# Pert

**Техническая документация**

*Paranoid Security*

# Table of contents

# 1. Главная


*WaaS-платформа для безопасных операций с цифровыми активами*

## О платформе

**Pert** — некастодиальная WaaS-платформа для финтеха и бизнеса. Она объединяет учёт цифровых активов, выпуск кошельков, обработку депозитов и выводов, а также управление политиками и аппрувами в едином защищённом контуре.

Платформа построена на базе **MPC (Multi-Party Computation)** — криптографического протокола, при котором приватный ключ никогда не существует целиком ни на одной машине. Это даёт **некастодиальность**: операции с активами невозможны без явного согласия владельца, даже со стороны оператора платформы.

**Некастодиальное хранение**

Ключи разделены между независимыми участниками по протоколу MPC. Ни один участник в одиночку не может подписать транзакцию.

**Ролевая модель и аппрувы**

Гибкое распределение прав внутри команды, многоступенчатые аппрувы исходящих транзакций и подтверждение операций с мобильного устройства.

**Политики операций**

Настраиваемые правила для исходящих транзакций и AML-скрининга: разрешить автоматически, потребовать аппрув или заблокировать.

**События в реальном времени**

Webhooks с подписью на стороне Pert и WebSocket для получения уведомлений о статусах транзакций, депозитах и других событиях.

**Аудит и мониторинг**

Полная история операций, событий безопасности и действий пользователей — для расследований и соответствия требованиям.

**Соответствие требованиям**

AML-скрининг адресов и пост-скрининг входящих транзакций, разделение ролей и прозрачный журнал решений.

## С чего начать

### Начало работы

Получите API-ключ, создайте vault-аккаунт и отправьте первую транзакцию за несколько шагов.

### Pert API

Полный справочник REST API: эндпоинты, схемы запросов и ответов, примеры использования.

### Webhooks

Подписка на события, верификация подписи, ретраи и лучшие практики интеграции push-уведомлений.

### Политики

Transfer, AML Screening и Post-Screening: как работают правила, приоритеты и действия.

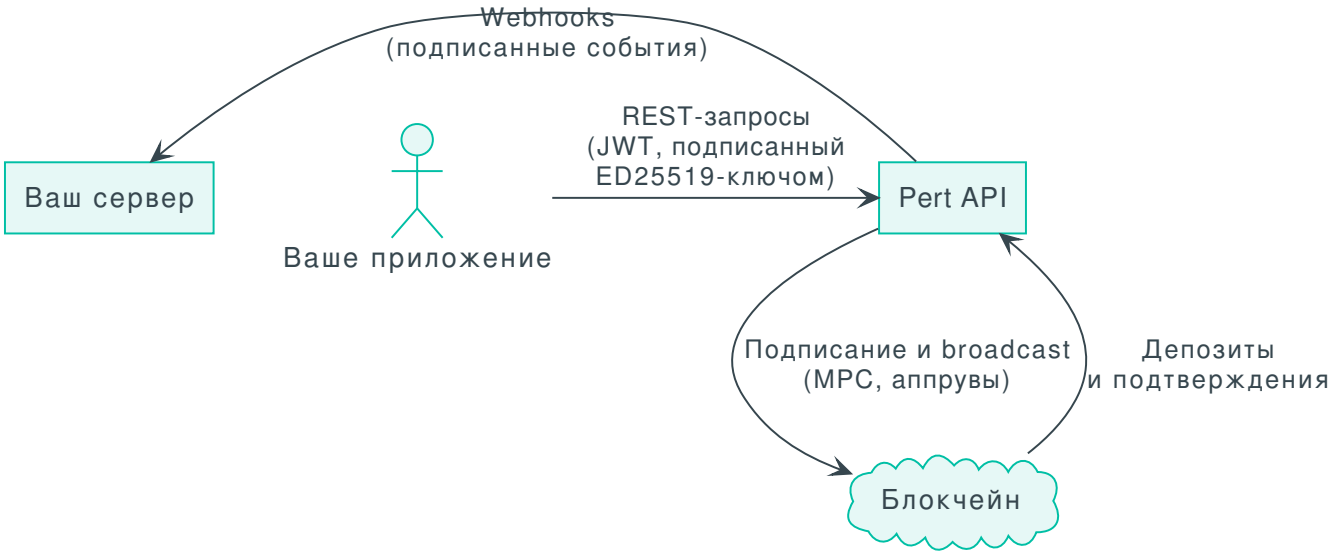
### Авторизация

Асимметричные ключи, JWT-токен, ротация и хранение секретов на стороне клиента.

### Коды ошибок

Единый формат ошибок API и справочник машиночитаемых кодов для обработки в клиенте.

## Как устроена интеграция




1. **Ваше приложение** обращается к Pert API по REST, аутентифицируясь асимметричным ключом, который никогда не покидает ваше окружение.
2. **Pert** обрабатывает запрос, при необходимости запускает политики и аппрувы, после чего подписывает транзакцию по протоколу MPC и отправляет её в блокчейн.
3. **Webhooks** доставляют события (создание транзакции, смена статуса, депозит, изменения настроек) на ваш сервер — с криптографической подписью, чтобы вы могли убедиться в подлинности уведомления.

**Готовы начать?**

Перейдите к руководству по началу работы — оно проведёт вас через создание API-ключа и первую транзакцию.

# 2. База знаний

## 2.1 Глоссарий

Справочник терминов, которые встречаются в документации, REST API и интерфейсах Pert. Термины сгруппированы по доменам; внутри домена идут от общих понятий к частным.

> **Как пользоваться**
>
> Если встретили в API, в ответе или в Customer Console незнакомое слово — ищите его здесь по `Ctrl+F`. В описаниях есть ссылки на подробные разделы документации.

### Платформа и доступ

**Pert**

Имя продукта. WaaS-платформа для безопасного хранения и операций с цифровыми активами на базе протокола **MPC-CMP**. Приватный ключ никогда не существует целиком ни на одной машине — операции невозможны без явного согласия владельца ключа.

**Компания**

Юридическое лицо — клиент Pert. Может владеть несколькими workspace (например, разные подразделения, продукты, тестовое и боевое окружения).

**Workspace (рабочее пространство)**

Изолированная единица учёта внутри Pert: набор vault, кошельков, пользователей, политик, AML-настроек и устройств-подписантов. Ключи разных workspace друг с другом не пересекаются; роли, политики и балансы — изолированы. Подробнее — Структура кошельков.

**Workspace Owner (владелец)**

Пользователь с ролью `OWNER` — инициатор создания workspace. Имеет полный набор прав внутри своего workspace, включая создание бэкапа ключей и добавление новых подписантов.

**Пользователь workspace**

Учётная запись внутри workspace с одной из ролей: `OWNER`, `ADMIN`, `NON_SIGNING_ADMIN`, `SIGNER`, `APPROVER`, `EDITOR`, `VIEWER`. Роли определяют, что пользователь может видеть и делать: подписывать транзакции, аппрувить, управлять политиками, просматривать историю.

**Сервисный аккаунт (service account)**

Машинный пользователь для интеграции по API. Не имеет мобильного устройства, аутентифицируется через API-ключ. Используется вашим бэкендом для автоматизированной работы с Pert API.

**API-ключ (API key)**

Пара ключей `ed25519`, привязанная к сервисному аккаунту. Приватный ключ хранится у вас и подписывает JWT для авторизации; публичный — загружается в Pert. См. Авторизация и Начало работы.

**Устройство (device)**

Носитель одной из MPC-долей пользователя: либо мобильное приложение **Pert Mobile** (для пользователей с правом подписи), либо **Co-Signer** — headless-инстанс, разворачиваемый клиентом на своей стороне для API-сценариев. Привязывается к учётной записи через процесс pairing.

**Pairing (привязка устройства)**

Процесс ассоциации нового устройства с пользователем workspace. Устройство генерирует ключевую пару, регистрирует публичный ключ в Pert, после чего владелец workspace выдаёт ему MPC-долю.

---

## Хранение активов

**Vault (vault-аккаунт)**

Логическая группа кошельков внутри workspace, объединённая общим набором MPC-ключей и общим назначением (например, «казначейство», «горячий кошелёк», «выплаты»). Большинство политик и многих ролей выдаются именно на vault. Подробнее — Структура кошельков.

**Wallet (кошелёк)**

Контейнер активов одного блокчейна внутри vault: баланс, адрес для приёма средств и история транзакций по конкретной сети/ассету. Бывает двух типов: **внутренний** (custodial, ключами владеет MPC-схема Pert) и **внешний whitelisted** (зарегистрированный адрес контрагента). См. Структура кошельков.

**Адрес (address)**

Конкретный on-chain адрес внутри кошелька. На каждый внутренний кошелёк приходится **ровно один** адрес — он создаётся вместе с кошельком и далее переиспользуется для всех поступлений, в том числе для UTXO-сетей вроде BTC. Адрес выводится из ключей vault по детерминированному HD-пути (BIP-32).

**Whitelisted wallet / адрес**

Внешний кошелёк или адрес контрагента, заранее добавленный в whitelist workspace. На whitelisted-адреса можно настроить упрощённый аппрув или автоматический пропуск через политику; выводы на любые другие адреса (см. one-time) обычно требуют более строгого подтверждения.

**One-time address**

Свободно введённый адрес назначения, не входящий в whitelist. На оценке политик такой адрес учитывается отдельно и обычно требует более строгого аппрува.

**Asset**

Описание токена/монеты: тикер, decimals, блокчейн, контракт (для ERC-20/TRC-20 и т. п.). Один wallet может содержать несколько asset (нативная монета + токены этой же сети).

**Smart-contract (custom asset)**

Пользовательский токен/контракт, заведённый администратором workspace. После регистрации становится обычным asset, к которому применяются те же политики и лимиты.

**Balance**

Доступный и заблокированный баланс по активу. При создании исходящей транзакции система проверяет, что доступного баланса достаточно с учётом комиссии (если она списывается тем же активом).

**Газ-станция (gas station)**

Назначенный vault, который автоматически снабжает базовым активом (газом) остальные vault'ы того же workspace — чтобы переводы токенов не застревали из-за нехватки газа на оплату комиссии. Одна на workspace; применима только к сетям с моделью газа (EVM). Подробнее — Газ-станция и автозаправка.

**Автозаправка (autofuel)**

Признак на отдельном vault: разрешено ли газ-станции автоматически доливать ему газ. Включается и выключается на каждый vault отдельно; пока не включена, заправок нет. См. Как работает автозаправка.

**Порог / Лимит баланса / Комиссия за газ (газ-станция)**

Параметры заправки, задаваемые на каждый базовый актив (в скобках — поле API): **порог** ( `threshold` ) — граница, ниже которой запускается долив; **лимит баланса** ( `cap` ) — целевой баланс, до которого доливают газ; **комиссия за газ, макс** ( `max_gas_price` ) — лимит комиссии за одну заправку. Текущие значения по умолчанию.

## Криптография и подписание

**MPC (Multi-Party Computation)**

Криптографический протокол вычислений над секретом, разделённым между несколькими сторонами. Никто из участников не имеет полного ключа — подпись формируется в результате взаимодействия минимум `t` из `n` сторон (threshold). Это основа некастодиальности Pert.

**MPC-CMP**

Конкретная схема ECDSA MPC (Canetti-Makriyannis-Peled, 2020) с параллельным pre-signing и быстрым sign-online. Используется во всех узлах MPC-схемы Pert.

**Доля ключа (key share)**

Часть приватного ключа, хранимая одной MPC-стороной. По отдельной доле невозможно ни восстановить ключ, ни подделать подпись. Доли распределяются при создании vault и пересчитываются при добавлении новых подписантов (без изменения публичного ключа).

**Keygen / Sign / Refresh**

Три основные процедуры MPC-CMP:

- **Keygen** — совместная генерация нового ключа (доли распределяются при создании vault).
- **Sign** — совместное подписание транзакции.
- **Refresh** — пересборка долей без изменения публичного ключа (используется при добавлении нового подписанта).

**Threshold (порог)**

`t-of-n` — минимальное число долей, нужное для подписи. В типовой конфигурации Pert схема — 2-of-3: **две доли** на стороне Pert (в защищённой серверной среде) и **одна доля** на стороне клиента — либо на headless-устройстве **Co-Signer**, либо в мобильном приложении **Pert Mobile** у пользователя-подписанта.

**Подписант (signer)**

Пользователь workspace с правом подписи и привязанным устройством. Технически — владелец одной MPC-доли. На стадии подписания получает запрос «подписать транзакцию» на своё устройство и подтверждает операцию вручную (Pert Mobile) или автоматически по политикам (Co-Signer).

**Sighash**

Хэш payload транзакции, который непосредственно подписывается. Для EVM/TRON — один хэш на транзакцию; для BTC — по одному на каждый UTXO-вход.

**Derivation (HD path)**

BIP-32-путь, по которому из master-доли получается доля для конкретного адреса. Позволяет MPC-схеме формировать подписи для нужного дочернего ключа, не собирая мастер-ключ.

**Backup ключей**

Экспорт зашифрованных MPC-долей в файл, который владелец может хранить у себя. Шифруется на публичный ключ из desktop-приложения **Pert Backup** и используется для восстановления при потере устройств-подписантов.

## Транзакции

**Transaction (транзакция)**

Учётная запись об исходящем выводе средств (**withdrawal**) или входящем зачислении (**deposit**). Хранит источник, назначение, актив, сумму, статус и историю переходов.

**Withdrawal**

Исходящая транзакция: создаётся вашим приложением через `POST /transaction`, проходит проверку политик и AML, после аппрувов подписывается по MPC и отправляется в блокчейн.

**Deposit**

Входящая on-chain транзакция, обнаруженная Pert на одном из адресов workspace. Создаётся сразу с финальным статусом — никаких аппрувов и подписей со стороны клиента не требуется.

**Статусы транзакции**

Жизненный цикл исходящей транзакции описывается набором статусов (`pending`, `awaiting_approvals`, `approved`, `awaiting_broadcast`, `broadcast`, `finished`, `declined`, `frozen` и т. п.) — схема и описание этапов в Жизненном цикле транзакции. Текущий статус и история переходов доступны через REST API и приходят событиями в webhooks.

**Approval (аппрув)**

Подтверждение исходящей транзакции одним из назначенных аппруверов. Подписывается устройством аппрувера. Сколько и каких аппрувов нужно — определяет политика workspace.

**Approval set / quorum**

Набор аппруверов, из которых нужно собрать `t` подписей для пропуска транзакции. Политика может требовать нескольких независимых наборов (например, «2-of-3 финансистов **и** 1-of-2 руководителей»).

**Freeze / Unfreeze**

**Freeze** — приостановка транзакции (исходящей) или vault (для входящих) вручную оператором или автоматически AML-правилом. Заморозить можно только транзакцию до этапа подписания и broadcast. **Unfreeze** возвращает предыдущий статус.

**Broadcast**

Отправка подписанной транзакции в блокчейн. После broadcast транзакция считается отправленной; финализация наступает после подтверждения сетью (см. Reconciliation).

**Reconciliation**

Сверка статуса транзакции в блокчейне с локальным состоянием Pert. По факту достаточного числа сетевых подтверждений транзакция финализируется ( `finished` ) либо отклоняется ( `declined` ).

**Idempotency-Key**

Заголовок `X-Idempotency-Key` на `POST /transaction` : уникальный ключ, сохраняемый в Pert. Повторный запрос с тем же ключом вернёт уже существующую транзакцию (HTTP 409) — это позволяет безопасно ретраить запросы.

## Политики и AML

**Policy (политика)**

Правило обработки транзакций, заданное на уровне workspace. Описывает условия (источник, назначение, сумма, частота, актив) и действие ( `allow` / `require approval` / `block` ). Подробнее — Политики.

**Transfer policy**

Подкласс политик, оценивающих исходящие транзакции. Может ссылаться на vault, whitelisted-кошельки и one-time адреса — например, «выводы на whitelisted из vault `payouts` до 10 000 USD/сутки проходят без аппрува, всё остальное — с аппрувом 2-of-3». См. Transfer-политики.

**AML rule (pre-screening / post-screening)**

Правило AML-провайдера, делящееся на:

- **Pre-screening** — локальная проверка адреса назначения до вызова провайдера, позволяет сэкономить запросы. См. AML pre-screening.
- **Post-screening** — обработка результата провайдера, в том числе автоматическая заморозка входящего депозита с высоким риск-скором. См. AML post-screening.

**Limit / Timeframe**

Накопительные ограничения политик: сумма или число операций за заданное окно времени (например, «не более 50 000 USD исходящих за 24 часа»). Лимиты освобождаются обратно, если транзакция была отклонена, заморожена или истёкла.

**Approval requirement**

Запись о том, какие аппруверы и в каком количестве должны подписать конкретную транзакцию. Формируется по политикам workspace в момент оценки транзакции.

## Интеграция

**Webhook**

HTTP-callback на ваш URL: Pert отправляет события о создании транзакции, смене статуса, депозите и других изменениях. Доставляется с повторными попытками и подписан на стороне Pert, чтобы вы могли убедиться в подлинности уведомления. Можно повторно проиграть пропущенные события через replay. См. Webhooks.

**Подпись webhook**

Криптографическая подпись тела события, передаваемая в HTTP-заголовке. Позволяет вашему приложению удостовериться, что событие пришло от Pert и не было изменено. См. Подпись webhook.

**Replay (повторное проигрывание)**

Механизм повторной отправки уже доставленных webhook-событий за выбранный интервал. Используется при сбое на стороне получателя или при первичной выгрузке истории. См. Replay.

## Приложения Pert

**Pert Console**

Web-интерфейс клиента: управление workspace, vault, кошельками, политиками, аппрувами, whitelist'ом, пользователями и сервисными аккаунтами.

**Pert Mobile**

Мобильное приложение пользователя-подписанта. Хранит долю MPC-ключа, подтверждает подписание и аппрувы вручную владельцем устройства — последняя линия обороны некастодиальной модели.

**Co-Signer**

Headless-устройство клиента, разворачиваемое на собственной инфраструктуре. В отличие от Pert Mobile действует **автоматически** на основании политик — используется для API-интеграций и высокочастотных операций без участия человека на каждой подписи.

**Pert Backup**

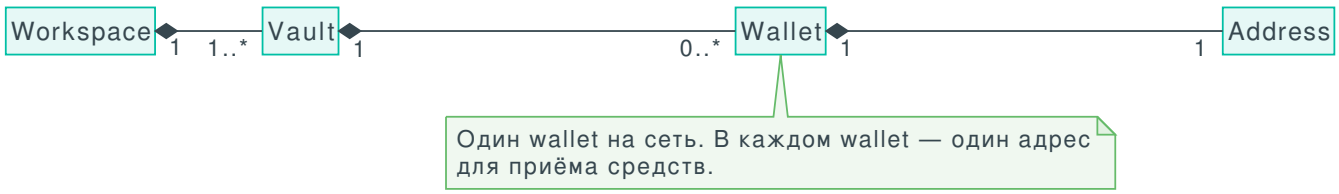
Desktop-приложение для безопасного создания и восстановления бэкапа MPC-долей. Шифрует доли публичным ключом, который никогда не покидает рабочую станцию владельца.

## 2.2 Структура кошельков

Все цифровые активы в Pert хранятся в строгой иерархии из четырёх уровней: **Workspace → Vault → Wallet → Address**. Каждый уровень вкладывается в предыдущий и наследует от него границы изоляции, ключи и часть правил доступа.

Эта страница — справочник по этой иерархии: что представляет собой каждый уровень, сколько вложений он может содержать, и как уровни связаны между собой.

### Иерархия одним взглядом




| Уровень | Сколько вложений |
|---|---|
| **Workspace** | `1..*` vault |
| **Vault** | `0..*` wallet |
| **Wallet** | `1` address |

### Workspace

**Изолированная единица учёта** внутри платформы: набор vault'ов, кошельков, пользователей, политик, AML-настроек и устройств-подписантов.

Workspace задаёт **границы безопасности и видимости**. Всё, что описано ниже, существует строго в рамках одного workspace:

- MPC-ключи разных workspace друг с другом не пересекаются;
- балансы, транзакции и истории — изолированы;
- роли и права выдаются строго в рамках одного workspace;
- политики (transfer, AML) настраиваются на уровне workspace.

Один клиент (юридическое лицо) может владеть несколькими workspace'ами — например, под разные подразделения, продукты или тестовое/боевое окружения.

### Vault

**Логическая группа кошельков** внутри workspace, объединённая общим набором MPC-ключей и общим назначением — например, «горячий кошелёк», «казначейство», «выплаты пользователям».

> **Vault — основная единица доступа**
>
> Большинство политик и многих ролей выдаются «на vault»: можно дать одному пользователю право подписи в хранилище казначейства, а другому — только в горячем кошельке.

Свойства vault:

- **Статус** — `active`, `frozen` или `deleted`. Заморозка vault блокирует исходящие операции по всем кошелькам внутри него.
- **Derivation path** — общий префикс HD-дерева ключей. Все кошельки и адреса внутри vault выводятся как дочерние ключи от этого пути, поэтому один набор MPC-ключей покрывает все блокчейны vault.
- **Auto-fuel** — флаг автоматического пополнения газа на дочерних адресах в EVM-сетях (если включён, система сама подбрасывает нативную монету для оплаты комиссии).

В каждом workspace всегда есть хотя бы один vault — он создаётся при первичной настройке.

## Wallet (кошелёк)

**Контейнер активов одного блокчейна** внутри vault. Хранит баланс, набор адресов и историю транзакций по конкретной сети/ассету.

Кошельки бывают двух типов:

**Внутренние (custodial)** **Внешние (whitelisted)**

Кошельки, ключами от которых владеет MPC-схема Pert (доли распределены между нодами и устройствами подписантов клиента).

При создании такого кошелька система сразу выдаёт **один** адрес на приём средств; этот адрес — публичный реквизит кошелька, его и нужно отдавать плательщикам. Дополнительные адреса в одном кошельке не создаются: это упрощает учёт балансов и наблюдение за поступлениями.

Используются для приёма депозитов и отправки исходящих транзакций. Доступный баланс и заморозки учитываются именно на уровне внутреннего кошелька.

Зарегистрированные адреса контрагентов, на которые разрешены выводы средств. Управляются через раздел *Whitelisted Wallets* в Customer Console.

Внешний кошелёк содержит ровно один адрес — сам whitelisted-адрес. Балансы у такого кошелька не отслеживаются: он нужен исключительно как «известное и проверенное» назначение для исходящих переводов.

Политики позволяют разрешать выводы *только на whitelisted-кошельки* и требовать дополнительный аппрув для остальных (one-time addresses).

Один vault обычно содержит несколько внутренних кошельков — по одному на каждую поддерживаемую сеть/ассет.

## Address

**Конкретный on-chain адрес** внутри кошелька, на который можно принимать средства. На каждый кошелёк приходится один такой адрес — он создаётся вместе с кошельком и далее переиспользуется для всех поступлений (в том числе для UTXO-сетей вроде BTC).

Адрес выводится из ключей vault по детерминированному HD-пути (BIP-32). По этому пути MPC-схема в момент подписи восстанавливает дочерний ключ для нужного адреса, не имея ни мастер-ключа, ни возможности его собрать.

## Как это работает вместе

**Пример: типичная конфигурация**

Один workspace «Acme Trading» содержит:

- **Vault** `treasury` (status: `active`)
  - Wallet `BTC` — один BTC-адрес на приём средств
  - Wallet `ETH` — один EVM-адрес
  - Wallet `USDT-TRC20` — один TRON-адрес
- **Vault** `payouts` (status: `active`, `auto_fuel: true`)
  - Wallet `ETH` — один EVM-адрес для выплат
- **Whitelisted кошельки**: 27 заранее одобренных адресов контрагентов, на которые разрешены выводы из vault `payouts` без ручного аппрува.

Политики настроены так, что выводы из `treasury` всегда требуют кворум 2-of-3 подписантов, а выводы из `payouts` на любой whitelisted-адрес проходят автоматически до лимита 10 000 USD в сутки.

## Что дальше

- Политики — как настроить правила обработки транзакций для конкретных vault и кошельков.
- Webhooks — как получать уведомления о новых адресах, депозитах и изменениях статусов транзакций по интересующим вас кошелькам.
- API — справочник REST-методов для управления vault, кошельками и адресами.

# 2.3 Политики

## Политики

Политики определяют, **что разрешено** делать с цифровыми активами внутри вашего рабочего пространства и **на каких условиях**: какие транзакции проходят автоматически, какие требуют ручного аппрува, какие должны дополнительно проверяться AML-провайдером, а какие блокируются полностью.

Раздел описывает все типы политик, их правила и параметры. Используйте его как справочник при настройке политик в **Customer Console**: **Workspace → Policies**.

### Типы политик

В системе существует три независимых типа политик. Каждый workspace имеет ровно по одной политике каждого типа; при первом обращении к разделу они автоматически создаются с безопасными значениями по умолчанию.

| Тип политики | На какой стадии применяется | Что решает |
|---|---|---|
| **Transfer** | На приёме исходящей транзакции, до выхода в блокчейн | Разрешить транзакцию автоматически, потребовать ручной аппрув или заблокировать |
| **AML Screening** | До отправки на проверку AML-провайдеру | Нужно ли вообще обращаться к провайдеру (экономия запросов и времени) |
| **AML Post-Screening** | После получения результата AML-провайдера | Принять транзакцию или заморозить из-за рисков |

### Общая модель

Все политики устроены одинаково: это **упорядоченный список правил** (rules). Правила оцениваются по возрастанию поля `position` — побеждает **первое подходящее**.

**Главное правило настройки**

Размещайте более **специфичные** правила **выше** по списку, более общие — ниже. Дефолтные правила (`allow all` / `block all` / `pass all` / `accept all`) почти всегда находятся на последних позициях и нужны как «ловушка» для всего, что не попало в предыдущие.

### Что общего у всех правил

Независимо от типа политики у каждого правила есть:

- **Position** — порядковый номер. Чем меньше — тем выше приоритет.
- **Name** — человекочитаемое имя; помогает в логе аудита и в UI.
- **Action** — что сделать, если правило сматчилось (см. таблицы на страницах конкретных политик).
- **Условия матчинга** — набор фильтров. Правило срабатывает, только если **все** условия выполнены одновременно (логическое «И»).

Внутри условий, где это поддерживается, можно выбирать:

- **Конкретный список** (например, `vault_uuids` — только эти vault'ы);
- **«Любой»** (`any_*` флаг — `any_vault`, `any_asset` и т. п.);
- **Не указывать поле вовсе** — обычно эквивалентно «любому».

### Видимость и аудит

- Изменения политик хранятся версионировано: можно посмотреть историю изменений и откатиться на предыдущую версию.
- Все изменения публикуются в аудит-лог.
- Активация / деактивация политики — отдельная операция; деактивированная политика трактуется системой как «политики нет» и приводит к дефолтному действию (`ALLOW` / `PASS` / `ACCEPT` соответственно).

### Кто может управлять политиками

Управление политиками регулируется правами:

| Право | Кому даётся по умолчанию | Что разрешает |
|---|---|---|
| `VIEW_POLICIES` | `OWNER`, `ADMIN`, `NON_SIGNING_ADMIN`, `PLATFORM_SUPPORT`, `PLATFORM_ADMIN` | Просмотр политик и их правил |
| `VIEW_POLICIES_HISTORY` | те же | Просмотр истории версий |
| `EDIT_POLICIES` | `OWNER`, `ADMIN`, `NON_SIGNING_ADMIN`, `PLATFORM_ADMIN` | Изменение правил, откат версии |
| `ACTIVATE_POLICY` / `DISABLE_POLICY` | только `PLATFORM_ADMIN` | Включение/ выключение политики на уровне платформы |

Полный список прав и их назначения по ролям доступен в Customer Console в разделе **Workspace → Users → Roles**.

## Transfer policy

**Transfer policy** определяет, как обрабатывать **исходящие транзакции (withdrawal)** ещё до того, как они уходят в блокчейн. Это центральный элемент управления рисками: на этой стадии решается, кому, куда, сколько и при каких условиях разрешено выводить средства.

### Когда применяется

Проверка Transfer policy — третья стадия обработки исходящей транзакции: после базовых валидаций (баланс, fee, активность vault) и AML Screening, но до того, как транзакция уходит на подписание и в блокчейн. В этой точке решение политики — последний барьер перед началом подписания.

### Action — что делает правило

| Action | Что произойдёт с транзакцией |
|---|---|
| `ALLOW` | Транзакция проходит автоматически, без аппрува. Если правило указывает подписантов — они будут назначены этой транзакции. |
| `REQUEST_APPROVAL` | Транзакция переходит в статус `awaiting_approvals`. Указанным аппруверам уходит push/WS-уведомление; решение принимается по правилам апрува. |
| `BLOCK` | Транзакция получает терминальный статус `blocked_by_policy` и больше никак не двинется — для повторной попытки клиент должен создать новую. |

### Структура правила

Правило состоит из четырёх логических блоков:

```
WHO  →  Initiator (кто инициировал транзакцию)
FROM →  Source    (откуда выводим)
TO   →  Destination (куда выводим)
WHAT →  Asset + Limitation (какой актив и в каких пределах)
```

Все блоки соединяются логическим **И**: правило срабатывает, только если совпадают все указанные условия. Дополнительно для `ALLOW` и `REQUEST_APPROVAL` правило описывает, кто будет подписывать ( `Signer` ) и кто должен подтвердить ( `Approval` ).

#### INITIATOR — ИНИЦИАТОР ТРАНЗАКЦИИ

Кто запустил withdrawal: пользователь или сервисный аккаунт.

| Поле | Описание |
|---|---|
| `any_initiator` | `true` — правило срабатывает для любого инициатора. Удобно для широких разрешающих/блокирующих правил. |
| `initiators` | Карта `uuid → type`. Конкретный список инициаторов; `type` — `USER` или `SERVICE_ACCOUNT`. Если запросивший подходит и его тип совпадает — матч. API-пользователя указывайте с типом `USER`: с `SERVICE_ACCOUNT` правило на него не сработает. |

**Зачем разделять User и Service Account**

Часто полезно разрешать сервисным аккаунтам только мелкие автоматические выводы (например, выплаты пользователям до 100 USD), а крупные операции отдавать пользователям с обязательным аппрувом. Это делается двумя правилами с разными `initiators`.

#### SOURCE — ОТКУДА СПИСЫВАЕМ

Источник средств для исходящей транзакции — всегда один из vault workspace.

| Поле | Описание |
|---|---|
| `any_source` | `true` — любой vault workspace. |
| `vault.any_vault` | `true` — любой vault (эквивалент `any_source`). |
| `vault.vault_uuids` | Конкретный список vault, к которым применимо правило. |
| `whitelisted.*` | Зарезервировано — реальные исходящие транзакции всегда идут из vault, но поле есть для симметрии с Destination. |

#### DESTINATION — КУДА ОТПРАВЛЯЕМ

Назначение бывает нескольких типов: внутренний vault (перевод между vault workspace), whitelisted-кошелёк (адрес из белого списка) или **one-time address** (свободно введённый адрес).

| Поле | Описание |
|---|---|
| `any_destination` | `true` — любое назначение. Самое широкое условие; используйте в дефолтных правилах. |
| `one_time_addresses_only` | `true` — правило срабатывает **только** для свободно введённых адресов. Хороший крючок для требования аппрува на любые «новые» адреса. |
| `vault.any_vault` / `vault.vault_uuids` | Перевод во внутренний vault: любой или конкретный. |
| `whitelisted.any_whitelisted_address` | Любой адрес из whitelist workspace. |
| `whitelisted.any_whitelisted_internal_address` | Любой whitelisted **внутренний** адрес. |
| `whitelisted.any_whitelisted_external_address` | Любой whitelisted **внешний** адрес. |
| `whitelisted.any_whitelisted_contract_address` | Любой whitelisted адрес **смарт-контракта**. |
| `whitelisted.whitelisted_external_addresses` | Конкретный список внешних адресов. |
| `whitelisted.whitelisted_internal_addresses` | Конкретный список внутренних. |
| `whitelisted.whitelisted_contract_addresses` | Конкретный список контрактных. |

**One-time addresses**

Включите `one_time_addresses_only` в правило с `REQUEST_APPROVAL` — это самый простой способ гарантировать, что любая транзакция на «новый» адрес проходит через ручное подтверждение.

#### ASSET — КАКОЙ АКТИВ И КАКОЙ ПОРОГ

Описывает актив и порог суммы, при превышении которого правило активируется.

| Поле | Описание |
|---|---|
| `any_asset` | `true` — правило действует для любого актива. В этом режиме сумма всегда сравнивается в **USD-эквиваленте**. |
| `asset_id` | Конкретный актив (например, `BTC`, `ETH`, `USDT-TRC20`). |
| `amount` | Пороговое значение (decimal-строка). Правило **матчится, если сумма транзакции** `≥ amount`. Для дефолтных правил укажите `0` — тогда правило применяется к любым суммам. |
| `currency` | `USD` — сравнивать в USD-эквиваленте; `NATIVE` — в единицах самого актива (например, в BTC). При `any_asset = true` всегда используется `USD`. |

**Семантика порога**

Условие — **«сумма больше или равна порогу»**. Чтобы правило срабатывало для всех сумм, поставьте `amount = 0`. Чтобы разделить «мелкие» и «крупные» переводы, заведите два правила: одно с `amount = 1000` (например, требует аппрув от 1000 USD), второе ниже по списку с `amount = 0` (мелкие — auto-allow).

#### SIGNER — КТО ПОДПИШЕТ

Применимо для `ALLOW` и `REQUEST_APPROVAL`.

| Поле | Описание |
|---|---|
| `signer_uuids` | Конкретный список подписантов, которым будет назначена транзакция. |
| `group_uuids` | Группы подписантов: транзакция уйдёт тем участникам групп, у кого есть право `SIGN_TRANSACTION`. |
| `is_initiator_signer` | `true` — подписывает сам инициатор (если у него есть право `SIGN_TRANSACTION` и привязанное устройство). |

**Если среди подписантов есть API-пользователи, подписывают только они**

Когда в `signer_uuids` попадает API-пользователь, транзакция уходит на подпись только API-пользователям правила — остальные подписанты этого правила не привлекаются. API-пользователь подписывает программно, поэтому **подпись** ставится без ожидания человека.

Считаются только те API-пользователи, что перечислены в `signer_uuids` напрямую. Если API-пользователь попал в правило лишь как участник группы из `group_uuids`, правило работает как обычно.

Аппрувов это не отменяет: у правила с действием `REQUEST_APPROVAL` подтверждения по-прежнему собираются с указанных аппруверов, и без них перевод не пойдёт.

Условие: у API-пользователя должно быть действующее право `SIGN_TRANSACTION` и не заблокированная учётная запись. Если это не так, правило работает как обычно — транзакция уходит всем перечисленным подписантам.

Чтобы перевод подписывали именно сотрудники, не включайте API-пользователей в `signer_uuids` этого правила.

#### APPROVAL — КТО И КАК ДОЛЖЕН ПОДТВЕРДИТЬ

Применимо только для `REQUEST_APPROVAL`. Описывает, какие именно подписи аппруверов нужны, чтобы транзакция была одобрена.

| Поле | Описание |
|---|---|
| `initiator_can_approve` | `true` — инициатор может сам выступить в роли одного из аппруверов. По умолчанию `false`, чтобы избежать «само-аппрува». |
| `users_sets` | Карта именованных наборов аппруверов. Каждый набор — `{ user_uuids, threshold }`. Имя набора (`A`, `B`, …) используется в `expression`. |
| `expression` | Логическое выражение, описывающее, какие наборы должны выполниться. Поддерживает `&&`, `\|\|`, `(…)` и имена наборов. Примеры: `A`, `A && B`, `(A && B) \|\| C`. |

Внутри одного набора `threshold` задаёт минимальное число подписей. Например, набор `A` со списком из 5 финансистов и `threshold = 2` требует подписи **любых двух** из них. Если выражение `(A && B) || C`, транзакция проходит, если выполнен **набор C целиком** или **оба набора A и B**.

**Имена наборов**

По умолчанию один безымянный набор называется `A` (`DefaultApprovalSetName = "A"`). При работе через UI имена обычно проставляются автоматически.

#### LIMITATION — ЛИМИТ НА СУММУ

Описывает, **как** считается порог суммы из блока Asset: по одной транзакции или накопительно за окно времени.

##### `SINGLE_TX` (по одной транзакции)

Самый простой вариант. Правило срабатывает, если сумма **текущей** транзакции `≥ asset.amount`. Историю предыдущих транзакций не учитывает.

##### `TIMEFRAME` (накопительный лимит за окно)

Правило срабатывает, если **сумма всех подходящих транзакций за последние `hours` часов** (включая текущую) превысит порог.

Поля `*_mode` определяют, как именно агрегируются суммы:

| Поле | Значения | Что означает |
|---|---|---|
| `hours` | положительное число | Размер окна в часах (sliding window). Максимум — 15 дней (360 часов). |
| `initiator_mode` | `PER_INITIATOR` | Считать отдельно для каждого инициатора. У каждого пользователя/SA — свой счётчик. |
| | `ACCUMULATED_INITIATORS` | Один общий счётчик на всех инициаторов вместе. |
| `source_mode` | `PER_SOURCE` | Отдельный счётчик для каждого vault-источника. |
| | `ACCUMULATED_SOURCES` | Общий счётчик по всем источникам. |
| `destination_mode` | `PER_DESTINATION` | Отдельный счётчик для каждого получателя. |
| | `ACCUMULATED_DESTINATIONS` | Общий счётчик по всем получателям. |

**Примеры**

- **«Каждый сотрудник — не более 10 000 USD в сутки»**: `hours = 24`, `initiator_mode = PER_INITIATOR`, `source_mode = ACCUMULATED_SOURCES`, `destination_mode = ACCUMULATED_DESTINATIONS`, `amount = 10000`, `currency = USD`.
- **«Со всех vault'ов не больше 1 BTC в час суммарно»**: `hours = 1`, все `*_mode = ACCUMULATED_*`, `asset_id = BTC`, `amount = 1`, `currency = NATIVE`.

**Резервирование лимита**

Когда транзакция входит в `awaiting_approvals`, её сумма **резервируется** в счётчике и блокирует дальнейшие переводы до достижения порога. Если транзакция позже отклонится (`declined` / `expired` / `failed`), резерв автоматически освобождается и счётчик уменьшается.

### Поведение по умолчанию

При первом обращении к политике в workspace создаются два правила:

1. **Default allow** (position 1) — `ALLOW`, `any_initiator`, `any_source`, `any_destination`, `any_asset`, `amount=0`, `is_initiator_signer=true`. Разрешает любые транзакции с подписанием инициатором.
2. **Default block** (position 2) — `BLOCK`, помечен как `global`, не редактируется. Срабатывает, если ни одно пользовательское правило не подошло (страховка от «дыр»).

Если политика отсутствует или деактивирована — все транзакции **автоматически одобряются** системой с пометкой `no active transfer policy, transaction automatically approved`.

**Не оставляйте default allow в проде**

Дефолтное `allow all` создано исключительно для быстрого старта. В реальной конфигурации замените или сдвиньте его вниз, чтобы выше срабатывали правила с конкретными лимитами и аппрувами.

### Типовые сценарии

#### 1. Whitelist-only режим

Разрешить переводы только на адреса из white-list, всё остальное — блокировать.

| # | Action | Destination | Asset | Limitation |
|---|---|---|---|---|
| 1 | `ALLOW` | `whitelisted.any_whitelisted_address` | `any_asset`, `0 USD` | `SINGLE_TX` |
| 2 | `BLOCK` | `any_destination` | `any_asset`, `0 USD` | `SINGLE_TX` |

#### 2. Двойной контроль для крупных сумм

Мелкие — авто-разрешать, крупные — два аппрува, очень крупные — блокировать.

| # | Action | Asset.amount/currency | Approval |
|---|---|---|---|
| 1 | `BLOCK` | `100000 USD` | — |
| 2 | `REQUEST_APPROVAL` | `1000 USD` | `A && B` (директор и финансист) |
| 3 | `ALLOW` | `0 USD` | — |

#### 3. Любой новый адрес — через подтверждение

| # | Action | Destination | Approval |
|---|---|---|---|
| 1 | `REQUEST_APPROVAL` | `one_time_addresses_only = true` | `A` (любой из админов) |
| 2 | `ALLOW` | `any_destination` | — |

#### 4. Дневной лимит на сотрудника

| # | Action | Initiator | Limitation |
|---|---|---|---|
| 1 | `BLOCK` | `any_initiator` | `TIMEFRAME` 24h, `PER_INITIATOR`, accumulated by src/dst, `amount=10000 USD` |
| 2 | `ALLOW` | `any_initiator` | `SINGLE_TX`, `0` |

## AML Screening policy

**AML Screening policy** решает, **нужно ли вообще обращаться к внешнему AML-провайдеру** для конкретной транзакции. Это «локальный фильтр»: позволяет пропускать заведомо безопасные операции мимо платного скрининга и экономить как деньги, так и время обработки.

> **Что такое AML-провайдер**
>
> Внешний сервис (CoinKyt и аналоги), который анализирует адреса и on-chain историю на предмет рисков: связь с миксерами, санкционными адресами, даркнет-маркетами и т. п. AML Screening решает, **нужно ли спрашивать провайдера**; интерпретацией ответа занимается отдельная политика — AML Post-Screening.

### Когда применяется

AML Screening — **первая стадия** пайплайна транзакции, ещё до policy review. Применяется к **исходящим** транзакциям (withdrawal).

Стадия пропускается полностью, если:

- в workspace **не подключён** AML-провайдер;
- это перевод внутри одного workspace ( `source` и `destination` в одном workspace);
- инициатор — администратор и включена опция «admin bypass».

### Action — что делает правило

| Action | Что произойдёт |
|---|---|
| `AML_SCREENING_PASS` | Транзакция **пропускается** мимо AML-провайдера. Дальше — обычный pipeline (policy review). |
| `AML_SCREENING_SCREEN` | Транзакция отправляется на проверку AML-провайдеру. Результат потом обрабатывается в AML Post-Screening. |

### Структура правила

Правило срабатывает, если **все** указанные условия выполнены одновременно.

| Блок | Описание |
|---|---|
| `source` | Источник транзакции. Те же поля, что и в Transfer policy → Source. |
| `destination` | Назначение. См. Transfer policy → Destination. |
| `asset` | Актив и порог суммы. См. Transfer policy → Asset. Условие — `amount транзакции ≥ asset.amount`. |
| `direction` | Направление: `AML_SCREENING_DIRECTION_OUTGOING`, `AML_SCREENING_DIRECTION_INCOMING` или не указано (= любое). |

> **Когда указывать `direction`**
>
> Большинство правил AML Screening нацелены на исходящие транзакции ( `OUTGOING` ). Поле `direction` нужно, если для одного workspace вы хотите по-разному скринить депозиты и выводы — система обрабатывает их по разным цепочкам.

### Поведение по умолчанию

При первом обращении в политике создаётся одно правило:

- **Default pass** (position 1) — `AML_SCREENING_PASS`, `any_source`, `any_destination`, `any_asset`, `amount = 0`. Все транзакции по умолчанию **проходят мимо AML-провайдера**.

Если политика отсутствует или деактивирована — поведение такое же, как у `default pass` : все транзакции пропускаются без скрининга.

Если все правила настроены, но **ни одно не сматчилось**, дефолтным fallback'ом является `SCREEN` — система предпочитает «лишний раз проверить», чем пропустить подозрительную транзакцию.

### Типовые сценарии

#### 1. Скринить только выводы выше порога

Не тратить запросы на маленькие суммы.

| # | Action | Direction | Asset |
|---|---|---|---|
| 1 | `SCREEN` | `OUTGOING` | `any_asset, 1000 USD` |
| 2 | `PASS` | (не указано) | `any_asset, 0` |

#### 2. Всегда скринить one-time addresses

Свободно введённые адреса — всегда через AML, остальные — пропускать.

| # | Action | Destination | Asset |
|---|---|---|---|
| 1 | `SCREEN` | `one_time_addresses_only = true` | `any_asset, 0` |
| 2 | `PASS` | `any_destination` | `any_asset, 0` |

#### 3. Скринить переводы на конкретный «горячий» список адресов

Когда compliance ведёт собственный watchlist whitelisted-адресов, которые требуют дополнительной проверки.

| # | Action | Destination |
|---|---|---|
| 1 | `SCREEN` | `whitelisted.whitelisted_external_addresses = [addr_1, addr_2, …]` |
| 2 | `PASS` | `any_destination` |

## AML Post-Screening policy

**AML Post-Screening policy** интерпретирует **результат**, полученный от AML-провайдера, и решает: принять транзакцию как «чистую» или **отклонить** её из-за обнаруженных рисков.

В отличие от AML Screening, эта политика работает с уже готовой оценкой: разбивкой истории адреса по категориям риска (`mixer`, `darknet`, `sanctions` и т. п.) с долей каждой категории в общем exposure (в процентах, 0–100).

### Когда применяется

После того как AML-провайдер вернул результат скрининга. Применяется к обеим направлениям:

- **исходящие** — блокирует withdrawal с переводом транзакции в статус `frozen` (reason = `aml`);
- **входящие (депозиты)** — приводит к заморозке vault, на который пришёл депозит.

### Action — что делает правило

Логика AML Post-Screening **инвертирована** относительно других политик: правила здесь описывают **рисковые случаи**, при которых транзакцию нужно отклонить.

| Action | Что произойдёт |
|---|---|
| `AML_POST_SCREENING_REJECT` | Транзакция отклоняется. Для withdrawal — freeze с reason `aml`; для deposit — freeze vault. Снять заморозку может админ через `UNFREEZE_TRANSACTION` / `UNFREEZE_VAULT`. |
| `AML_POST_SCREENING_ACCEPT` | (Дефолтное действие.) Используется как «штатное» поведение, когда ни одно reject-правило не сматчилось. |

> **Дефолт — `ACCEPT`**
>
> Все правила в этой политике пишутся **только** под reject. Если ни одно reject-правило не сработало, транзакция автоматически `ACCEPTED`. Создавать правила с `ACCEPT` вручную обычно не нужно.

### Структура правила

Правило срабатывает, если **все** указанные условия выполнены одновременно **и** среди рисков, которые вернул провайдер, нашёлся хотя бы один, удовлетворяющий критериям правила.

#### ФИЛЬТРЫ НА САМУ ТРАНЗАКЦИЮ

| Блок | Описание |
|---|---|
| `direction` | Направление: `AML_POST_SCREENING_DIRECTION_OUTGOING`, `..._INCOMING` или `..._ANY` (по умолчанию — любое). |
| `asset` | Актив и порог суммы. Если `asset.amount` задан — правило применяется, только когда сумма транзакции `≥ amount` (в `USD` или `NATIVE`). Поведение полей идентично Transfer policy → Asset. |

КРИТЕРИИ РИСКА

| Поле | Описание |
|---|---|
| `category` | Категория риска. Поле `name` — конкретная категория (например, `mixer`, `darknet_market`, `sanctions`, `gambling`, …). Флаг `any_category = true` — подходит любая категория. |
| `min_exposure_percent` | Порог по **доле exposure** категории в истории адреса, **0–100**. Условие: среди категорий из ответа провайдера есть хотя бы одна, проходящая фильтр `category`, с долей `≥ rule.min_exposure_percent`. Если поле не задано, а `category` указана — достаточно любого ненулевого появления категории. |
| `exposure_type` | Тип контакта. Пока принимается только `..._ANY` и в оценке не участвует — текущие провайдеры не разделяют прямой и косвенный контакт. Поле зарезервировано под провайдеров с таким разделением. |

**Логика матча**

Провайдер возвращает **разбивку exposure по категориям** — долю каждой категории в истории адреса. Правило срабатывает, если среди них есть хотя бы одна, проходящая фильтр `category`, с долей не ниже порога. В отклонении (`Rejected`) транзакция получает список совпавших категорий с их долями — он попадает в reason заморозки и аудит-лог.

**Устаревшее поле**

Поле `risk_score` (шкала 0–10) устарело: старые правила автоматически переведены в `min_exposure_percent` (`×10`).

### Поведение по умолчанию

При первом обращении создаётся одно правило:

- **Default accept** (position 1) — `AML_POST_SCREENING_ACCEPT`, `direction = ANY`, `any_asset`, `amount = 0`, `min_exposure_percent = 0`.

При отсутствующей или деактивированной политике, а также при пустом списке правил — транзакция всегда `ACCEPTED`.

**Не оставляйте политику без reject-правил**

Дефолтное `accept` ничего не блокирует. Чтобы AML-проверка имела смысл, добавьте хотя бы одно reject-правило (см. сценарии ниже).

### Типовые сценарии

**1. Жёсткий запрет высокорискованных адресов**

Больше 70% истории адреса приходится на любую одну рискованную категорию — отклонить.

| # | Action | Direction | category | min_exposure_percent |
|---|---|---|---|---|
| 1 | `REJECT` | `ANY` | `any_category = true` | `70` |

### 2. Чёрный список категорий

Любое появление этих категорий в истории адреса — отклонить, независимо от доли.

| # | Action | category | min_exposure_percent |
|---|---|---|---|
| 1 | `REJECT` | `sanctions` | — |
| 2 | `REJECT` | `darknet_market` | — |
| 3 | `REJECT` | `mixer` | — |

### 3. Порог по доле рискованной категории

Отклонять, если больше четверти истории адреса приходится на любую одну категорию, а также при заметной доле санкционных контактов.

| # | Action | category | min_exposure_percent |
|---|---|---|---|
| 1 | `REJECT` | `sanctions` | `5` |
| 2 | `REJECT` | `any_category = true` | `25` |

### 4. Мягче для маленьких сумм, строже для крупных

Депозиты до 100 USD — пропускать почти всё; крупные — резать жёстче.

| # | Action | Direction | Asset | category | min_exposure_percent |
|---|---|---|---|---|---|
| 1 | `REJECT` | `INCOMING` | `any_asset, 100 USD` | `any_category = true` | `50` |
| 2 | `REJECT` | `INCOMING` | `any_asset, 0` | `any_category = true` | `80` |

**Что делать с заблокированной транзакцией**

- **Withdrawal в `frozen`** — администратор с правом `UNFREEZE_TRANSACTION` может снять заморозку через UI; транзакция вернётся к предыдущему статусу и продолжит обработку.
- **Замороженный vault** — снимается через `UNFREEZE_VAULT`. До разморозки на vault нельзя принимать новые депозиты.
- В обоих случаях reason содержит совпавшие категории с их долями exposure — это удобно для compliance-отчёта.

# 2.4 Транзакции

## Жизненный цикл транзакции

Исходящая транзакция проходит путь от создания пользователем до финализации в блокчейне через несколько этапов: автоматические проверки, согласование, MPC-подписание и отправка в сеть. На каждом переходе платформа присваивает транзакции новый статус и отправляет webhook-событие — по ним ваше приложение может отслеживать процесс, ничего не опрашивая.

**Схема**