

Описание технической архитектуры программного обеспечения

Программное обеспечение: драйвер банковских POS-терминалов для платформы Эвотор

1. Назначение системы

Драйвер обеспечивает связь между кассой Эвотор и банковскими POS-терминалами. Касса инициирует платёжные операции (оплата, возврат, закрытие сессии). Драйвер передаёт запросы на терминал и возвращает результат `PayResult` на кассу.

Фактический режим работы — USB: терминал виден как CDC ACM COM-порт. Обмен с терминалом не идёт напрямую из платёжной логики: он проходит через локальный прокси (TCP-порт 9876), который упаковывает SA-пакеты в кадры `UsbProxy` (TLV) и записывает их в USB COM-порт.

2. Архитектурный стиль (в)

Система построена в **слоистом стиле (layered architecture)** с двумя направляющими дополнениями:

- **Плагин / адаптер к внешней платформе.** Драйвер не является самостоятельной кассой: он подключается к Эвотор как Android-сервис с AIDL-интерфейсами SDK (`IUsbDriverManagerService`, `IPaySystemService`, `IPaySystem`).
- **Прокси-канал (проxy) между прикладным обменом и транспортом.** Платёжная логика всегда работает через сокет `127.0.0.1:9876`. Прокси изолирует USB COM-порт от нескольких потоков и единообразно маркирует трафик (тип сообщения, `connectionId`).

Слои строгие: верхний слой знает только интерфейс соседнего нижнего слоя и не обращается к USB или TLV напрямую. Преобразование данных идёт конвейером (`pipes-and-filters`): `PayInfo` → SA-пакет → TCP → TLV `UsbProxy` → USB CDC ACM → терминал.

Стиль задаёт правила для всех элементов:

- элементы группируются по слоям ответственности, а не по вендору терминала;
- внешняя видимость слоя — только его интерфейс (AIDL, `callback`, сериализация);
- сотрудничество — синхронный вызов сверху вниз и асинхронный `callback` снизу вверх;
- соединение транспорта всегда проходит через прокси, а не через прямую запись в COM из `AisinoDevice`.

Рис. 1. Архитектурный стиль: слои, интерфейсы, направление связей

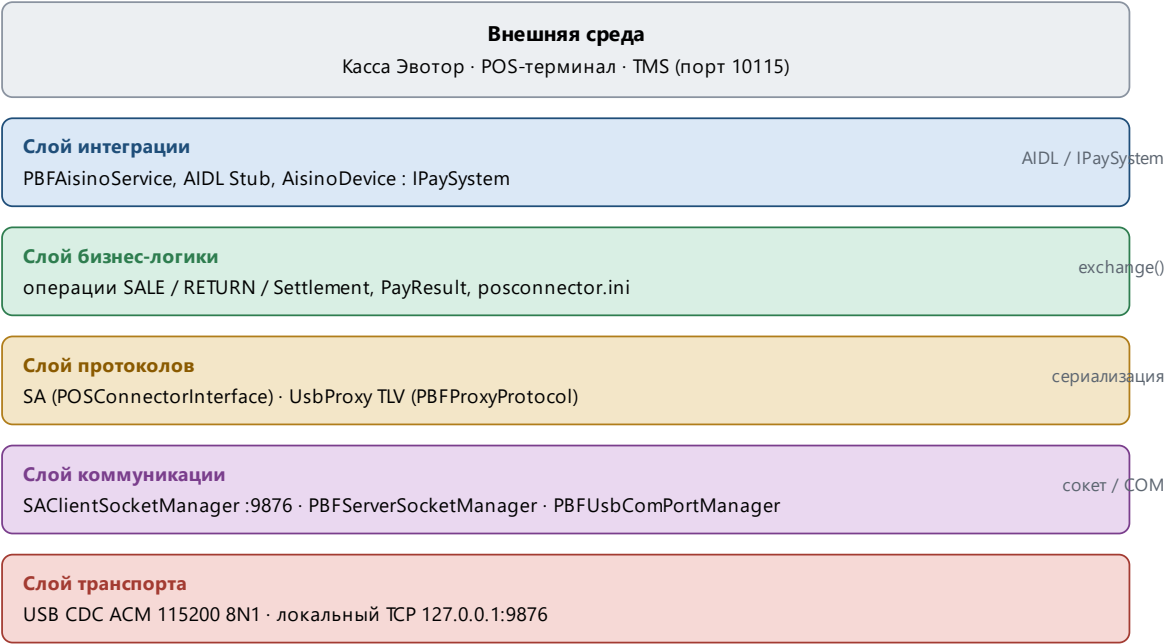


Рисунок 1. Слоистый стиль с адаптером к Эвотор и прокси-каналом к USB

3. Структурные элементы и их интерфейсы (а)

Система составлена из элементов четырёх ролей: точка входа платформы, экземпляр устройства, протокольные преобразователи и каналы ввода-вывода. Внешне видимое поведение каждого элемента задано интерфейсом.

3.1. Элементы интеграции с платформой Эвотор

Элемент	Роль	Интерфейс	Внешне видимое поведение
PBFAisinoService	Android-сервис, точка входа драйвера	Intent-filter SDK Эвотор; onBind()	По action возвращает Binder: USB-менеджер, виртуальный менеджер или платёжный сервис. Создаёт и уничтожает экземпляры устройств по instanceId.
AisinoDriverManagerStub	USB-драйвер менеджер	IUsbDriverManagerService.Stub	addUsbDevice(UsbDevice, usbPortPath) → instanceId; destroy(instanceId).
AisinoVirtualDriverManagerStub	Менеджер виртуального устройства	IVirtualDriverManagerService.Stub	addNewVirtualDevice(), recreateNewVirtualDevice(), destroy().
AisinoPaySystemStub	Фасад платёжных операций	IPaySystemDriverService.Stub	Делегирует вызовы в getMyDevice(instanceId): payment, payback, closeSession, openServiceMenu, реквизиты терминала.
AisinoDevice	Экземпляр подключённого терминала	IPaySystem, IClientSocketHandler	Исполняет операцию, ждёт ответ, формирует PayResult. При ошибке — PayResult с null-кодом и текстом на чеке.

3.2. Элементы протоколов

Элемент	Роль	Интерфейс	Внешне видимое поведение
---------	------	-----------	--------------------------

InitParser / PosConnectorItem	Конфигурация операций SA	загрузка posconnector.ini	Для тега операции (SALE, RETURN, Settlement) отдаёт набор полей запроса и ответа.
SAPacket, SA_Message, SA_Field	Модель SA-сообщения	IMessage, IField, IFormat	Сериализация/десериализация полей AMOUNT, RRN, RECEIPT_DATA, TERMINAL_ID и др.
POSConnectorInterface	Кадрирование SA	статические методы обмена	getExchangeData(): STX + длина + тело + CRC16; deserialize(); служебные msgACK, msgFinalizing.
PBFProxyProtocol	Маршрутизатор UsbProxy	IClientSocketHandler, IServerSocketHandler, IComPortHandler	Упаковывает данные сокета в TLV (PMT_SEND_DATA) и пишет в COM; кадры с COM (PMT_EVENT_DATA) отдаёт в сокет по connectionId.
ReceiptDataParser	Разбор слипа	разбор RECEIPT_DATA	Преобразует поле чека терминала в массив строк для PayResult.

3.3. Элементы коммуникации и вспомогательные

Элемент	Роль	Интерфейс	Внешне видимое поведение
PBFServerSocketManager	Локальный TCP-сервер прокси	IServerSocketHandler	Слушает порт 9876, принимает клиентские соединения прикладного слоя.
SAClientSocketManager	Клиент SA-обмена	наследник PBFClientSocketManager; callback IClientSocketHandler	Подключается к 127.0.0.1:9876, отправляет SA-кадр, возвращает байты ответа.
PBFUsbComPortManager	Владелец USB CDC ACM	IComPortHandler	Открывает порт по VID/PID, 115200 8N1, пишет кадры, асинхронно читает через SerialInputOutputManager.
LogServerSocketManager	Диагностический канал TMS	TCP :10115, команда 0x21	Собирает ZIP логов и отдаёт клиенту с MD5.
SettingsActivity, ConfigParser	Конфигурация	config.xml, config.json, SharedPreferences	IP/порт прокси, таймаут операции, параметры UI.

Рис. 2. Выбор структурных элементов и интерфейсов

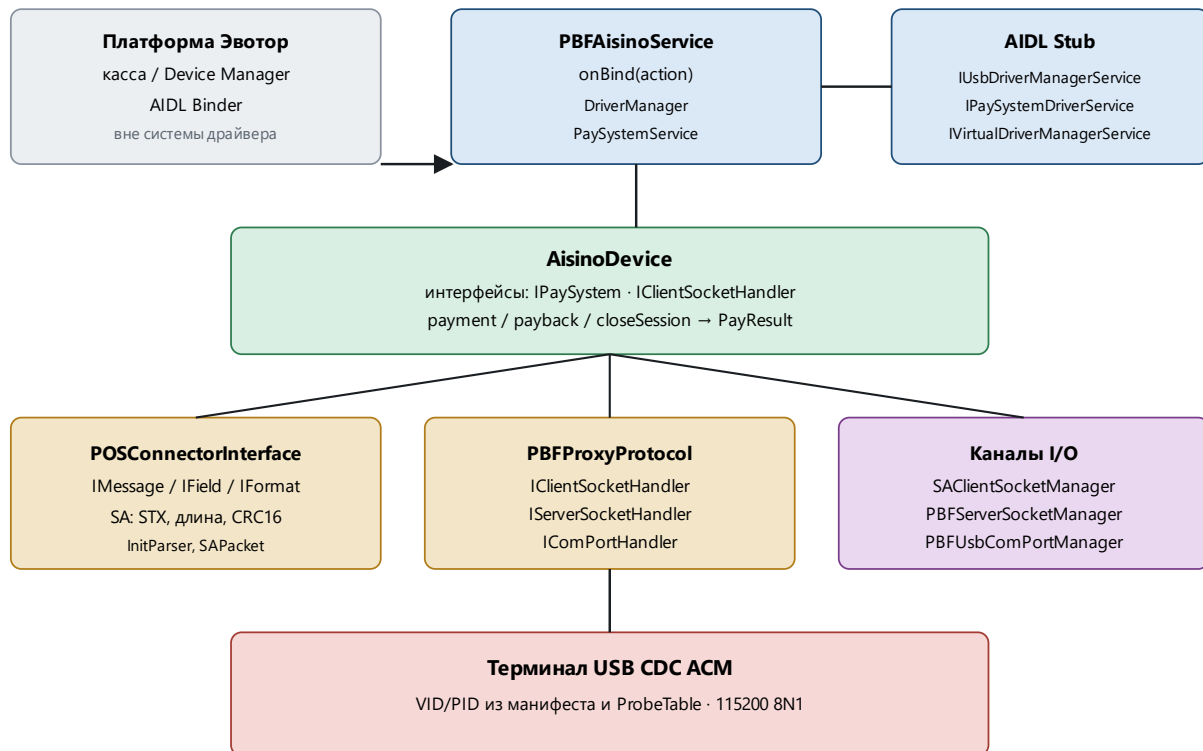


Рисунок 2. Элементы системы и интерфейсы, через которые они видны соседям

4. Соединение выбранных элементов (б)

Элементы соединены тремя видами связей: AIDL-вызов кассы, локальный TCP-прокси и USB-последовательный канал. Сквозной путь USB-сценария:

```
Эвотор → IPaySystem.payment/payback → AisinoDevice.exchange()
→ SAClientSocketManager (127.0.0.1:9876)
→ PBFPProxyProtocol (TLV PMT_SEND_DATA)
→ PBFUsbComPortManager.writePort() → USB CDC ACM → терминал
```

Обратный путь: терминал → USB → COM → PMT_EVENT_DATA → клиентский сокет → AisinoDevice.OnDataFromClientSocket → разбор SA → PayResult.

Рис. 3. Соединение элементов (USB-сценарий)

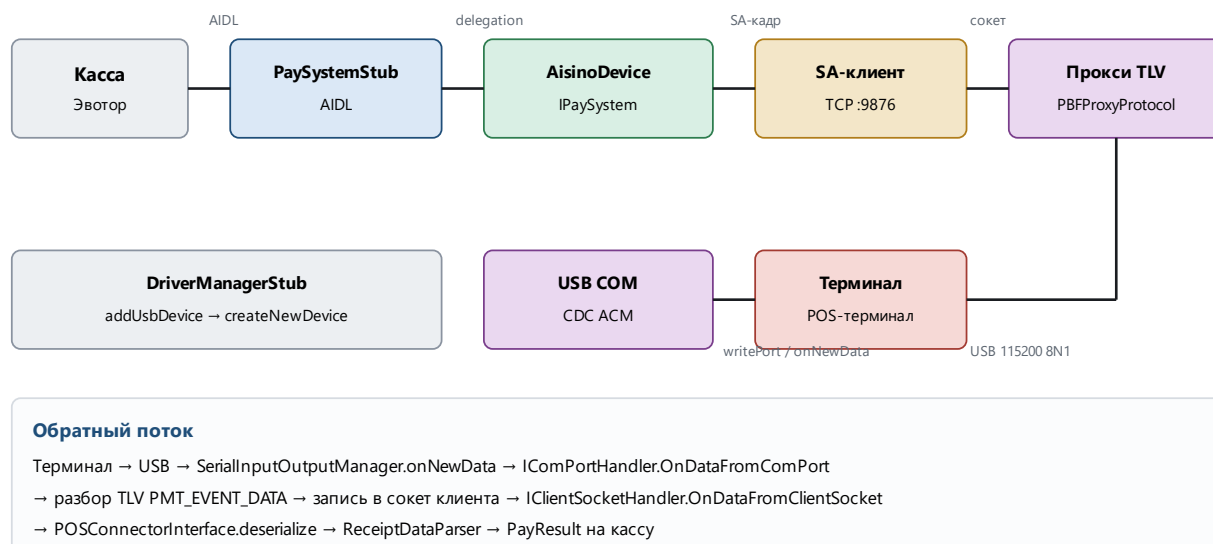


Рисунок 3. Соединение элементов: AIDL, локальный TCP-прокси, USB COM

4.1. Зачем введён локальный прокси

Это ключевое решение о соединении. Прикладной слой не пишет в COM напрямую. Все SA-данные идут в сервер 9876; прокси сериализует их в TLV и единолично владеет портом (блокировка записи). Так исключается гонка нескольких потоков за COM и сохраняется один путь обмена и для USB, и для будущего TCP-подключения к терминалу.

5. Блок-схема важнейших архитектурных решений

Ниже — решения, которые направляют организацию системы. Каждое решение фиксирует элемент, интерфейс и способ соединения.

Рис. 4. Важнейшие решения об организации программной системы

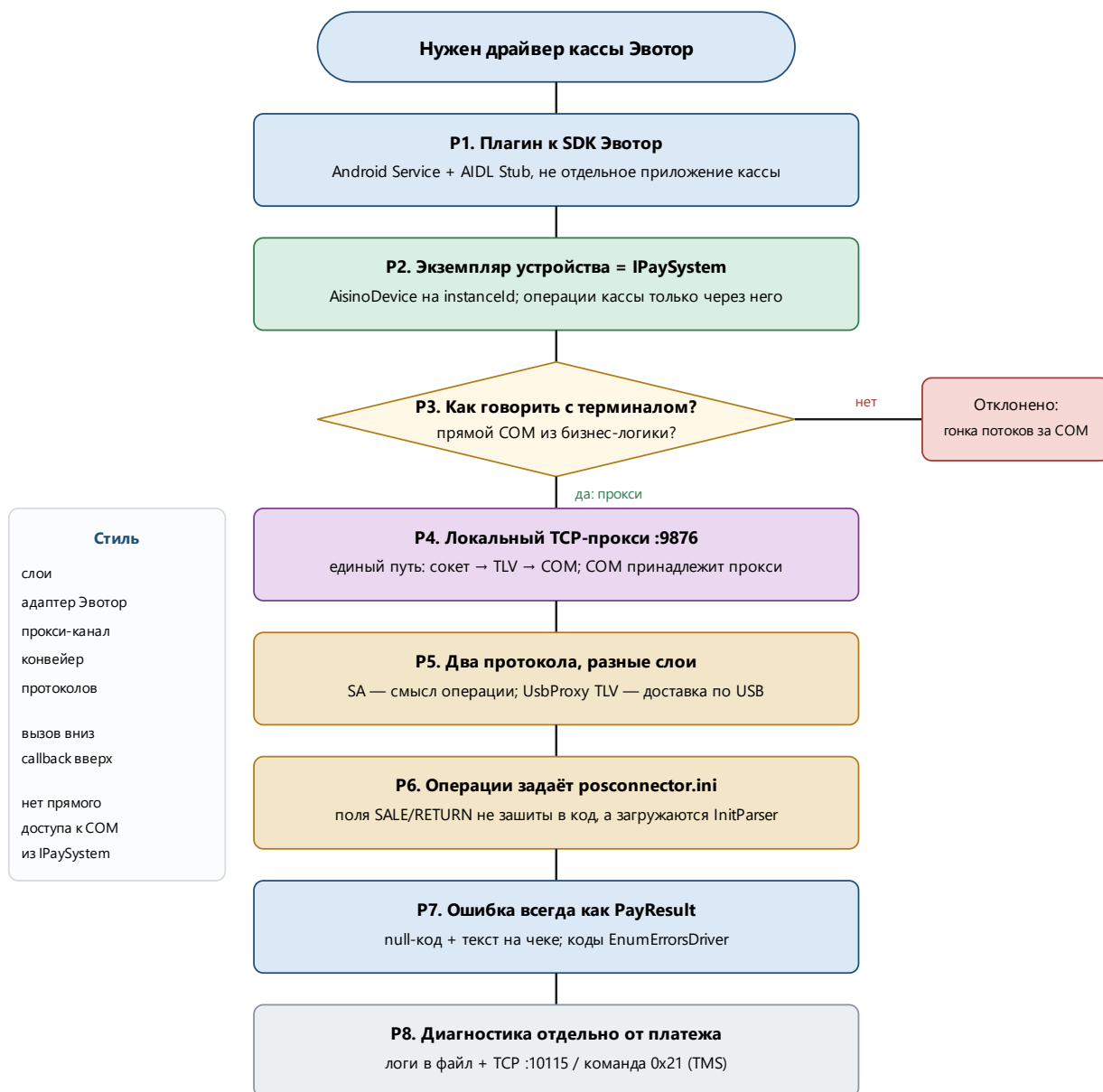


Рисунок 4. Блок-схема важнейших решений: элементы, интерфейсы, соединения, стиль

6. Сотрудничество элементов на сценарии оплаты

Сотрудничество подчиняется стилю: касса вызывает интерфейс слоя интеграции; устройство формирует SA; коммуникация доставляет кадр через прокси; ответ поднимается callback-ами и снова становится PayResult.

Рис. 5. Сотрудничество при payment()

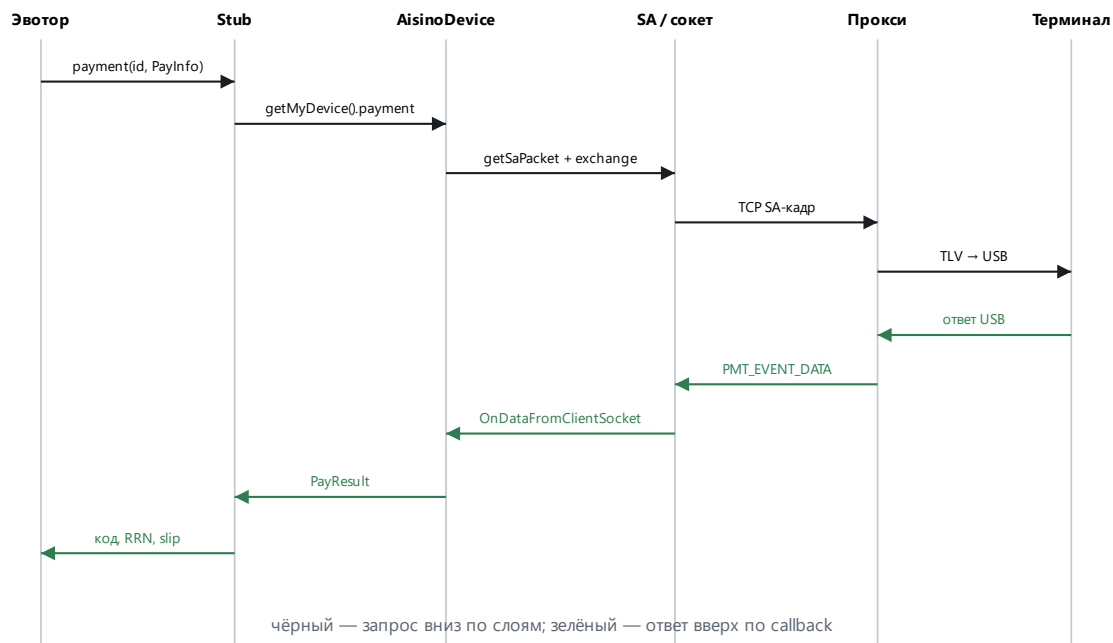


Рисунок 5. Сотрудничество элементов на операции оплаты

7. Сводка: как стиль направляет организацию

Вопрос организации	Решение стиля
Какие элементы входят в систему	Только слои интеграции, логики, протоколов, коммуникации и транспорта. UI настроек и логи — вспомогательные, не входят в платёжный конвейер.
Какими интерфейсами они видны	Снаружи — AIDL SDK Эвотор и IPaySystem. Внутри — callback-интерфейсы сокета и COM, абстракции SA (IMessage/IField).
Как элементы сотрудничают	Синхронный запрос кассы; внутри — обмен через сокет и асинхронное чтение USB; завершение — PayResult в том же вызове AIDL.
Как элементы соединены	Касса↔сервис — Binder; логика↔прокси — TCP 9876; прокси↔терминал — USB CDC ACM с TLV UsbProху. Прямого соединения логики с COM нет.
Почему выбран этот стиль	Слои отделяют контракт Эвотор от протокола банка и от USB-железа. Прокси даёт один безопасный владелец COM-порта. Конфигурация SA позволяет менять поля операций без смены архитектуры.