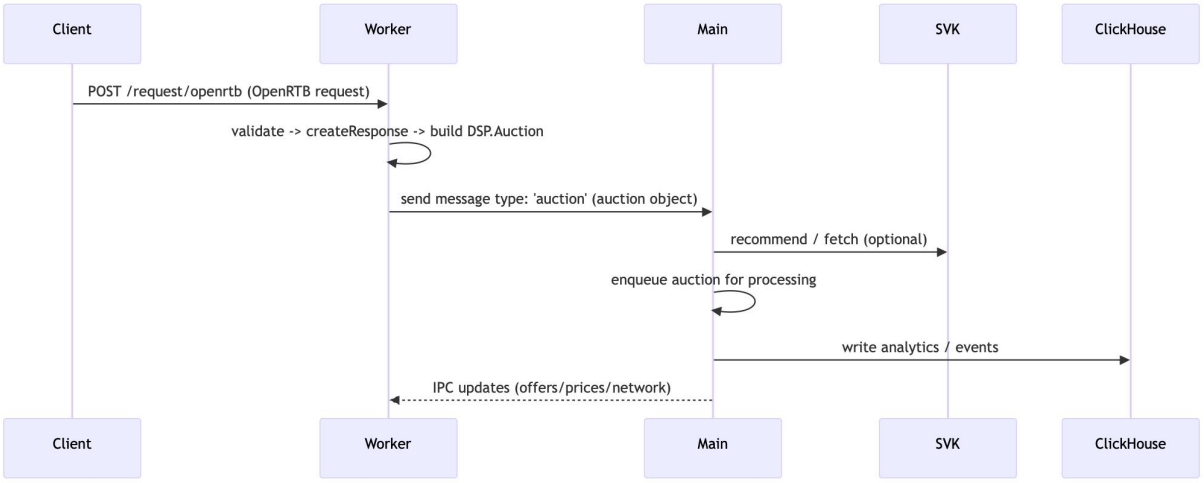
- Платформа SVK:
  - Периодически вызываются `fetchTeasers`, `fetchCampaigns`, `fetchTeaserStats`.
  - Данные рассылаются по workers для обновления локальных кэшей.

- Redis pub/sub: `src/main/sync.ts` подписывается на канал `network:${networkId}:bidder` и ретранслирует сообщения в main/воркерам. Ожидаемые виды сообщений: `network`, `offers`, `prices`.
- Mongo network fetch: `src/main/network.ts` при `NETWORK_SYNC=mongo` регулярно загружает конфигурацию сети из коллекции `networks` в Mongo и рассылает её воркерам; также выполняет локальную валидацию.

Обработка событий и сохранение аналитики:

- Main получает от воркеров `auction` объекты и:
  - ставит их в очередь для дополнительной обработки/репликации (`main/auctions.ts`).
  - отправляет запросы к SVK recommendation API (`main/svk/offers.ts`) при необходимости.
  - события кликов/наблюдения отправляются в `main/events.ts` и далее могут записываться в ClickHouse (см. `src/service/clickhouse.ts`) и/или Redis time-series.

Описание процесса обработки OpenRTB запроса



SSP платформа вызывает POST /request/openrtb с корректным key и в теле запроса содержатся OpenRTB запрос с imp (один и более).

1) Входные проверки:
- Проверка валюты: если `request.cur` присутствует, сравнивается с `network.currency`.
- Подготовка `request.site.domain` (функция `getDomain` в `src/util/openrtb.ts`).

2) Выбор bidder:
- `pickBidder` выбирает одного `bidder` из `network.bidders` по весам и флагу `enabled`.

3) Формирование общего объекта ответа:
- Инициализируется `response` с `id` (из `request`), `bidid` (ULID), `cur`.
- `bids` — массив мета-информации о ставках (для последующей аналитики).

4) Обработка каждого `imp` (предложения о ставки):
- Если `imp.bidfloor > network.maxBid` — пропускается.
- При `useImpIdAsTag` имп `imp.id` используется как `tagid`.
- Для каждого `imp` вызывается `createBids`.

5) Создание bids (ставок) для imp:
- Сбор тизеров на которые данный запрос и предложение могут распространяться: `matchTeasers(request, imp)`.
- В зависимости от модели взаимодействия с SSP (cpm или cpc) расчитывается цена ставки каждого тизера.
- Фильтрация по `imp.bidfloor` и `teaser.bidPrice`.
- Применение модификаторов.
- Применение рекомендаций весов.
- Выбор тизера случайно по весам: `getRandomByWeight`.

6) Формирование поля `bid` (ставка):
- Генерация `id` (ULID), `adomain` (hostname из teaser.url для SSP требующих домен конечного лэндинга клика), `impid`, `adid`, `price`.
- Генерация URL для `nurl` и `burl` с подписями.
- Генерация `clickURL` через и вставка в `adm` (markup) через `createTeaserResponse` для native.
- При включённом `network.enableORD` добавляется `ext.nroa` с полями ОРД.

7) Мета-данные:
- Мета данные по ставкам размещаются в объекте аукциона (эти данные внутренние и SSP не передаются).

8) Возврат ответа:
- `createResponse` собирает seatbid(ы) и возвращает объект аукциона воркеру.
- Worker процесс отправляет объект аукциона с мета информацией в main и отвечает клиенту: если есть bids внутри ответа — OpenRTB response, иначе 204.

Процесс подключения сети

1. Сеть регистрируется в базе данных DSP (MongoDB) в виде конфигурационного файла в формате JSON.
2. Создаются один или несколько узлов обработки запросов в зоне .dsp.sparrow.ru, например rtbads.dsp.sparrow.ru.
3. Создается один или несколько модулей для обработки статистики и генерации стэка цен ставок.
4. Партнеру (SSP платформа, supply сторона) передается URL для принятия запросов содержащий уникальный ключ доступа к API.
5. ПО «Sparrow Ads» синхронизирует актуальные данные по рекламным объявлениям, рекламным кампаниям и фильтрам.
6. Параметры настройки согласовываются с партнером (интерпретация событий, сопоставление статистики и т.д.).

Учет аукционов может вестись по моделям CPM, CPMv и CPC.
Настройки сети позволяют учитывать как именно сеть реализует списание для синхронизации статистики между SSP и DSP.

API

API ПО «Sparrow Ads» делится на две части - открытая и закрытая. Открытая часть предоставляет точку входа для источника OpenRTB запроса и диагностическую статистику по конкретному worker процессу. Закрытая часть предоставляет точки входа для управления ценами ставок, статистику текущих аукционов и другими функциональными возможностями рассмотренными ниже.

Открытая часть (worker)

Не требует аутентификации.
Выполняется на индивидуальных worker процессах.

Общая информация по worker процессу:

GET /

Отображает общее состояние worker процесса, например uptime, имя текущей конфигурации, кол-во поступивших запросов, кол-во тех или иных ошибок для диагностики при интеграции с SSP платформой.

Обработка запроса на аукцион:

POST /request/openrtb

Параметры (query) запроса:

- key - API ключ доступа для сети (передается партнеру SSP платформы).

Основная точка входа для аукционов — принимает OpenRTB запрос и возвращает OpenRTB ответ или статус 204.
Тело запроса проверяется по схеме в src/validate/openrtb/request.ts.
В случае успеха и наличия ставок по запросу ответ будет в формате OpenRTB Response.
В иных случаях возвращается статус 204.

Регистрация клика по креативу и ставке аукциона:

GET /click

Обработка перехода по клику. Валидирует подпись запроса и выполняет redirect на `CLICK_REDIRECT_URL`. Возвращает статус 302 и регистрирует клик в системе.

Регистрация событий по ставке аукциона:

GET /event

Обработка событий показа креатива и биллинга показа в зависимости от настроек сети. Валидирует подпись. Передает событие в основной процесс для обработки состояния аукциона.

Параметры (query) запроса:

- type — Тип события
- adId — Идентфикатор креатива
- auctionId — Идентфикатор аукциона
- bidId — Идентфикатор ставки
- sign — Подпись параметров URL
- price — Цена (опциональный)
- trackerId — Идентификатор для отслеживания групп ставок (опциональный)

Дополнительно для отладки доступны следующие точки API:

GET /debug/test — возвращает заголовки запроса.
GET /debug/network — вывод текущей network-конфигурации.
GET /debug/campaigns — список текущих кампаний.
GET /debug/prices/last — последние ~200 записей цен.
GET /debug/prices/dump — бинарный дамп рекомендаций ставок.
GET /debug/offers/matches — число совпадений по offers.
GET /debug/offers/dump — бинарный дамп предложений/рекомендаций.
GET /debug/teasers/dump — бинарный дамп идентификаторов тизеров.
GET /debug/teasers/stats — статистика по тизерам.

Важно! Как и для закрытой части ниже доступ к этим точкам ограничен Bearer-аутентификацией по ключу из переменной окружения MAIN_KEY.

Закрытая часть (main)

Точки доступа располагаются на main процессе и требуют ключа доступа (определяется в переменных окружения под MAIN_KEY) для Bearer-аутентификации.

Установка рекомендуемых ставок по параметрам:

POST /prices

В тело запроса приходит JSON, состоящий из массива объектов включающие свойства:

- bidderId (string) — Идентификатор bidder из настроек сети
- adId (number) — Идентификатор креатива
- price (number) — Рекомендуемая цена ставки*
- weight (number) — Вес рекомендации (опционально)
- match (object) — Параметры совпадения (опционально)
- trackerId (string) — Идентификатор группы (опционально)

* — отрицательное значение отменяет все предыдущие установки по этому креативу.

Объект match может содержать один или несколько свойств:

- geoCountryExact (string) — По стране (гео) клиента (request.device.geo.country)
- deviceTypeExact (number) — По типу устройства (см. OpenRTB device)
- siteIdExact (number) - По идентификатору сайта
- siteDomainExact (string) — По домену сайта
- sitePageExact (string) — По URL страницы
- impTagExact (string) — По request.imp.tagid (см. OpenRTB Imp)

```
[
  {
    "bidderId": "test",
    "adId": 1,
    "price": 4,
    "weight": 10,
    "trackerId": "exp01",
    "match": {
      "geoCountryExact": "RU",
      "siteDomainExact": "any-example.ru"
    }
  },
  …
]
```

Для нескольких инстансов ПО «Sparrow Ads» (объединенных в одну группу посредством общего сервера Redis) не имеет значение какому был отправлен запрос - записи будут продублированы для всех.

Информация о текущих параметрах:

GET /prices

Отображает 1000 последних записей переданных ранее.

Статистика по ставкам, показам и кликам:

GET /bids/search

Query параметры:

- start (string) — Начало периода в формате ISO8601
- end (string) — Конец периода в формате ISO8601
- interval (string) — Интервал статистики, допустимые значения: minute, day, month

Следующие параметры опциональны и могут быть использованы для уточнения статистики по конкретным позициям и комбинациям позиций:

- bidderId (string) — Идентификатор bidder из настроек сети
- adId (integer) — Идентификатор креатива
- domain (string) — Доменное имя сайта
- page (string) — URL страницы
- country (string) — Страна из request.device.geo
- device (number) — Тип устройства
- widgetId (number) — Идентификатор виджета по которому был выполнен показ и/или клик

Возвращает статистику за выбранный период по указанным параметрам в формате:


```
{
  "entries": [
    {
      "timestamp": "2026-02-16T21:00:00Z",
      "bids": 2008510,
      "wins": 276923,
      "clicks": 3162,
      "expenses": 2059.15,
      "gains": 5189.37
    },
    ...
  ],
  "total": {
    "bids": 66272827,
    "wins": 3892345,
    "clicks": 43969,
    "expenses": 35505.47,
    "gains": 72570.22999999998
  }
}
```

Требования к системе

Для установки ПО «Sparrow Ads» необходимо:

- Любой современный дистрибутив операционной системы на базе Linux.
- Установлены nodejs версии 23 и выше включая пакетный менеджер npm (по-умолчанию входит в пакет с nodejs).
- Оперативная память объемом от 4 гб.
  - Минимум по 400 мегабайт на каждый worker процесс и столько же сверху для основного процесса (например если доступно 8 ядер то 400 * 8 + 400, итого 3,6 гигабайта).
- Доступ к сервисам (базам данных) хранения данных: redis, clickhouse и mongo.
- Доступ к платформе SVK.

ПО «Sparrow Ads» поставляется в виде .zip архива, содержащего исходный код ПО, примеры настроек dockerfile и docker compose, для развертывания на серверах пользователя (клиента/заказчика).

Настройка окружения

Основные переменные окружения для запуска (поместить например в .env файл или в docker-compose.yml для развертывания):

```
NETWORK_ID=1
MAIN_KEY=changeme
EVENT_URL=http://localhost:8080
REDIS_URL=redis://localhost:6379
CLICKHOUSE_URL=http://localhost:8123/dsp
MONGO_URL=mongodb://localhost:27017/dsp
SVK_URL=https://api.svk-native.ru
SVK_KEY=your_key_here
CLICK_REDIRECT_URL=https://widgets/dsp
EVENT_URL=http://dsp-1.dsp.com
EVENT_SIGN_KEY=changeme
WORKER_COUNT=8
WORKER_PORT=8080
MASTER_PORT=8081
LOG_LEVEL=info
```

Форматы URL для соединения с Redis, ClickHouse и Mongo можно найти в их документации. Как правило они имеют формат:
protocol://username:password@hostname:port/database

Рекомендуется использовать Docker контейнеры для развертывания.
В исходном коде содержатся примеры для развертывания посредством Docker compose а также Dockerfile для ПО «Sparrow Ads».

Переменные окружения ПО

Имя	Пример	Описание
NETWORK_ID	1	Идентификатор SSP в api SVK
REDIS_URL	http://redis:8123/dsp	URL доступа к Redis
CLICKHOUSE_URL	redis://clickhouse:6379	URL доступа к ClickHouse
MONGODB_URL	mongodb://mongo:27017/dsp	URL доступа к MongoDB
SVK_URL	https://api.svk-native.ru	URL для SVK API
SVK_KEY	test	Ключ доступа к SVK API
WORKER_COUNT	8	Кол-во worker процессов
WORKER_PORT	8080	TCP порт доступа к worker процессам
MASTER_PORT	8081	TCP порт доступа к API ПО
MAIN_KEY	test	Bearer ключ для доступа к API
EVENT_URL	http://dsp-1.dsp	Адрес на который будут приходить события по аукциону
EVENT_SIGN_KEY	test	Ключ подписи URL событий
OFFER_BUFFER	4000000	Размер буфера кэша предложений
PRICE_BUFFER	4000000	Размер буфера кэша цен
CLICK_REDIRECT_URL	https://widget.promo	Адрес на который будет происходит редирект клиента по клику

Для CLICK_REDIRECT_URL предопределены следующие переменные, которые передаются при редиректе:

- n — Идентификатор SSP (соответствует NETWORK_ID)
- t — Идентификатор объявления (adid)
- s — Идентификатор сайта (из рекомендаций)
- w — Идентификатор виджета (из рекомендаций)