Разбор нераспределённых расходов/доходов и настройка правил автоклассификации операций

Универсальный регламент для пользователей сервиса «Money Sellers».

Период разбора: (например 2026-06-01…2026-06-30 или июнь 2026).

Все id статей резолвятся в рантайме по имени — ничего не хардкодить. Документ состоит из двух частей: **A** — разовый разбор и исправление платежей; **B** — настройка постоянных правил, чтобы впредь раскладывалось само.

Доступ

- Финплан резолвится из api-key — `finPlanId` не передавать.
- Канал: MCP `https://mcp.api.report.finance/mcp` (Bearer) и/или REST `https://rest.api.report.finance/api/...` (`X-API-KEY`).
- Массовые выгрузки — лучше REST `/api/Payments`. При `401 ipIsNotInWhitelist` добавить исходящий IP (из текста 401 или `curl https://api.ipify.org`) в whitelist; либо работать через MCP `payment.list` (whitelist не проверяет).
- Пагинация: `offset`, лимит 100, ориентир по `totalLineCount`.
- Перед записью — всегда резервная выгрузка и тест на одном объекте.

Часть A. Разбор и исправление нераспределённых платежей

Шаг 0 — резолв id нераспределённых статей по имени

id системных статей привязаны к финплану и меняются при пересоздании. Источник истины — имя, а не число.

1. Эталонные имена — из `report.schema.get {reportType:"dds"}`: листья «Нераспределённый приход» (доход) и «Нераспределённый расход» (расход). id узла схемы — это id строки отчёта, НЕ `factStreamId`.
2. Имя → `factStreamId`: основной путь — REST `/api/FactStreams?reportTypeId=115` по имени; fallback (REST закрыт) — кандидаты = `factStreamId` на платежах, которых нет в `factStream.list`, доход/расход делить по `directionId` (500=приход, 510=расход), брать доминирующий по объёму.
3. Cross-check: направление совпадает с именем (приход↔500, расход↔510). Зафиксировать, какие id использованы и как получены.

Шаг 1 — выгрузка платежей за период

1. Выгрузить все платежи за {ПЕРИОД} постранично.
2. Отфильтровать по `factStreamId` нераспределённых статей (фильтра по статье в API нет — фильтровать на клиенте; поле статьи в платеже — `factStreamId`).
3. Зафиксировать количество и сумму отдельно по приходу и расходу.
4. Даты: фильтр `dateFrom/dateTo` по operationDate; `paymentDate` может расходиться с `internalPaymentDate`. При сомнениях — проверять весь период и сверять каждую дату через API, не доверяя экспорту из интерфейса.

Шаг 2 — обогащение

Для каждого платежа подтянуть: контрагент (`contragentId`→имя), счёт, проект, организацию, назначение (`paymentPurpose`), комментарий (`comment`), `responsible`, теги, `operationTypeId`/`directionId`, `externalId`. Справочники — `contragent.list`, `account.list`, `project.list`, `organisation.list`.

Шаг 2.5 — справочник решений по истории (приоритетный источник статьи)

Не угадывать статью эвристиками сразу — сперва учиться на уже размеченных платежах самого клиента.

1. Выгрузить платежи за ≥2 предыдущих месяца до периода (если совпадений мало — увеличить глубину), у которых стоит нормальная статья (не нераспределённая).
2. Построить индексы «признак → статья» с частотой: по `contragentId`/имени контрагента (учитывать разное написание); по нормализованному назначению (`paymentPurpose` без сумм/номеров/дат); по нормализованному комментарию (`comment`).
3. Для каждого нераспределённого платежа искать решение по приоритету: точный контрагент → совпадение назначения → совпадение комментария. Однозначное историческое сопоставление → брать статью (в «Почему»: «по истории: X в N платежах относился на Y»). Конфликт (один признак → разные статьи) → на решение клиента.
4. Только если истории нет — переходить к эвристикам (Шаг 3). Этот же индекс — основа для правил Части B.

Шаг 3 — мэппинг и Excel на согласование

Таблица: **Старая статья | Новая статья | Почему** (+ для проверки: id, дата, сумма, контрагент, счёт, назначение, комментарий, уверенность, источник решения).

Приоритет определения статьи: (1) история (Шаг 2.5) → (2) эвристики по назначению/контрагенту → (3) спорное на решение клиента.

- «Почему» — логика: по истории / назначению / контрагенту / счёту / проекту / комментарию-источнику / типу операции.
- Правило «нет данных — не трогаем»: если пусто И назначение, И комментарий — оставить на нераспределённой статье как есть (в обновление не включать).
- Спорные группы не угадывать, а выносить на решение клиента (личные траты, переводы физлицам, технические остатки).
- Статьи не менять до согласования. Сначала показать Excel.

Шаг 4 — исправление (после подтверждения)

1. Резервная JSON-выгрузка всех затрагиваемых платежей до изменений (полный объект каждого).
2. Тест на одном платеже: обновить, заново прочитать через API, убедиться, что изменилась только статья, прочие поля целы.
3. Обновление: слать полный объект платежа с заменённым `factStreamId` через `payment.update` (по `id`) — чтобы не затереть остальные поля.
4. `payment.update` требует непустой `externalId`. Если пустой — подставить в payload технический `codex-update-<id>` (сервер обновит по `id`). Альтернатива для платежей с реальным externalId — upsert через `payment.create`.
5. Контрольная проверка через API: сколько осталось на нераспределённых; сверить остаток с набором «оставленных по правилу пустоты».

Шаг 5 — отчёт по части A

Найдено / исправлено / осталось; что не удалось и почему; ссылки на Excel и JSON (бэкап, лог обновлений).

Часть B. Правила автоклассификации

Цель — превратить подтверждённый мэппинг в постоянные правила: новые импортируемые платежи сразу получают статью, минуя нераспределённую. **Правила применяются только при импорте/создании платежей и задним числом ничего не пересчитывают.**

Шаг B1 — кандидаты в правила (только стабильные сигналы)

- Контрагент → одна статья (бизнес-контрагент, не «Имя Ф.»), все его платежи легли в одну статью.
- Ключевое слово в назначении → одна статья (единообразно по всем совпадениям).

НЕ создавать правила для личных трат по карте, переводов физлицам/СБП, технических остатков и для сигналов, лежащих только в комментарии (надёжно адресуемы лишь поля «назначение» и «контрагент»). Если контрагент платит по разным статьям — правило по нему не делать.

Шаг B2 — ОБЯЗАТЕЛЬНО: проверка существующих правил против дублей

Перед созданием прочитать `rule.list {reportType:115, ruleType:1}` и сопоставить каждое предлагаемое условие с существующими:

1. Дубль = совпадают поле + значение (нормализация регистра/пробелов) + целевая статья → НЕ создавать.
2. Поле+значение совпадает, но статья другая → это конфликт: не создавать молча, вынести на решение клиента.
3. Если по теме правило уже есть → дополнять его (тот же `id` в `rule.create` = обновление, добавив новые `listRuleConditionModel`), а не плодить второе.
4. Учитывать варианты написания контрагента (напр. «ИП Борисенко А. А» ≠ «ИП БОРИСЕНКО АНДРЕЙ АНДРЕЕВИЧ») — ловить «содержит» по короткому устойчивому фрагменту.
5. Зафиксировать в отчёте: какие кандидаты пропущены как дубли и какие конфликты найдены.

Шаг B3 — показать список правил на согласование

Таблица: Имя правила | Условие | → Статья (id) | Статус (новое / дополнение существующего id / пропущено-дубль / конфликт). Создавать только после «ок».

Шаг B4 — создание правил

Механика `rule.create` (одно правило):

```
{ "reportType":115, "id":0,            // 0=создать, >0=обновить
  "name":"<тема> (авто)", "ruleType":1,  // 1=factStream (статья)
  "listRuleConditionModel":[
    { "valueTrue":<factStreamId>,          // целевая статья
      "listConditionModel":[
        { "columnTypeId":565, "operatorId":271, "valueIf":"<значение>" } ] }
    // несколько элементов = OR; условия внутри listConditionModel = AND
  ] }
```

Словарь (подтверждать по примерам из `rule.list` своего финплана):

- `columnTypeId`: 565 = назначение платежа, 566 = контрагент.
- `operatorId`: 271 = содержит/regex, 256 = равно.
- `valueTrue` = `factStreamId` целевой статьи (резолвить по имени из `factStream.list`).

Рекомендация: группировать по теме/статье — одно правило на статью со всеми триггерами; в имя добавлять маркер запуска для опознаваемости и удаления (`rule.delete` по id).

Шаг B5 — верификация и отчёт по части B

Перечитать `rule.list`, подтвердить, что правила появились с присвоенными id и корректными условиями. В отчёте: созданные правила (id, условия, статья), дополненные существующие, пропущенные дубли, конфликты на решение.

Технические напоминания

- Имя нераспределённой статьи — из `report.schema.get`; целевые статьи — из `factStream.list` (reportType=dds, 115).
- Никаких захардкоженных `factStreamId` — всегда резолв по имени.
- `payment.list` фильтрует только по id/externalId/contragentId/projectId/organisationId/accountId/requestId — не по статье.
- `payment.update` требует непустой `externalId` (иначе `codex-update-<id>`).
- Правила: перед `rule.create` всегда `rule.list` и дедуп. Правила не действуют задним числом.
- Не полагаться на экспорт из интерфейса — сверяться с API.