

TINKOFF ACQUIRING SDK FOR ANDROID

20.10.2017



Версия документа: 1.5

Содержание

1.	ТЕРМИНЫ И СОКРАЩЕНИЯ.....	4
2.	ОСНОВНЫЕ ПОЛОЖЕНИЯ	5
1.1	Требования.....	5
1.2	Подготовка к использованию.....	5
1.2.1	Подключение	5
1.2.2	Подготовка к работе.....	5
1.2.3	Выбор режима работы	8
2.	ПРОВЕДЕНИЕ ОПЛАТЫ	9
	Безопасная клавиатура	11
3.	ПРИВЯЗКА КАРТ	12
	Схема работы	13
4.	РЕКУРРЕНТНЫЙ ПЛАТЕЖ С ПРИВЯЗАННОЙ КАРТЫ	14
5.	ИНТЕГРАЦИЯ С ОНЛАЙН-КАССАМИ	15
6.	НАСТРОЙКА СТИЛЕЙ.....	17
7.	СТРУКТУРА	18
7.1	Core	18
7.1.2	Диаграмма классов.....	19
7.1.3	AcquiringApi.....	19
7.1.4	AcquiringSdk.....	23
7.1.5	CryptoUtils	24
7.1.6	Алгоритм формирования подписи запроса (Token).....	24
7.1.7	Алгоритм шифрования карточных данных	25
7.1.8	Пояснения	25
7.2	UI.....	25
7.2.1	Классы	26
7.2.2	Ввод реквизитов карты	26
7.2.3	3DS.....	27
7.2.4	Сканирование NFC.....	28
7.2.5	Список карт.....	28
7.3	Sample.....	29
7.4	ProGuard	29
8.	МАРШРУТ	30
9.	ПОДДЕРЖКА.....	32
10.	КОДЫ ОШИБОК API И ВОЗМОЖНЫЕ ИСКЛЮЧЕНИЯ	33

История изменений

Версия	Описание	Вносил изменения	Дата
1.0	Первоначальное составление документа	Никита Кашин	20.10.2017
1.1	Добавлено: <ul style="list-style-type: none"> • Структура SDK; • Описание AttachCardFormActivity; • Описание подключения к онлайн-кассам; • Описание настройки стилей; • Коды ошибок 	Никита Кашин	23.10.2017
1.2	Изменение структуры и форматирование документы	Никита Кашин	25.10.2017
1.3	Добавлено: <ul style="list-style-type: none"> • Скриншот безопасной клавиатуры; • Описание подключения через Maven; • Ограничение название платежа; • Скриншот привязки карты. Дополнена информация по привязке карты	Никита Кашин	26.10.2017
1.4	Незначительные изменения в структуре документа	Никита Кашин	31.10.2017
1.5	Добавлены изменения релиза 1.3.1	Никита Кашин	01.12.2017

1. Термины и сокращения

Термин	Определение
Продавец	Участник, принимающий через интернет в свою пользу платежи по банковским картам за товары и услуги через протокол Merchant API
Клиент	Участник, производящий оплату с использованием банковской карты на сайте продавца
Терминал	Точка приема платежей продавца (в общем случае привязывается к сайту, на котором осуществляется прием платежей)
3-D Secure	Протокол, который используется как дополнительный уровень безопасности при осуществлении онлайн-платежей с банковских карт. 3-D Secure добавляет ещё один шаг аутентификации при совершении онлайн-платежей
PCI DSS	Payment Card Industry Data Security Standard, стандарт безопасности данных индустрии платежных карт. Стандарт утверждён международными платежными системами Visa и MasterCard, American Express, JCB и Discover. Стандарт представляет собой совокупность 12 детализированных требований по обеспечению безопасности данных о держателях платёжных карт, которые передаются, хранятся и обрабатываются в информационных инфраструктурах организаций.
Интернет-эквайринг	Технология приёма оплаты в интернете через платёжную страницу банка-эквайера. С помощью интернет-эквайринга покупатели оплачивают покупки на сайтах, в мобильных приложениях, по прямым ссылкам на страницу с платёжной формой банка.
Оплата	Операция с использованием карты и с обязательной авторизацией, проводимой банком-эквайером по поручению владельца карты в магазине в целях покупки товаров или услуг, и по которой был сформирован счёт.
Рекуррентные платежи	<p>Регулярная оплата с банковской карты без подтверждения владельца карты. Банк-эквайер списывает оплату по графику, заранее оговоренному покупателем и магазином. По правилам интервал между двумя оплатами не превышает одного года.</p> <p>Магазин и покупатель заранее договариваются, какие товары или услуги магазин предоставляет покупателю, пока действует соглашение об оплате регулярными платежами. Соглашением о регулярных платежах служит первичная оплата покупателем стандартным способом — с вводом реквизитов карты и проверкой 3-D Secure.</p>

2. Основные положения

Acquiring SDK позволяет интегрировать [Интернет-Эквайринг Tinkoff](#) в мобильные приложения для платформы Android.

Возможности SDK:

- Прием платежей (в том числе рекуррентных);
- Сохранение банковских карт клиента;
- Сканирование и распознавание карт с помощью камеры или NFC;
- Получение информации о клиенте и сохраненных картах;
- Управление сохраненными картами;
- Поддержка английского;
- Кастомизация окна оплаты;
- Интеграция с онлайн-кассами.

1.1 Требования

Для работы Tinkoff Acquiring SDK необходимо:

- Поддержка Android 4.0 и выше (API Level 14);
- Не использовать сторонние библиотеки кроме Card IO и Gson.

1.2 Подготовка к использованию

1.2.1 Подключение

1.2.1.1 Подключение через Gradle

Для подключения SDK добавьте в *build.gradle* вашего проекта следующую зависимость:

```
compile 'ru.tinkoff.acquiring:ui:$latestVersion'
```

1.2.1.2 Подключение через Maven

Для подключения SDK в Maven необходимо добавить в pom.xml:

```
<dependency>
  <groupId>ru.tinkoff.acquiring</groupId>
  <artifactId>ui</artifactId>
  <version>1.3.0</version>
</dependency>
```

1.2.2 Подготовка к работе

Для начала работы с SDK вам понадобятся:

- **Terminal key** - терминал Продавца;
- **Password** - пароль от терминала;
- **Public key** – публичный ключ. Используется для шифрования данных.

Указанные данные можно получить в личном кабинете после подключения к [Интернет-Эквайрингу](#).

Для получения данных необходимо:

1. Выбрать раздел «Терминалы» в дополнительном меню в профиле магазина.

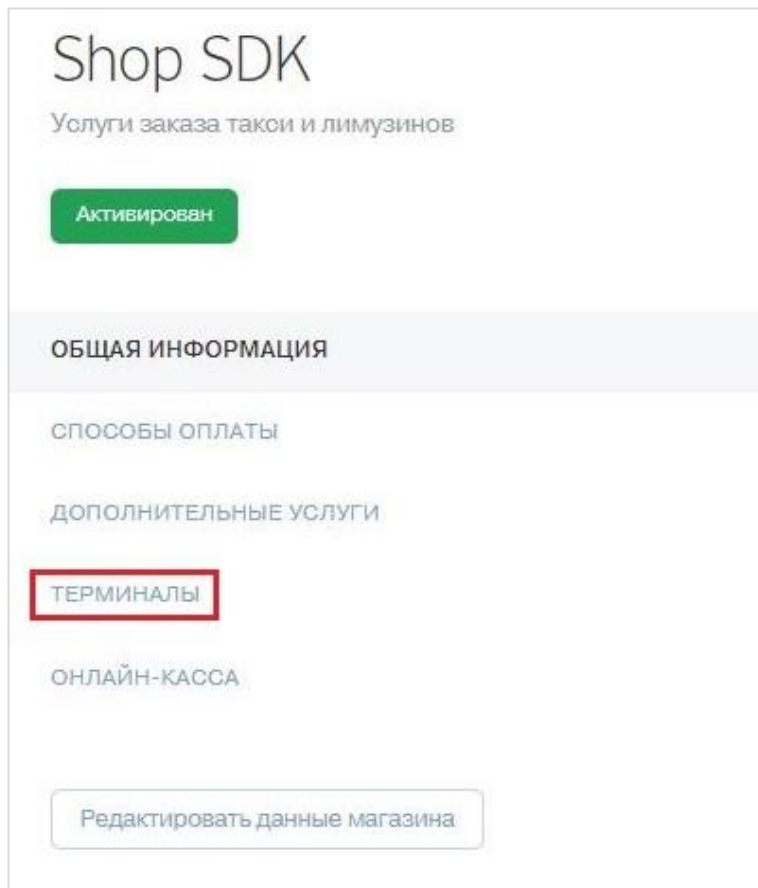


Рисунок 1. Раздел "Терминалы"

2. Выбрать терминал и нажать кнопку **НАСТРОИТЬ**.

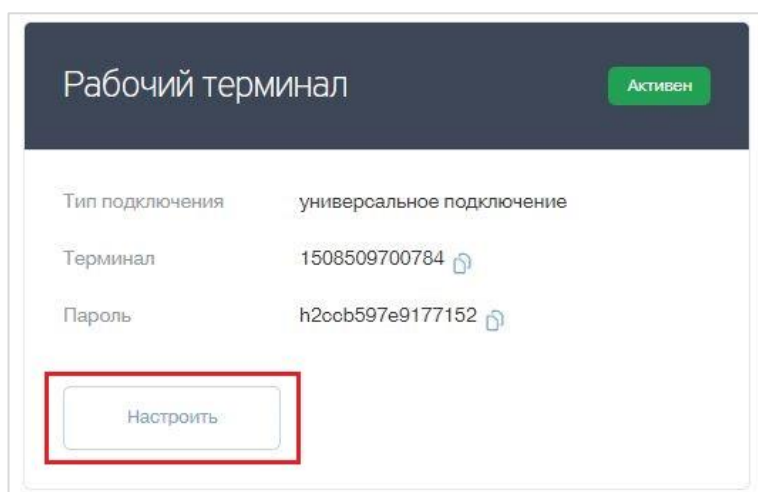


Рисунок 2. Терминал

3. Выбрать вкладку «Мобильное приложение».

Примечание. Если в настройках терминала выбрана не вкладка «Мобильное приложение», а «Универсальное подключение» или «Jivosite», это может привести к сбросу настроек терминала для типа подключения: Мобильный SDK.

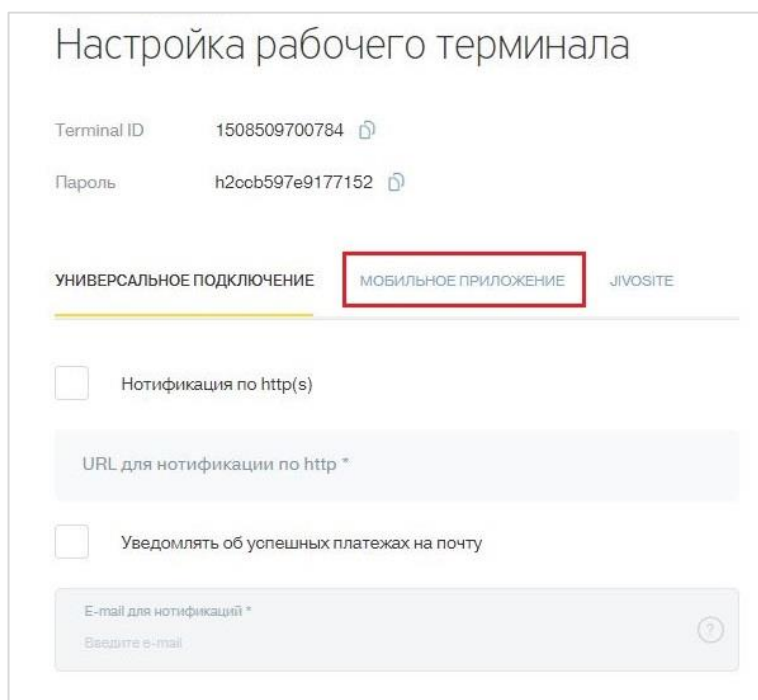




Рисунок 3. Настройка терминала

4. На вкладке указаны:
- **Terminal ID** – Terminal Key;
 - **Пароль** - Password;
 - **Открытый ключ** – Public Key.

Для завершения работы необходимо нажать кнопку **СОХРАНИТЬ**.

Настройка рабочего терминала

Terminal ID 1508509700784 


Пароль h2ocb597e9177152 

УНИВЕРСАЛЬНОЕ ПОДКЛЮЧЕНИЕ **МОБИЛЬНОЕ ПРИЛОЖЕНИЕ** JIVOSITE


☐ Нотификация по http(s)

URL для нотификации по http *

☐ Уведомлять об успешных платежах на почту

Е-mail для нотификаций * 

Введите e-mail

Открытый ключ 


MIIIBjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5y9ka3ZQ
E0feuGtemYv3lqOILck8zHUM7ITr0za6IXtszRSXfUO7jMb+L5C7e2QNFs+
7sIX2OQJ6a+HG8kr+jwJ4tS3cVsWtd9NXpsU40PE4MeNr5RqiNXjcDxA+
L4OsEm/BlyFOEOh2epGyYUd5/iO3OiQFRNicomT2saQYAeqIwWELPs1Xp
Lk9HLx5qPbm8fRrQhjeUD5TLO8b+4yCnObe8vy/BMUwBfq+ieWADljwW
CMp2KTPMGLz48qnaD9kdrYU0iyHqzb2mkDhdIzkm24A3WoYtJCBrB2
xM05sm9+OdCl1f7nPNJbl5URHobSwR94IRGT7CJeUjwWDAQAB 

Рисунок 4. Данные терминала

1.2.3 Выбор режима работы

Дополнительно в конструкторе можно настроить режим работы SDK (debug / prod) с помощью метода `setDebug(Boolean)`.

По умолчанию `debug = true`. При этом в debug режиме осуществляется логгирование запросов/ответов к API и обращение SDK осуществляется к тестовому URL.

2. Проведение оплаты

Для проведения оплаты необходимо запустить *PayFormActivity*.

Активити должна быть настроена на обработку конкретного платежа, поэтому для получения интента для ее запуска необходимо вызвать цепочку из методов **PayFormActivity#init**, **PayFormStarter#prepare** и **PayFormStarter#setCustomerKey**:

```
PayFormActivity
    .init("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY") // данные продавца
    .prepare(
        "ORDER-ID",      // ID заказа в вашей системе
        1000,             // сумма для оплаты
        "НАЗВАНИЕ ПЛАТЕЖА", // название платежа, видимое пользователю
        "ОПИСАНИЕ ПЛАТЕЖА", // описание платежа, видимое пользователю
        "CARD-ID",        // ID карточки
        "batman@gotham.co", // E-mail клиента для отправки уведомления об оплате
        false,             // флаг определяющий является ли платеж рекуррентным 1
        true               // флаг использования безопасной клавиатуры
    )
    .setCustomerKey("CUSTOMER_KEY") // уникальный ID пользователя для сохранения данных
    // его карты
    .startActivityForResult(this, REQUEST_CODE_PAYMENT);
```

Более подробное описание приведено в таблице:

Таблица 1. Описание параметров метода PayFormStarter#prepare

Параметр	Описание
«1000»	Сумма для оплаты, указывается в рублях
«НАЗВАНИЕ ПЛАТЕЖА»	Название платежа, видимое покупателю. Название платежа передается в параметр title и отображается в шапке окна оплаты (ограничение 20 символов)
«ОПИСАНИЕ ПЛАТЕЖА»	Описание платежа, видимое покупателю. Описание платежа передается в параметр Description (ограничение 250 символов)
«CARD-ID»	ID карточки. Для проведения платежа по привязанной карте передается CardId. Данный параметр можно получить после привязки карты через окно AttachCard или в нотификации по платежу

Таблица 2. Описание методов

¹ см. Рекуррентный платеж с привязанной карты.

Метод	Описание
.PayFormStarter#setCustomerKey («CUSTOMER_KEY»)	Метод вызывается в случае, если осуществляется привязка карты к покупателю. (см. Привязка карт) Не является обязательным для вызова.
.startActivityResult(this, REQUEST_CODE_PAYMENT);	Метод запускает активности и возвращает параметр REQUEST_CODE_PAYMENT Параметр RESPONSE_CODE_PAYMENT передает информацию о статусе проведения платежа.

В результате вызова указанной выше цепочки методов отображается окно оплаты.

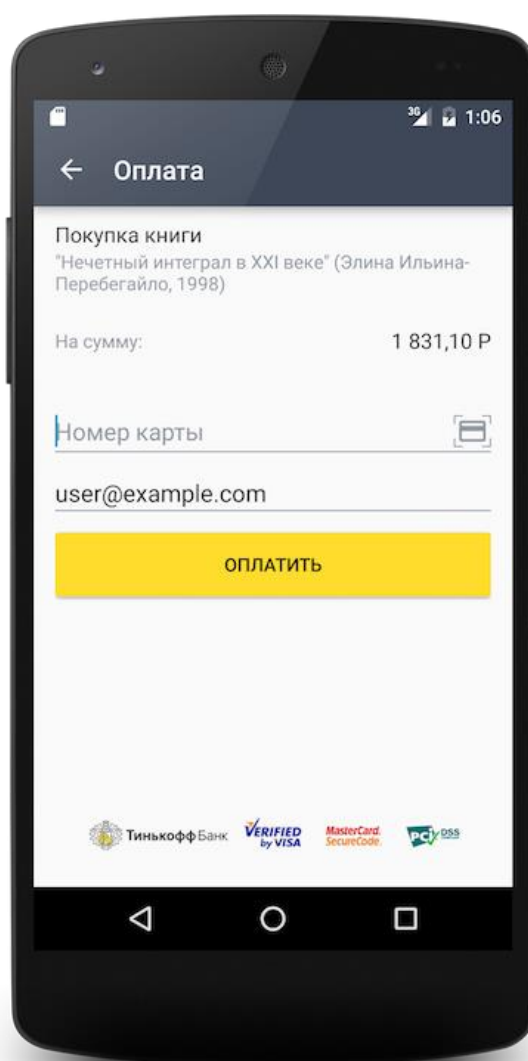
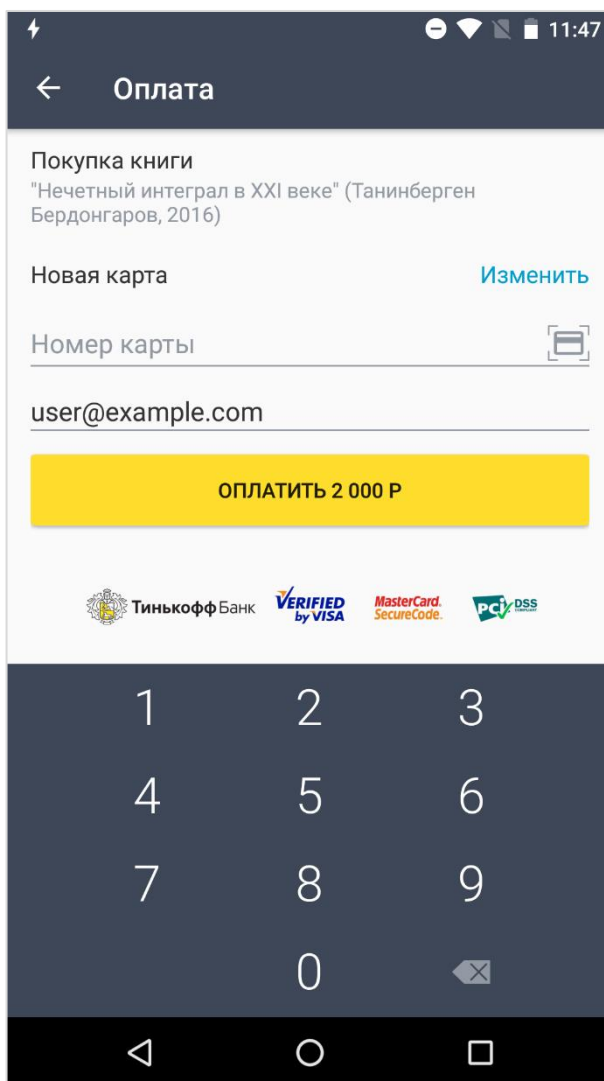


Рисунок 5. Форма оплаты

Форму оплаты можно кастомизировать (см. Настройка стилей).

Безопасная клавиатура

Безопасная клавиатура используется вместо системной и обеспечивает дополнительную безопасность ввода, т.к. сторонние клавиатуры на устройстве клиента могут перехватывать данные и отправлять их злоумышленнику. Безопасную клавиатуру рекомендуется использовать на всех устройствах. При возникновении проблем с использованием безопасной клавиатуры обращайтесь в Поддержка.




The screenshot shows a mobile application interface for a payment screen titled "Оплата" (Payment). The screen displays transaction details for a book purchase, a card selection option, a card number input field, an email address, and a large yellow payment button. At the bottom, there is a numeric keypad (safe keyboard) and a row of logos for Tinkoff Bank, Visa Verified by Visa, MasterCard SecureCode, and PCI DSS. The status bar at the top shows the time as 11:47.

⚡ 11:47

← Оплата





Покупка книги
"Нечетный интеграл в XXI веке" (Танинберген
Бердонгаров, 2016)


Новая карта [Изменить](#)

Номер карты 

user@example.com

ОПЛАТИТЬ 2 000 Р

 Тинькофф Банк   

1 2 3
4 5 6
7 8 9
0 

◀ ○ ◻

Рисунок 6. Безопасная клавиатура

3. Привязка карт

Для запуска привязки карт необходимо запустить *AttachCardFormActivity*.

Активности должна быть настроена на обработку конкретного платежа, поэтому для получения интента для ее запуска необходимо вызвать цепочку из методов *AttachCardFormActivity#init*, *AttachCardFormActivity#prepare*:

```
AttachCardFormActivity
    .init("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY") // данные продавца
    .prepare(
        "CUSTOMER_KEY", // уникальный ID пользователя для сохранения данных его карты
        CheckType.THREE_DS, // тип привязки карты
        true, // флаг использования безопасной клавиатуры
        "E-MAIL") // e-mail клиента
    .startActivityForResult(this, ATTACH_CARD_REQUEST_CODE);
```

В результате запуска активности отображается окно привязки карты. Экран добавления карты должен содержать:

- компонент ввода карты;
- поле ввода email;
- кнопку подтверждения;
- логотипы (TinkoffBank, Verified by Visa, MasterCard SecureCode, PCI DSS Compliant).

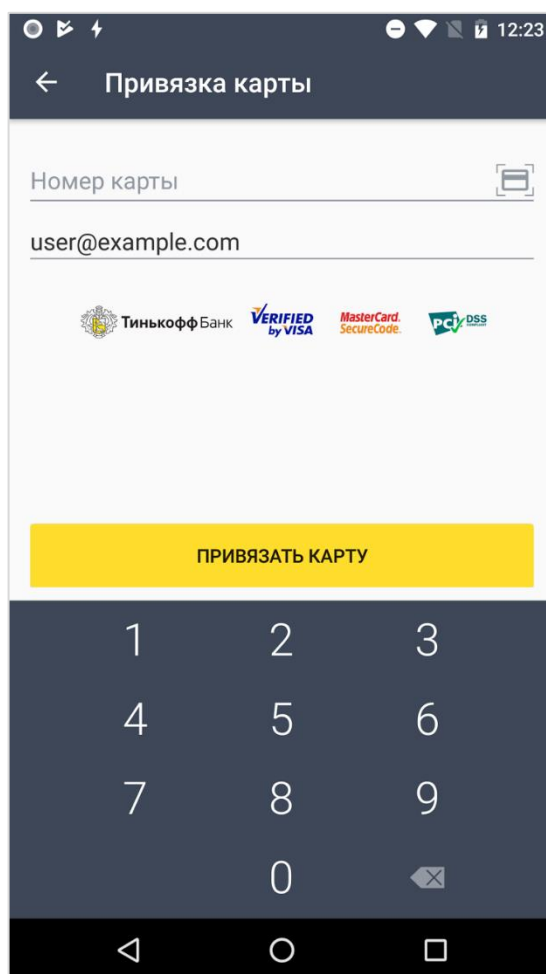


Рисунок 7. Форма привязки карты

Схема работы

1. Клиент передает необходимые параметры для запуска компонента привязки карты:
 - *terminalKey*;
 - *password*;
 - *publicKey*;
 - *customerKey*;
 - *checkType*;
 - параметры кастомизации интерфейса.
2. Пользователь вводит номер карты и нажимает кнопку **ДОБАВИТЬ**.
3. Выполняется запрос AddCard для получения RequestKey.
4. Выполняется запрос AttachCard.
5. Если в ответе присутствует *Status*:
 - *=3DS_CHECKING* - переходим к пункту 6,
 - *=LOOP_CHECKING* переходим к пункту 8,
 - в случае отсутствия поля *Status* и *Success=true* переходим к пункту 10 (иное следует трактовать как ошибку при привязке карты)
6. Отображается экран 3DS подтверждения.
7. После 3DS подтверждения выполняется запрос GetAddCardStatus.
 - Если *Status=COMPLETED* - переходим к пункту 10.
 - Иной результат следует трактовать как ошибку при привязке карты.
8. Показывается экран LOOP подтверждения.
9. Выполняется SubmitRandomAmount с суммой которую, ввел пользователь.
 - Если в ответе *Success=true* - переходим к пункту 10
 - Иной результат следует трактовать как ошибку при привязке карты.
10. Пользователь получает уведомление о успешной привязке.

В результате ввода данных к параметру CustomerKey будет привязана CardId.

По аналогии с PayFormActivity форму привязки карты можно кастомизировать (см. Настройка стилей)

Для кастомизации необходимо вызвать следующие методы:

```
AttachCardFormActivity
.init("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY") // данные продавца
.prepare("CUSTOMER_KEY", CheckType.THREE_DS, true, "E-MAIL")
.setData(data)
.setTheme(themeId)
.startActivityForResult(this, ATTACH_CARD_REQUEST_CODE);
```

4. Рекуррентный платеж с привязанной карты

Рекуррентный платеж нужен для дальнейшего списания средств с сохраненной карты без ввода ее реквизитов. Эта возможность используется, например, для осуществления платежей по подписке.

В метод для старта платежной формы добавляется параметр `charge` (Boolean, default = False). (см. Проведение оплаты).

Параметр передается в API при запросе *Init* в параметре DATA с ключом **chargeFlag** (Если клиент передал DATA, параметр добавляется к уже существующим данным).

Если параметр передан как **True**, форма запускается в режиме совершения рекуррентного платежа:

- В списке карт отображаются карты с `RebillId` (клиентский код может выбрать только карту с заполненным `RebillId`);
- Поле для ввода CVC не отображается;
- Вместо вызова *FinishAuthorize* asdk вызывает `Charge`.

В ответ на запрос `Charge` SDK может получить ответ с `ErrorCode = 104`, содержащий `CardId` (означает, что пользователю необходимо подтвердить платеж через ввод CVC).

В этом случае SDK показывает пользователю платформенный диалог с текстом "Для совершения платежа, введите CVC", при нажатии кнопки **ОК** фокус перемещается в виджет для ввода данных карты в поле для ввода cvc.

После нажатия кнопки **ОПЛАТИТЬ** выполняется запрос `Init`, в DATA, передаются два дополнительных параметра, `recurringType = 12` и `failMapiSessionId = PaymentId` неудачного рекуррента.

5. Интеграция с онлайн-кассами

Для подключения к онлайн-кассам необходимо передать данные чека на форму, указав параметр **Receipt** в метод **PayFormStarter#setReceipt** и кастомизировать форму, передав мапу с параметрами в метод **PayFormStarter#setData**.

URL: <https://securepay.tinkoff.ru/v2/>

Возможно указать тему и запустить форму для оплаты уже с привязанных карт (рекуррентный платеж):

```
PayFormActivity
    .init("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY") // данные продавца
    .prepare(//TODO params)
    .setCustomerKey("CUSTOMER_KEY") // уникальный ID пользователя для сохранения данных
    его карты
    .setReceipt(receipt)
    .setData(dataMap)
    .setTheme(themeId)
    .setChargeMode(chargeMode)
    .startActivityResult(this, REQUEST_CODE_PAYMENT);
```

Данные объекты при их наличии будут переданы на сервер с помощью метода **API Init**, где можно посмотреть детальное описание объекта **Receipt**.

Описание параметра **Receipt**:

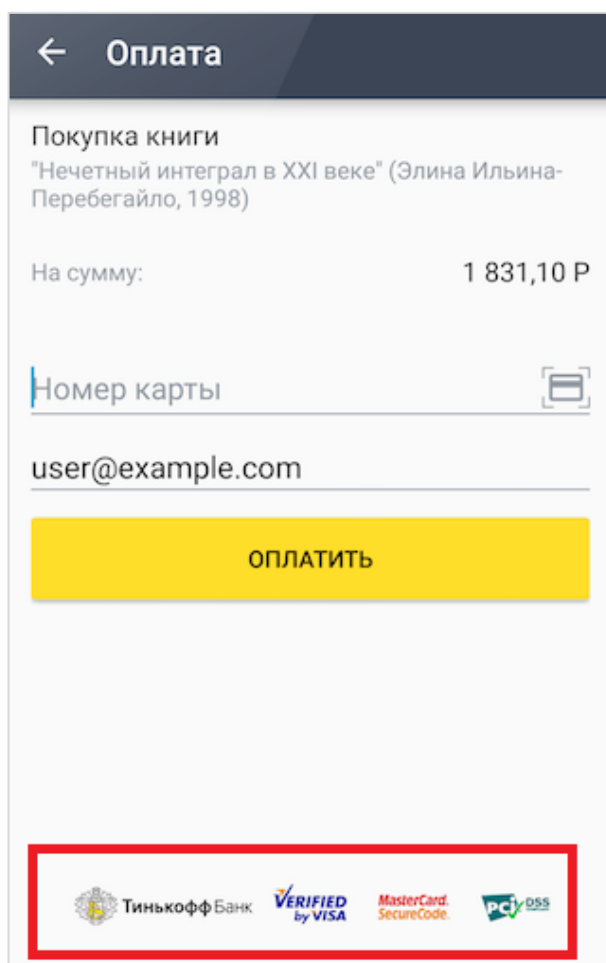
```
import java.io.Serializable;

public class Receipt implements Serializable {
    @SerializedName("Items")
    private Item[] items;
    @SerializedName("Email")
    private String email;
    @SerializedName("Taxation")
    private Taxation taxation;
    @SerializedName("Phone")
    private String phone;
    /**
     * @param items Массив, содержащий в себе информацию о товарах.
     * @param email Емейл.
     * @param taxation Система налогообложения.
     */
    public Receipt(Item[] items, String email, Taxation taxation) {
        this.items = items;
        this.email = email;
        this.taxation = taxation;
    }
    public Item[] getItems() {
        return items;
    }
    public String getEmail() {
        return email;
    }
    public Taxation getTaxation() {
        return taxation;
    }
    public String getPhone() {
        return phone;
    }
    /**
     * @param phone Телефон
     */
    public void setPhone(String phone) {
        this.phone = phone;
    }
}
```

```
}  
}
```

При передаче данных в метод происходит передача данных чека по операции.


6. Настройка стилей



← Оплата

Покупка книги
"Нечетный интеграл в XXI веке" (Элина Ильина-Перебегайло, 1998)

На сумму: 1 831,10 Р

Номер карты 

user@example.com

ОПЛАТИТЬ

Тинькофф Банк VERIFIED by VISA MasterCard SecureCode PCI DSS

Рисунок 8. Настройка формы оплаты

В приложении есть базовая тема `AcquiringTheme`. Для изменения темы необходимо переопределить атрибуты.

На обеих активити используется одна и та же тема, но на `AttachCardFormActivity` не используются некоторые атрибуты.

Есть возможность настройки:

- заголовка и описания формы (title и description) - скрыть/показать;
- поля суммы (amount) - отображение сверху или отображение на кнопке оплатить;
- поля для ввода email - скрыть/показать;
- secure иконок (обведено красным) - возможность скрыть/заменить; расположение внизу экрана или расположение под полями для ввода;
- кнопки оплатить - расположение внизу экрана (под secure иконками) или расположение под полями для ввода (на скриншоте).

Скрытие полей не влияет на отправку параметров. Возможности настройки описаны на github.

<https://github.com/TinkoffCreditSystems/tinkoff-asdk-android/wiki/Styles>

7. Структура

SDK состоит из следующих модулей:

- **Core**
- **UI**
- **Sample**

7.1 Core

Является базовым модулем для работы с Tinkoff Acquiring API.

Он содержит модель данных и позволяет осуществлять запросы к API с разбором ответов из JSON в имеющуюся модель. Запросы осуществляются синхронно. Модуль можно использовать отдельно от остальной SDK для реализации десктопных или веб-приложений.

Класс позволяет конфигурировать SDK и осуществлять взаимодействие с Тинькофф Эквайринг API. Методы, осуществляющие обращение к API, возвращают результат в случае успешного выполнения запроса или бросают исключение `AcquiringSdkException`.

Основной класс модуля - [AcquiringSdk](#) - предоставляет фасад для взаимодействия с Tinkoff Acquiring API. Для работы необходимы ключи и пароль продавца (см. Подготовка к работе).

«Tinkoff acquiring SDK for Android»

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✓	Идентификатор терминала, выдается Продавцу Банком
Amount	Number(10)	✓	Сумма в копейках
OrderId	String(50)	✓	Номер заказа в системе Продавца
Description	String(250)		Краткое описание
Token	String	✓	Подпись запроса
PayForm	String(20)		Название шаблона формы оплаты продавца
CustomerKey	String(36)		Идентификатор покупателя в системе Продавца. Если передается и Банком разрешена автоматическая привязка карт к терминалу, то для данного покупателя будет осуществлена привязка карты
Recurrent	String(1)		Если передается и установлен в Y, то регистрирует платеж как рекуррентный. В этом случае после оплаты в методе GetCardList у карты будет доступен параметр RebillId для использования в методе Charge
PayType (с версии 1.1.0)	String(1)		PayType= "O" - для одностадийного платежа, PayType= "T" - для двухстадийного

URL: <https://securepay.tinkoff.ru/V2/Init>

Метод: GET или POST

Пример запроса:

<https://securepay.tinkoff.ru/V2/Init?TerminalKey=TestB&Amount=100000&OrderId=21050&Description=Подарочная карта на 1000 рублей&DATA=Email=a@test.com>

Пример ответа:

```
{ "Success":true,"ErrorCode":"0","TerminalKey":"TestB","Status":"NEW","PaymentId":"13660","OrderId":"21050","Amount":100000,"PaymentURL":"https://securepay.tinkoff.ru/V2/Authorize/1b63d14a-4208-44a8-a288-ad1b04008e51" }
```

«Tinkoff acquiring SDK for Android»

7.1.2.2 FinishAuthorize

Описание: Подтверждает инициированный платеж передачей карточных данных. Оплата может осуществляться как по полным реквизитам карты (PAN, expiry date, secure code), так и по сохраненной ранее карте (CardId, secure code).

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✓	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number(20)	✓	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
SendEmail	Boolean		Отправлять email с квитанцией или нет. В JSON значение передается в виде строки "false" или "true". Чувствительно к регистру!
InfoEmail	String		Email на который будет отправлена квитанция об оплате
CardData	String	✓	Зашифрованные данные карты в формате "PAN=%pan%;ExpDate=%month% %year%;CVV=%secure_code%" при оплате по полным реквизитам, или "CardId=%id%;CVV=%secure_code%" при оплате с сохраненной карты
Token	String	✓	Подпись запроса

URL: <https://securepay.tinkoff.ru/V2/FinishAuthorize>

Метод: POST

7.1.2.3 Charge

Описание: Осуществляет рекуррентный (повторный) платеж — безакцептное списание денежных средств со счета банковской карты Покупателя.

Для возможности его использования Покупатель должен совершить хотя бы один платеж в пользу Продавца, который должен быть указан как рекуррентный (см. параметр Recurrent в методе Init), фактически являющийся первичным. В дальнейшем при совершении рекуррентного платежа Продавец должен инициировать его, вызвав метод Init, а затем, вызвать метод Charge для оплаты по тем же самым реквизитам, передав параметр RebillId, полученный при совершении первичного платежа.

Для использования рекуррентных платежей необходимо:

1. Совершить родительский платеж путем вызова Init с указанием дополнительного параметра Recurrent=Y.
2. Получить RebillId, предварительно вызвав метод GetCardList.

«Tinkoff acquiring SDK for Android»

3. Спустя некоторое время для совершения рекуррентного платежа необходимо вызвать метод Init со стандартным набором параметров (параметр Recurrent здесь не нужен).
4. Получить в ответ на Init параметр PaymentId.
5. Вызвать метод Charge с параметром RebillId, полученным в п.2 и параметром PaymentId, полученным в п.4.

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✔	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number(20)	✔	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
RebillId	Number(20)	✔	Идентификатор рекуррентного платежа (см. параметр Recurrent в методе Init)
Token	String	✔	Подпись запроса

URL: <https://securepay.tinkoff.ru/V2/Charge>

Метод: POST

Пример ответа:

```
{ "Success":true,"ErrorCode":"0","TerminalKey":"TestB","Status":"CONFIRMED","PaymentId":"63100", "OrderId":"100668","Amount":444 }
```

7.1.2.4 GetState

Описание: Возвращает статус платежа.

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✔	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number(20)	✔	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
Token	String	✔	Подпись запроса

URL: <https://securepay.tinkoff.ru/V2/GetState>

Метод: POST

Пример ответа:

```
{ "Success":true,"ErrorCode":"0","TerminalKey":"TestB","Status":"NEW","PaymentId":"10063","OrderId":"21057" }
```

«Tinkoff acquiring SDK for Android»

7.1.2.5 GetCardList

Описание: Возвращает список привязанных карт.

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✓	Идентификатор терминала, выдается Продавцу Банком.
CustomerKey	String(36)	✓	Идентификатор покупателя в системе Продавца
Token	String	✓	Подпись запроса.

URL: <https://securepay.tinkoff.ru/V2/GetCardList>

Метод: POST

Пример ответа:

```
[{"CardId":"4750","Pan":"543211*****4773","Status":"A"}, {"CardId":"5100","Pan":"411111*****111","Status":"I","RebillId":"12345"}]
```

7.1.2.6 RemoveCard

Описание: Удаляет привязанную карту.

Параметр	Тип	Обязательный	Описание
TerminalKey	String(20)	✓	Идентификатор терминала, выдается Продавцу Банком.
CardId	Number(20)	✓	Идентификатор карты в системе Банка.
CustomerKey	String(36)	✓	Идентификатор покупателя в системе Продавца.
Token	String	✓	Подпись запроса.

URL: <https://securepay.tinkoff.ru/V2/RemoveCard>

Метод: POST

Пример ответа:

```
{"cardId":"4750","Status":"D","Success":true,"ErrorCode":"0","TerminalKey":"TestB","CustomerKey":"Customer1"}
```

7.1.3 AcquiringSdk

Для программиста SDK представлена классом `AcquiringSdk`, выполняющим запросы к API в синхронном режиме. Для создания инстанса класса нужно передать в конструктор необходимые параметры (выдаются мерчанту при подключении к эквайрингу):

- `terminalKey` - имя терминала продавца;

«Tinkoff acquiring SDK for Android»

- password - пароль от терминала;
- publicKey - публичный ключ продавца.

Помимо переданных параметров в конструкторе можно настраивать:

- setLogger(Logger) - позволяет управлять системой, осуществляющей запись логов. По умолчанию, если разработчик не установил кастомный логгер, используется реализация DefaultLogger, основанная на
 - Android - android.util.Log
 - iOS - функция print
 - WP - Debug.WriteLine

При конструировании AcquiringSdk создает инстанс AcquiringApi для отправки запросов в API. Для вызова API нужно вызвать соответствующий метод и передать ему на вход необходимые параметры.

Для каждого класса *Request есть соответствующий класс *RequestBuilder, который содержит:

- методы, принимающие аргументом значения для реквеста;
- метод validate, проверяющий ограничения доменной модели;
- метод makeToken, вычисляющий подпись запроса (см. описание алгоритма ниже);
- метод build, конструирующий инстанс реквеста с подписью, если validate вернул true.

В общем случае вызов Api выглядит так:

1. Создается инстанс класса AcquiringSdk и конфигурируется.
2. На вход соответствующего метода передаются нужные параметры.
3. Внутри метода создается билдер.
4. С помощью билдера собирается объект реквеста. Во время сборки билдер выполняет валидацию переданных параметров и вычисляет token.
5. Собранный билдером реквест передается соответствующему методу AcquiringApi, который вернет респонс или исключение.
6. Респонс из JSON парсится в класс модели *Response.
7. В полученном респонсе проверяется поле Success. В случае успеха возвращается результат, в случае ошибки кидается AcquiringApiException.

7.1.4 CryptoUtils

Содержит набор методов для работы с хешами и криптографией.

- sha256 - принимает на строку для вычисления хеша;
- encryptRsa - принимает на вход данные и шифрует их алгоритмом RSA/ECB/PKCS1Padding;
- encodeBase64 - принимает на вход данные и энкодит их в base64.

7.1.5 Алгоритм формирования подписи запроса (Token)

1. Собираем все параметры запроса Ключ-Значение, кроме параметра Token. Например, [{"TerminalKey", "TestB"}, {"PaymentId", "20150"}].
2. Добавляем туда пару Password-Значение. [{"TerminalKey", "TestB"}, {"PaymentId", "20150"}, {"Password", "123456789"}].
3. Сортируем по ключам. [{"Password", "123456789"}, {"PaymentId", "20150"}, {"TerminalKey", "TestB"}].

«Tinkoff acquiring SDK for Android»

4. Конкатенируем значения. 12345678920150TestB.
5. Вычисляем SHA-256 от пункта 4.

7.1.6 Алгоритм шифрования карточных данных

1. Введенные пользователем номер карты, expiry date и secure code приводятся к виду:
"PAN=%pan%;ExpDate=%month%%year%;CVV=%secure_code%".
2. Выполняется шифрование строк с шага 1 алгоритмом RSA/ECB/PKCS1Padding используя publicKey в качестве ключа.
3. Полученная криптограмма на шаге 2 энкодится алгоритмом Base64.

7.1.7 Пояснения

Одностадийный и двухстадийный платеж

В случае двухстадийного платежа у продавца есть возможность отказаться продавать товар (примеры: товар закончился или страховщик не желает страховать клиента по каким-либо причинам).

7.2 UI

Содержит минимально необходимый UI и функционал для осуществления приема платежей и управления списком карт. Позволяет асинхронно осуществлять запросы к API с помощью модуля core.

Основной класс - **PayFormActivity** - экран с формой оплаты, который позволяет:

- просматривать информацию о платеже;
- вводить или сканировать реквизиты карты для оплаты;
- проходить 3DS подтверждение;
- управлять списком ранее сохраненных карт.

Класс **AttachCardFormActivity** позволяет привязать карту пользователя.

UI часть предоставляет пользователю 3 экрана:

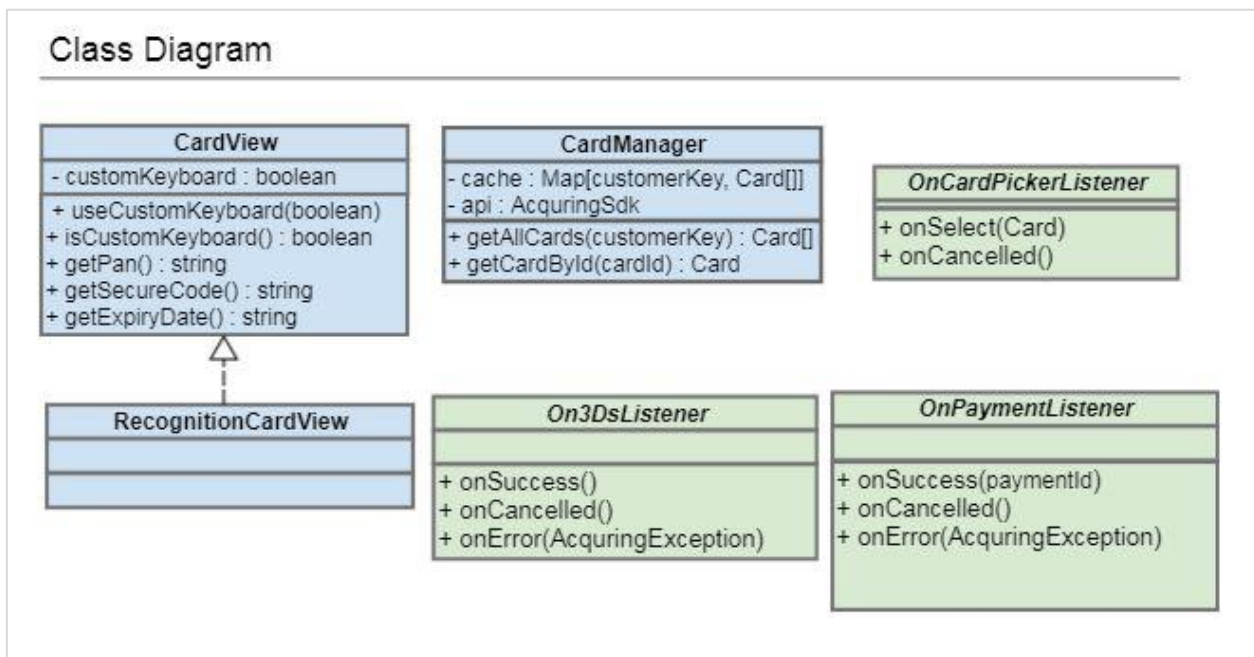
- Ввод реквизитов карты;
- 3DS;
- Список карт.

Прототипы доступны в DropBox по [адресу](#): Mobile/Эквайринг mobile SDK

Все строки, представленные в интерфейсе, имеют две локализации: на русском и на английском в зависимости от языка, выбранного в системе. Существует возможность добавить локализации на нужные языки стандартными средствами платформы (strings.xml - Android, localizable.strings - iOS).

«Tinkoff acquiring SDK for Android»

7.2.1 Классы



7.2.2 Ввод реквизитов карты

7.2.2.1 Параметры, передаваемые на экран

Принимает следующие параметры на вход:

Параметр	Обязательный	Значение по умолчанию	Ограничения	Описание
paymentId	✓			paymentId, полученный после вызова метода Init
amount	✓			Сумма покупки
title		Оплата	String (20)	Заголовок окна
name		null	String	Название товара
description		null	String(250)	Описание товара
cardId		null		Выбранная по умолчанию карта
email		null	Стандартная валидация данных для e-mail	Предзаполненный email покупателя для квитанции

«Tinkoff acquiring SDK for Android»

Параметр	Обязательный	Значение по умолчанию	Ограничения	Описание
customKeyboard		true		Использование кастомной клавиатуры для ввода реквизитов карты

7.2.2.2 При открытии экрана

При открытии экрана в интерфейс подставляются переданные значения

- Заголовок;
- Сумма;
- Описание товара;
- Email.

Если был передан cardId, то с помощью **CardManager#getCardById(cardId)** находится соответствующая карта. В виджет ввода реквизитов карты подставляются значения найденной карты: маскированный номер, **/** на место expiry date. Виджет ожидает ввода secure code.

В зависимости от значения параметра customKeyboard показывается системная или кастомная клавиатура.

7.2.2.3 Логика

При нажатии кнопки **ОПЛАТИТЬ** выполняется асинхронный вызов **AcquiringSdk#finishAuthorize**, на экране на время вызова отображается лоадер.

В случае успешного ответа (success=true) от FinishAuthorize нужно проанализировать поле Status в респонсе:

- если статус 3DS_CHECKING, осуществляется переход на экран 3DS;
- если статус CONFIRMED, экран закрывается, вызывается колбек **OnPaymentListener#onSuccess**;
- во всех остальных случаях колбек **OnPaymentListener#onError**.

Если пользователь выбрал **ОТМЕНИТЬ** - экран закрывается, вызывается колбек **OnPaymentListener#onCancelled**.

7.2.3 3DS

Вызывается в ответ на FinishAuthorize, если терминал продавца поддерживает 3DS (Status=3DS_CHECKING)

7.2.3.1 Параметры, передаваемые на экран

Параметр	Обязательный	Значение по умолчанию	Ограничения	Описание
paymentId	✔			Берется из ответа FinishAuthorize

«Tinkoff acquiring SDK for Android»

Параметр	Обязательный	Значение по умолчанию	Ограничения	Описание
ThreeDsData	✓			Берется из ответа FinishAuthorize

7.2.3.2 При открытии экрана

В webview выполняется POST запрос по адресу ACSUrl с параметрами:

- PaReq - параметр, переданный на экран;
- MD - параметр, переданный на экран;
- TermUrl - AcquiringSdk#getUrl() + "/Submit3DSAuthorization". Константа Submit3DSAuthorization зашивается в экран 3DS.

Отображаем в WebView полученную форму для прохождения 3DS

7.2.3.3 Логика

При срабатывании в WebView редиректа на адрес из TermUrl (см. При открытии экрана) нужно дать сработать запросу, после выполнения - делаем асинхронный вызов AcquiringSdk#getState.

Если:

- Status = CONFIRMED, вызываем колбэк **On3DsListener#onSuccess**;
- Иначе **On3DsListener#onError**.

Если пользователь нажал **НАЗАД** и ушел с экрана без прохождения 3DS, вызываем колбек **On3DsListener#onCancelled**. Также onCancelled вызывается в случае, если была нажата кнопка **ОТМЕНА** в самом WebView. В этом случае будет переход на URL, содержащий в строке адреса cancel.do. При наступлении данного события нужно закрыть WebView и вызвать колбек.

В результате отмены прохождения 3DS осуществляется переход на экран, предшествующий экрану ввода реквизитов карты. Это связано с тем, что при попытке пройти 3DS после вызова FinishAuthorize платеж переводится в статус 3DS_CHECKING и не может быть проведен с использованием реквизитов другой карты, поэтому экран ввода данных карты нужно закрыть, а весь процесс оплаты начинать заново с вызова метода Init.

7.2.4 Сканирование NFC

При выборе опции "Сканировать данные карты по NFC" открывается новый экран с инструкциями "приложите карту к телефону" и кнопкой **ОТМЕНА**. По завершении считывания данных карты экран закрывается, и данные подставляются в форму ввода.

7.2.5 Список карт

7.2.5.1 Параметры, передаваемые на экран

Принимает следующие параметры на вход:

Параметр	Обязательный	Значение по умолчанию	Ограничения	Описание
customerKey	✓		String(36)	Id покупателя

«Tinkoff acquiring SDK for Android»

7.2.5.2 При открытии экрана

Экран асинхронно с помощью **CardManager#getAllCards(customerKey)** загружает все сохраненные карты пользователя и отображает их. CardManager получает список карт от API только при первом запросе и кеширует их в памяти. При следующих запросах, если закрыть и открыть экран заново - данные берутся из кеша, запрос к серверу не выполняется.

На время загрузки карт отображается лоадер.

Если у пользователя нет сохраненных карт - API выкидывает ошибку 7 - *Неверный статус покупателя*. Ее необходимо обрабатывать в SDK и показывать в этом случае заглушку "Нет сохраненных карт".

7.2.5.3 Логика

При выборе карты из списка вызывается колбек **OnCardPickerListener#onSelect(Card)**

При отмене (выход с экрана без выбора карты) вызывается колбек **OnCardPickerListener#onCancelled()**

При удалении карты асинхронно вызывается метод **AcquiringSdk#removeCard(cardId)**. На время выполнения запроса показывается лоадер. После получения от API подтверждения об успешности выполнения запроса - карта удаляется из локального кеша **CardManager#cache** и из списка карт на экране.

7.3 Sample

Демо-приложение, показывающее основной функционал SDK и вариант его использования.

Содержит пример интеграции Tinkoff Acquiring SDK в мобильное приложение по продаже книг.

Демо-приложение эмулирует приложение интернет-магазина, занимающегося продажей книг. Приложение состоит из следующих экранов

- **Основной экран** – экран со списком книг в продаже;
- **Детали** - экран с подробной информацией о товаре;
- **Корзина** - экран с набранными товарами;
- **Подтверждение** - экран об успешности / неуспешности операции покупки;
- **О программе** - информационный экран: версия SDK, EULA и проч.

7.4 ProGuard

При использовании ProGuard необходимо добавить следующую строку:

```
-keep class ru.tinkoff.acquiring.sdk.views.** { *; }
```

8. Маршрут

Схема проведения платежа: <https://oplata.tinkoff.ru/landing/images/scheme.svg>

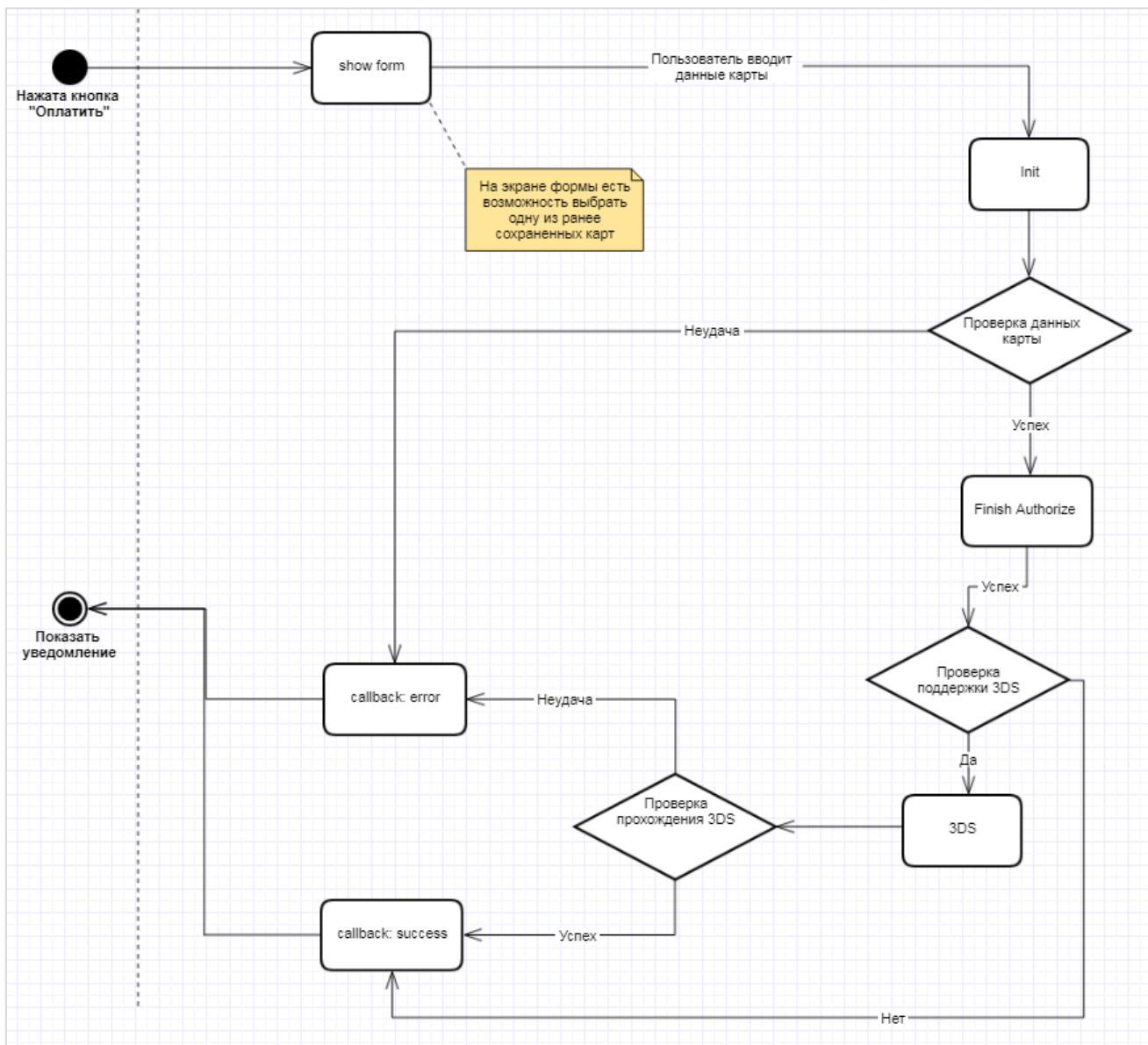


Рисунок 9. Диаграмма состояний платежа

1. При нажатии кнопки **ОПЛАТИТЬ** открывается форма ввода данных карты.
2. Пользователь вводит данные.
3. Запускается метод Init.
4. Совершается проверка данных карты:
 - а) Если проверка прошла успешно – запускается метод FinishAuthorize.
 - б) Если нет – операция отменяется, показывается уведомление об ошибке.
5. Совершается проверка на поддержку терминалом 3-D Secure:
 - а) Если 3-D Secure есть и проверка прошла успешно – операция успешна, отображается уведомление об успехе операции.

«Tinkoff acquiring SDK for Android»

- b) Если 3-D Secure есть и проверка не прошла – операция отменяется (?), отображается уведомление об ошибке.
- c) Если 3-D Secure нет – операция успешна, отображается уведомление об успехе операции.

9. Поддержка

- Просьба по возникающим вопросам обращаться на card_acquiring@tinkoff.ru;
- Баги и feature-реквесты можно направлять в раздел [issues](#);
- Подробное и актуальное описание методов указано в [JavaDoc](#).

«Tinkoff acquiring SDK for Android»

10. Коды ошибок API и возможные исключения

Ошибки должны пробрасываться классом AcquiringSdk как исключения AcquiringApiException. API может возвращать следующие ошибки:

Таблица 3. Ошибки валидации

Код ошибки	Описание
0	Нет ошибки
1	Параметры не сопоставлены
3	Внутренняя ошибка системы интернет эквайринга
4	Запрашиваемое состояние транзакции является неверным
5	Неверный запрос
6	Неверный статус карты
7	Неверный статус покупателя
8	Неверный статус транзакции
9	Переадресовываемый url пуст
10	Метод Charge заблокирован для данного терминала
11	Невозможно выполнить платеж
50	Ошибка отправки нотификации
51	Ошибка отправки Email
52	Ошибка отправки Sms
53	Обратитесь к продавцу
54	Метод вызван повторно
201	Поле {0} должно быть больше или равно {value}
202	Терминал заблокирован
203	Параметры запроса не должны быть пустыми
204	Неверный токен. Проверьте пару TerminalKey/SecretKey
205	Неверный токен. Проверьте пару TerminalKey/SecretKey
206	Email не может быть пустым
207	Параметр {0} превышает максимально допустимый размер
208	Наименование ключа из параметра DATA превышает максимально допустимый размер
209	Значение ключа из параметра DATA превышает максимально допустимый размер

«Tinkoff acquiring SDK for Android»

210	Поле {0} должно быть формата "{regex}"
211	Неверный формат IP
212, 213	Поле {0} должно быть формата "{regex}"
214	Поле {0} числовое значение должно укладываться в формат (<{integer} цифр>.<{fraction} цифр>)
215	Поле {0} должно быть формата "{regex}"
216	Поле {0} должно быть формата "{regex}"
217	Поле {0} должно быть больше или равно {value}
218	Значение {0} не является числовым
219	Неверный срок действия карты
220	Поле {0} должно быть формата "{regex}"
221	Значение {0} не является числовым
222	Поле {0} должно быть больше или равно {value}
223	Поле {0} должно быть больше или равно {value}
224, 225	Неверный формат Email
226-230	Поле {0} должно быть формата "{regex}"
231	Не найден идентификатор карты
233-239	Поле {0} должно быть формата "{regex}"
240, 241	Поле {0} должно быть больше или равно {value}
242	Поле {0} должно быть формата "{regex}"
243	Ошибка шифрования карточных данных
244	Ошибка сопоставления карточных данных
245-250	Параметр {0} не сопоставлен
251	Неверная сумма. Сумма должна быть больше или равна {0} копеек
252	Срок действия карты истек
253	Валюта {0} не разрешена для данного терминала

Таблица 4. Ошибки оплаты

Код ошибки	Описание
99	Воспользуйтесь другой картой, банк выпустивший карту отклонил операцию
101	Не пройдена идентификация 3DS
1006	Проверьте реквизиты или воспользуйтесь другой картой
1012	Воспользуйтесь другой картой
1013	Повторите попытку позже

**«Tinkoff acquiring SDK
for Android»**

1014	Неверно введены реквизиты карты. Проверьте корректность введенных данных
1030	Повторите попытку позже
1033	Проверьте реквизиты или воспользуйтесь другой картой
1034-1043	Воспользуйтесь другой картой, банк выпустивший карту отклонил операцию
1051	Недостаточно средств на карте
1054	Проверьте реквизиты или воспользуйтесь другой картой
1057, 1065	Воспользуйтесь другой картой, банк выпустивший карту отклонил операцию
1082	Проверьте реквизиты или воспользуйтесь другой картой
1089	Воспользуйтесь другой картой, банк выпустивший карту отклонил операцию
1091	Воспользуйтесь другой картой
1096	Повторите попытку позже
9999	Внутренняя ошибка системы