

# TINKOFF ACQUIRING SDK FOR ANDROID



# Содержание

1. ТЕРМИНЫ И СОКРАЩЕНИЯ .....	4
2. ОСНОВНЫЕ ПОЛОЖЕНИЯ .....	5
3. ПОДГОТОВКА К ИСПОЛЬЗОВАНИЮ.....	6
ТРЕБОВАНИЯ И ОГРАНИЧЕНИЯ .....	6
ПОДКЛЮЧЕНИЕ.....	6
ПОДКЛЮЧЕНИЕ ЧЕРЕЗ GRADLE .....	6
ПОДКЛЮЧЕНИЕ ЧЕРЕЗ MAVEN .....	6
ПОДГОТОВКА К РАБОТЕ.....	6
ВЫБОР РЕЖИМА РАБОТЫ.....	9
4. ПРОВЕДЕНИЕ ОПЛАТЫ .....	10
БЕЗОПАСНАЯ КЛАВИАТУРА .....	11
СХЕМА ПРОВЕДЕНИЯ ПЛАТЕЖА .....	13
5. ПОДКЛЮЧЕНИЕ GOOGLE PAY.....	14
6. ИНТЕГРАЦИЯ С ОНЛАЙН-КАССАМИ.....	15
7. ПРИВЯЗКА КАРТ.....	16
8. РЕКУРРЕНТНЫЙ ПЛАТЕЖ С ПРИВЯЗАННОЙ КАРТЫ.....	20
9. НАСТРОЙКА СТИЛЕЙ.....	21
10. СТРУКТУРА.....	22
ASDKCORE.....	22
ASDKUI .....	22
SAMPLE.....	22
МЕТОДЫ.....	22
МЕТОД INIT.....	23
МЕТОД FINISHAUTHORIZE.....	27
МЕТОД CHARGE .....	30
МЕТОД GETSTATE.....	32
МЕТОД ADDCARD.....	33
МЕТОД ATTACHCARD.....	35
МЕТОД GETCARDLIST.....	36
МЕТОД REMOVECARD .....	37
ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ .....	39
ACQURINGSDK.....	39
CRYPTOUTILS .....	39
АЛГОРИТМ ФОРМИРОВАНИЯ ПОДПИСИ ЗАПРОСА (TOKEN) .....	39
АЛГОРИТМ ШИФРОВАНИЯ КАРТОЧНЫХ ДАННЫХ .....	40
11. Коды ошибок API и возможные исключения.....	41
12. Поддержка.....	44

## История изменений

Версия	Описание	Дата
1.0	Первоначальное составление документа	20.10.2017
1.1	Добавлено: <ul style="list-style-type: none"> <li>– Структура SDK;</li> <li>– Описание AttachCardFormActivity;</li> <li>– Описание подключения к онлайн-кассам;</li> <li>– Описание настройки стилей;</li> <li>– Коды ошибок</li> </ul>	23.10.2017
1.2	Изменение структуры и форматирование документа	25.10.2017
1.3	Добавлено: <ul style="list-style-type: none"> <li>– Скриншот безопасной клавиатуры;</li> <li>– Описание подключения через Maven;</li> <li>– Ограничение название платежа;</li> <li>– Скриншот привязки карты.</li> </ul> Дополнена информация по привязке карты	26.10.2017
1.4	Незначительные изменения в структуре документа	31.10.2017
1.5	Добавлены изменения релиза 1.3.1	01.12.2017
1.6	Добавлено описание подключения Google Pay. Изменение формата документа	07.09.2018

# 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 в магазине с целью покупки товаров или услуг, по которой был сформирован счёт; с использованием карты и обязательной авторизацией, проводимой банком-эквайером по поручению владельца карты
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 рекуррентными платежами. Соглашением о рекуррентных 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.

## 3. Подготовка к использованию

### Требования и ограничения

Для работы Tinkoff Acquiring SDK необходима поддержка Android 4.0 и выше (API Level 14).

### Подключение

#### Подключение через Gradle

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

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

#### 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>
```

### 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. Необходим для интеграции вашего приложения с интернет-эквайрингом Тинькофф.

Данные выдаются в личном кабинете после подключения к [Интернет-Эквайрингу](#).

Для получения данных необходимо:

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

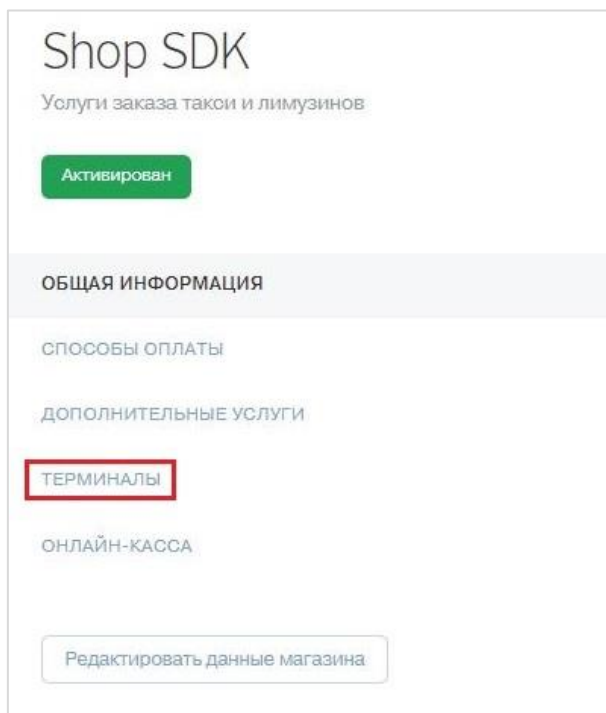


Рисунок 1. Раздел "Терминалы"

2. Выбрать терминал и нажать кнопку [НАСТРОИТЬ](#).

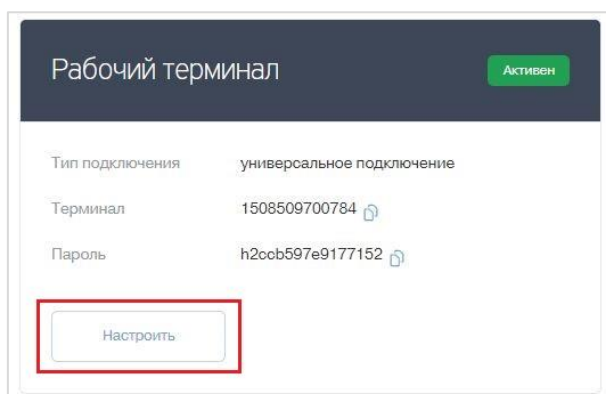


Рисунок 2. Терминал

3. Выбрать вкладку «Мобильное приложение» и заполнить поля.

---

**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.

---

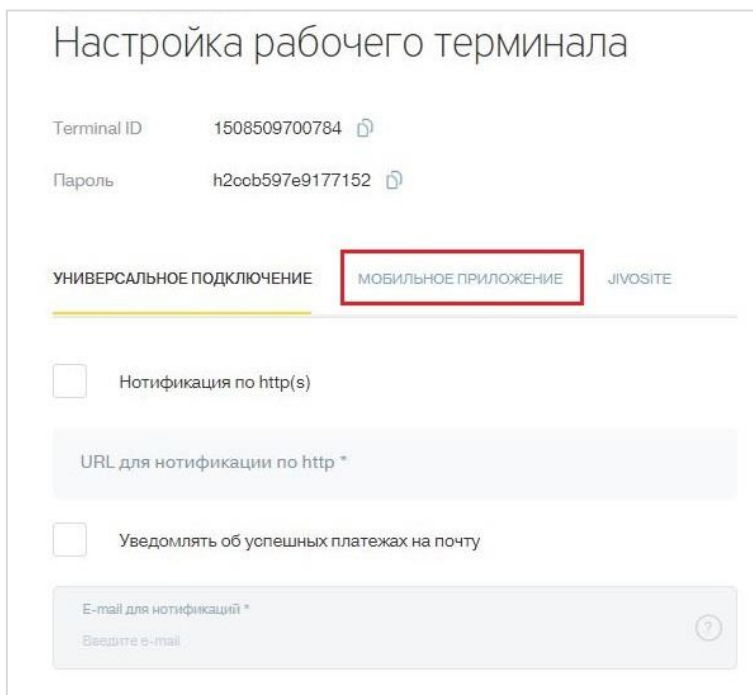


Рисунок 3. Настройка терминала

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

h2cob597e9177152

УНИВЕРСАЛЬНОЕ ПОДКЛЮЧЕНИЕ

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ

JIVOSITE

☐

Нотификация по http(s)

URL для нотификации по http \*

☐

Уведомлять об успешных платежах на почту

E-mail для нотификаций \*

Введите e-mail

Открытый ключ

MIIBlJANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5yse9ka3ZQ  
E0feuGtemYv3lqOILok8zHUM7ITr0za6IXtszRSXfUO7jMb+L5C7e2QNFs+  
7sIX2OQJ6a+HG8kr+jwJ4tS3cVsWtd9NXpsU40PE4MeNr5RqiNXjcDxA+  
L4OsEm/BlyFOEOh2epGyYUd5/iO3OiQFRNicomT2saQYAeqIwuELPs1Xp  
Lk9HLx5qPbm8fRrQhjeUD5TLO8b+4yCnObe8vy/BMUwBfq+ieWADljwW  
CMp2KTPMGLz48qnaD9kdrYJ0iyHqzb2mkDhldzkim24A3MwYitJCBrrB2  
xM05sm9+OdCI1f7nPNJbl5URHobSwR94IRGT7CJcUjwWDAQAB

Отмена

Сохранить

Рисунок 4. Данные терминала

## Выбор режима работы

Дополнительно в конструкторе можно настроить режим работы SDK (debug/prod).

Метод `setDebug(Boolean)` – логирование запросов, метод `setTestDomain` – тестовый URL.

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

## 4. Проведение оплаты

Для проведения оплаты необходимо запустить *PayFormActivity*.

Активности 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

Метод	Описание
<i>.PayFormStarter#setCustomerKey</i> («CUSTOMER_KEY»)	Метод вызывается в случае, если осуществляется привязка карты к покупателю. (см. Привязка карт) Не является обязательным для вызова.

<sup>1</sup> см. Рекуррентный платеж с привязанной карты.

Метод	Описание
<code>.startActivityResult(this, REQUEST_CODE_PAYMENT);</code>	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.

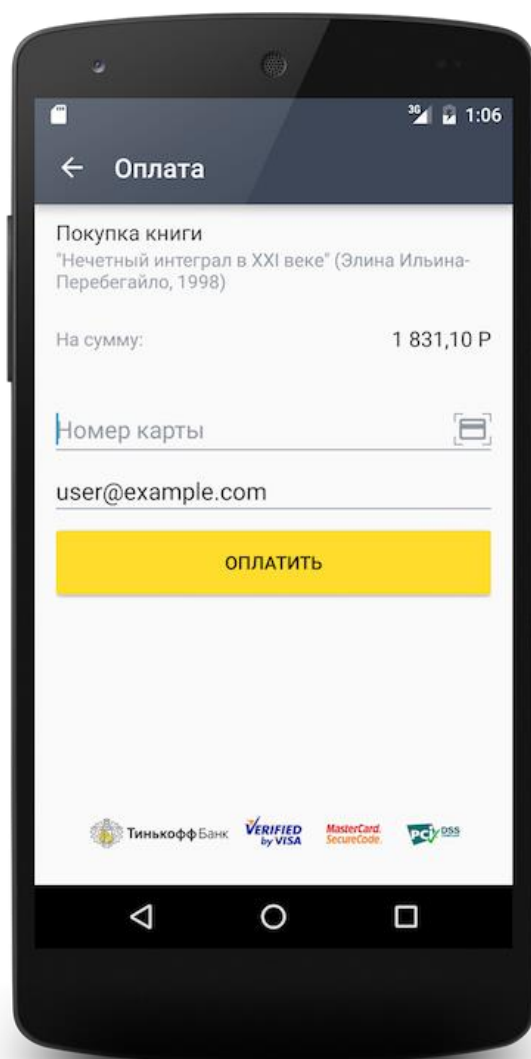
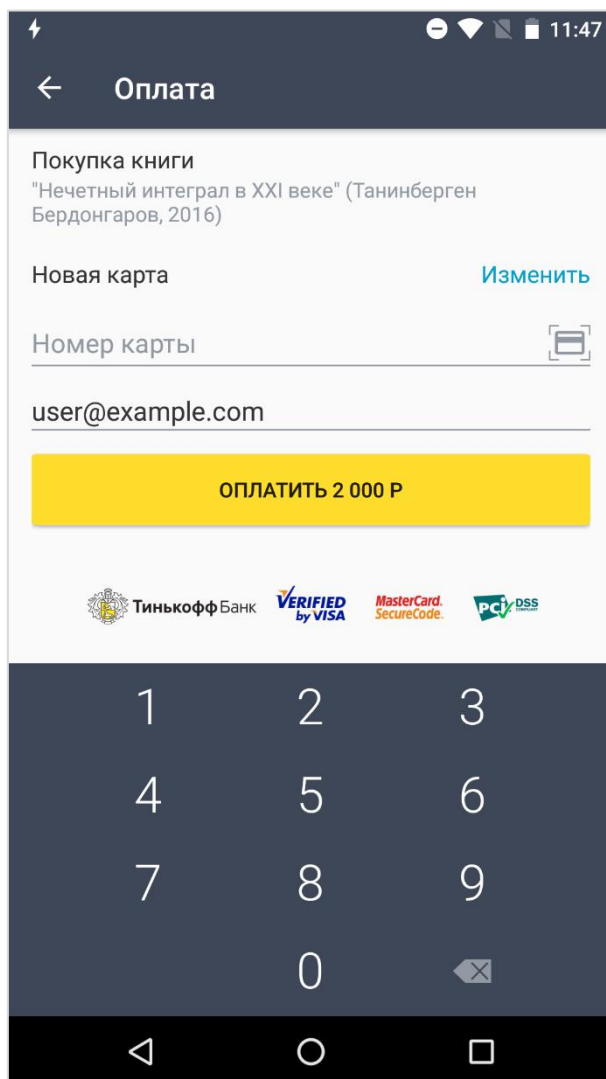


Рисунок 5. Форма оплаты

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](#).




lightning bolt icon, signal strength icon, Wi-Fi icon, battery icon, 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

## Схема проведения платежа

Полная схема: <https://oplata.tinkoff.ru/landing/images/scheme.svg>

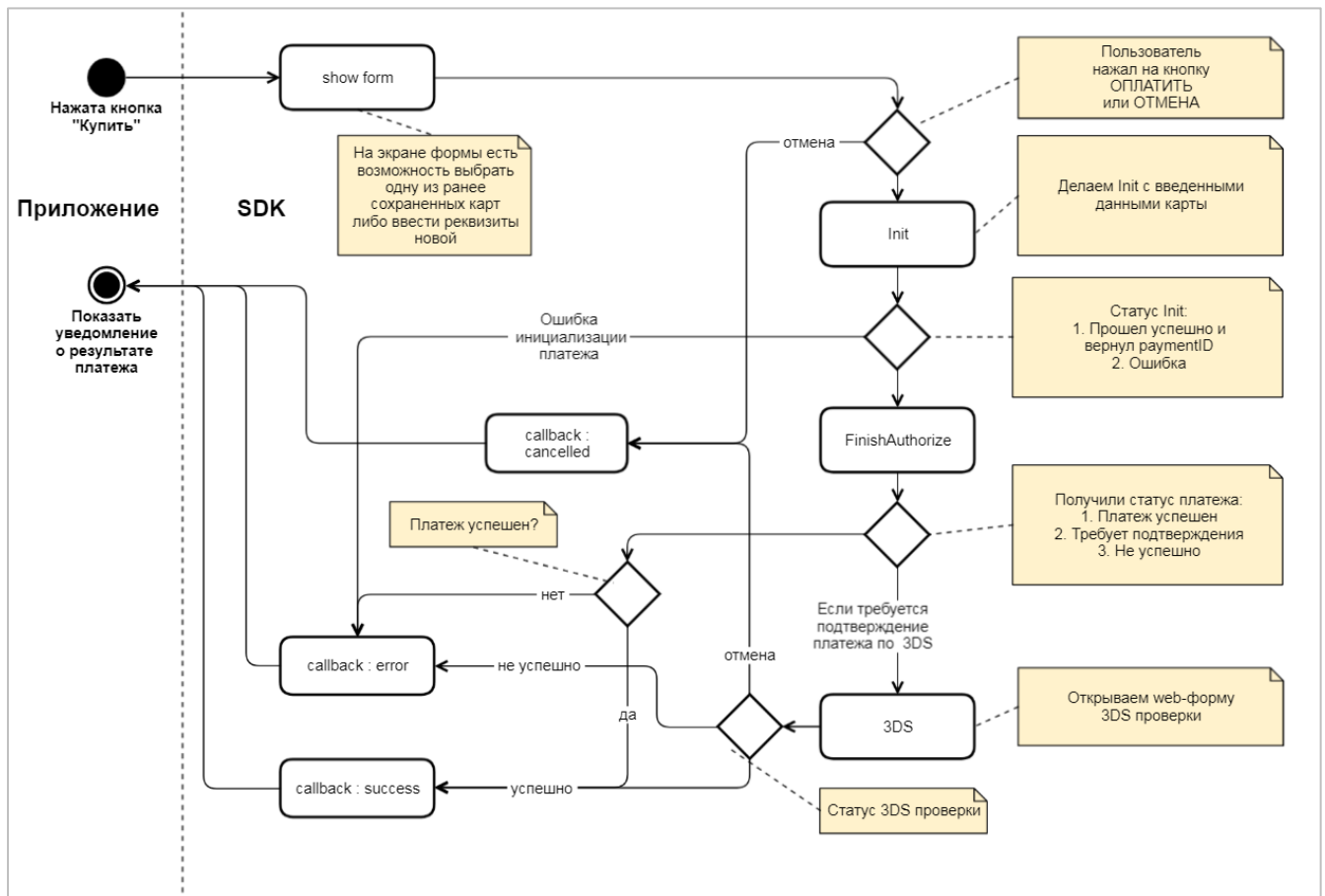


Рисунок 7. Схема проведения платежа

- При нажатии кнопки **КУПИТЬ** открывается форма ввода данных карты.
- Пользователь вводит данные карты либо выбирает одну из сохраненных карт.
  - Если пользователь нажал кнопку оплаты – запускается метод `Init`.
  - Если кнопку отмены – операция отменяется.
- Проверка статуса `Init`:
  - Если `Init` прошел успешно – возвращается `paymentId` и запускается метод `FinishAuthorize`.
  - Если нет – показывается уведомление об ошибке.
- Проверка статуса платежа:
  - Если платеж успешен – операция успешна, отображается уведомление об успехе операции.
  - Если платеж не успешен – отображается уведомление об ошибке.
  - Если платеж требует подтверждения 3-D Secure – открывается форма 3-DS проверки.
  - Если 3-DS проверка прошла успешно – отображается уведомление о результате платежа, если нет – об ошибке.

## 5. Подключение Google Pay

**Внимание!** Для подключения Google Pay необходимо иметь аккаунт на [Google Play Console](#), приложение APK и веб-сайт вашего бизнеса.

Для подключения необходимо:

1. Добавить мета информацию в манифест приложения.

```
<meta-data
    android:name="com.google.android.gms.wallet.api.enabled"
    android:value="true" />
```

2. Сконфигурировать необходимые параметры.

```
GooglePayParams params = new GooglePayParams()
    .setMerchantName(getString(R.string.merchant_name))
    .setAddressRequired(false)
    .setPhoneRequired(false)
    .setTheme(WalletConstants.THEME_LIGHT) // Светлая кнопка
    .setEnvironment(WalletConstants.ENVIRONMENT_TEST) // Для прода значение
ENVIRONMENT_PRODUCTION
    .build();
```

3. Передать параметры в PayFormActivity.

```
PayFormActivity
    .init(TERMINAL_KEY, PASSWORD, PUBLIC_KEY) // данные продавца
    .prepare()
    .setGooglePayParams(params)
    .setCustomerKey(CUSTOMER_KEY)
    .startActivityForResult(this, REQUEST_CODE_PAYMENT);
```

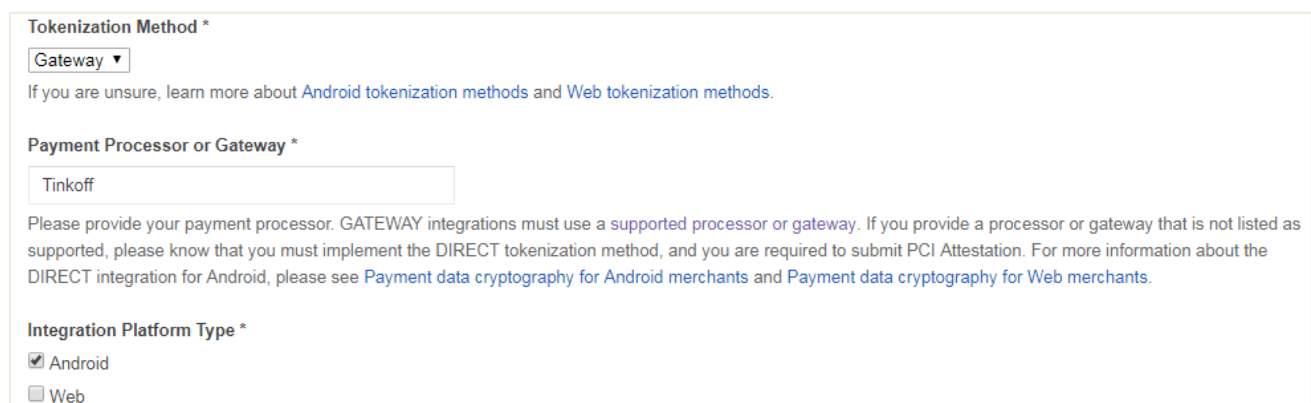
4. Провести тестовый платеж.
5. Написать письмо в Tinkoff на адрес [eacq\\_accounts@tinkoff.ru](mailto:eacq_accounts@tinkoff.ru) с запросом включения оплаты через Google Pay, указав в письме ваш Terminal Key.
6. Проверить дизайн кнопки «Оплатить через Google Pay» на соответствие [brand guidelines](#).

**Примечание.** В пункте 2 в параметре `.setTheme` уже передается корректный дизайн.

7. Отправить запрос в Google для регистрации приложения в Google Pay.

<https://services.google.com/fb/forms/googlepayAPlenable/>

8. В поле формы «Payment Processor or Gateway» указать значение «Tinkoff» и отметить чекбокс «Android» в «Integration Platform Type».



Tokenization Method \*

Gateway ▼

If you are unsure, learn more about [Android tokenization methods](#) and [Web tokenization methods](#).

Payment Processor or Gateway \*

Tinkoff

Please provide your payment processor. GATEWAY integrations must use a [supported processor or gateway](#). If you provide a processor or gateway that is not listed as supported, please know that you must implement the DIRECT tokenization method, and you are required to submit PCI Attestation. For more information about the DIRECT integration for Android, please see [Payment data cryptography for Android merchants](#) and [Payment data cryptography for Web merchants](#).

Integration Platform Type \*

☒ Android

☐ Web

Рисунок 8. Запрос доступа в Google

9. После получения доступа опубликовать приложение в Google Pay.

## 6. 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)
    .startActivityForResult(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.

## 7. 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 платежных систем (TinkoffBank, Verified by Visa, MasterCard SecureCode, PCI DSS Compliant).

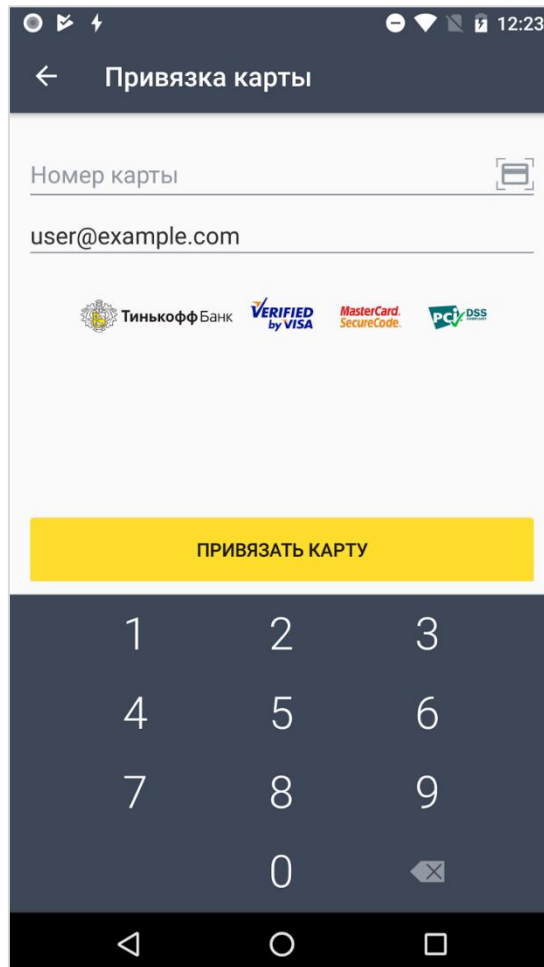


Рисунок 9. Форма привязки карты



## Схема работы

1. Клиент передает необходимые параметры для запуска компонента привязки карты:
  - terminalKey;
  - password;
  - publicKey;
  - customerKey;
  - checkType;
  - параметры кастомизации интерфейса.
  - Пользователь вводит номер карты и нажимает кнопку [ДОБАВИТЬ](#).
2. Выполняется Метод AddCard для получения RequestKey.
3. Выполняется Метод AttachCard для привязки карты.
4. Если в запросе Метод AddCard параметр CheckType=:
  - a) NO – сохранить карту без проверок. Rebill ID для рекуррентных платежей не возвращается.
  - b) HOLD – при сохранении сделать списание и затем отмену на 1 руб. RebillID для рекуррентных платежей возвращается в ответе.
  - c) 3DS – при сохранении карты выполнить проверку 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. В этом случае RebillID будет только для 3DS карт. Карты, не поддерживающие 3DS, привязаны не будут.
  - d) 3DSHOLD – при привязке карты выполнить проверку поддержки картой 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. Если карта не поддерживает 3DS, выполняется списание и последующая отмена на произвольную сумму от 100 до 199 копеек. Клиент будет перенаправлен на страницу для ввода списанной суммы, где должен корректно указать случайную сумму. В случае успешного подтверждения случайной суммы карта будет привязана и возвращен Rebill ID.

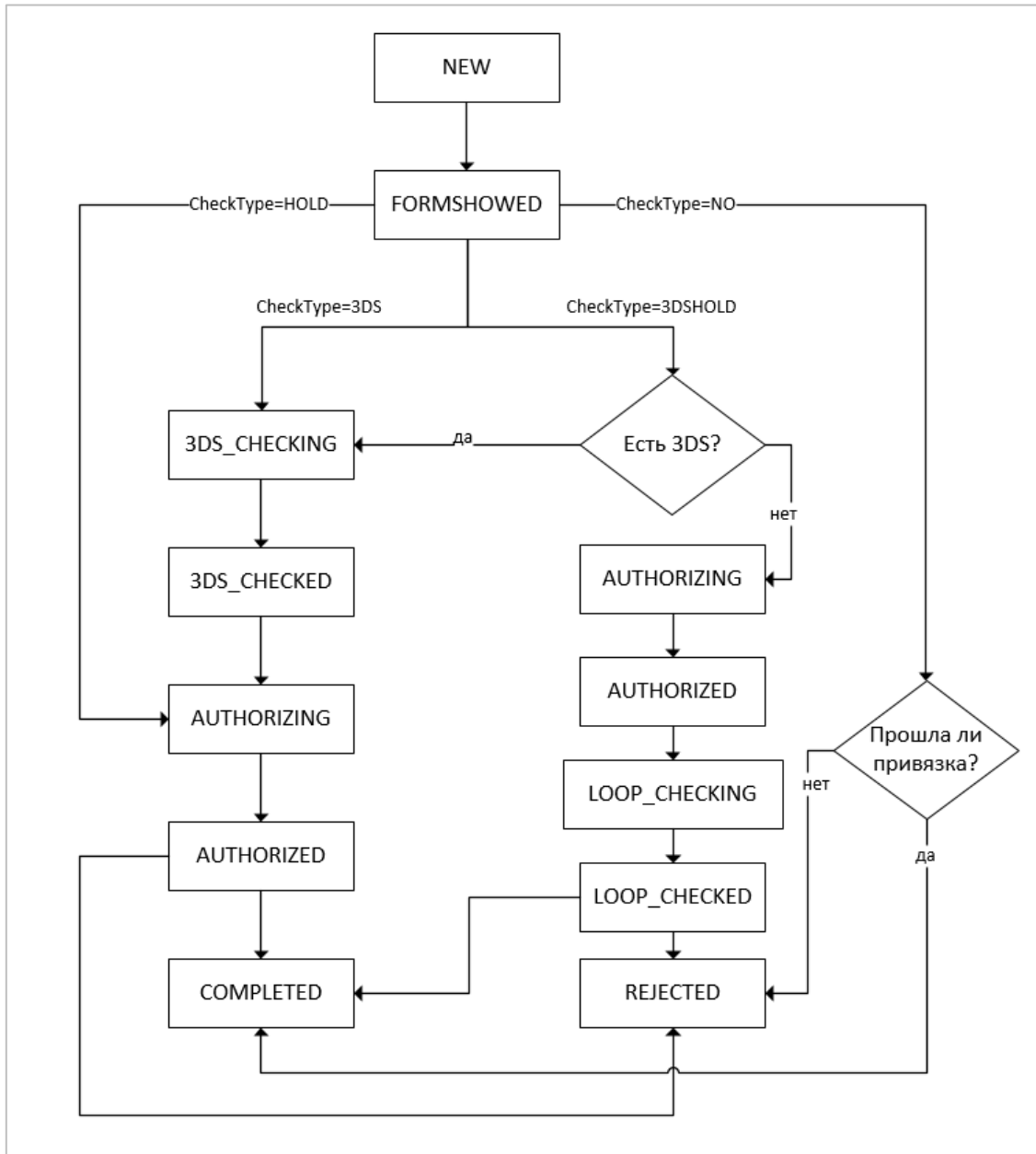


Рисунок 10. Статусная схема привязки карт

Описание статусов:

- NEW — новая сессия;
- FORMSHOWED — показ формы привязки карты;
- 3DS\_CHECKING — отправка пользователя на проверку 3DS
- 3DS\_CHECKED — пользователь успешно прошел проверку 3DS;
- LOOP\_CHECKING — пользователь отправлен на проверку блокирования случайной суммы;
- LOOP\_CHECKED — пользователь успешно прошел проверку блокирования случайной суммы;
- AUTHORIZING — блокировка 1 рубля;
- AUTHORIZED — успешно заблокировали и разблокировали 1 рубль;
- COMPLETED — привязка успешно завершена;
- REJECTED — привязка отклонена.

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);
```

## 8. 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.

## 9. Настройка стилей

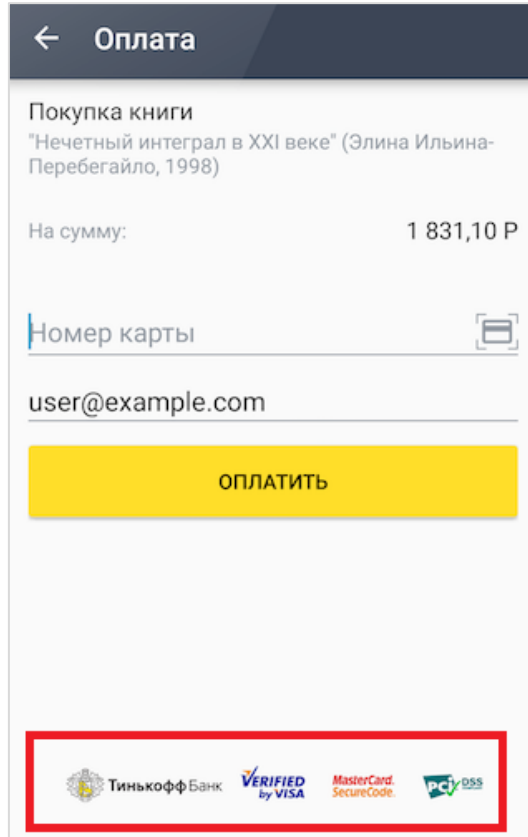


Рисунок 1 1. Настройка формы оплаты

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» или «Оплатить через Google Pay» - 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>

## 10. Структура

- SDK состоит из следующих модулей:ASDKCore;
- ASDKUI;
- Sample.

### ASDKCore

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 отдают 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).

### ASDKUI

Содержит интерфейс, необходимый для приема платежей через мобильное приложение.

Основной класс - `ASDKPaymentFormViewController` - экран с формой оплаты, который позволяет:

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

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).

### Sample

Содержит пример интеграции Tinkoff Acquiring SDK в мобильное приложение по продаже книг. Основные классы и файлы `ASDKTestKeys` содержат *Terminal key*, *Пароль*, *Public key* и *PayController* фасад для `ASDKAcquiringSdk`, который создает экземпляр `ASDKAcquiringSdk` и предоставляет функционал для оплаты.

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.

### Методы

`AcquiringApi` позволяет выполнять синхронные запросы к API. В качестве параметра на вход принимает соответствующий `Request`. В ответ возвращает `Response`, распарсенный из JSON ответа сервера.

Поддерживаемые методы:

- Метод Init;
- Метод FinishAuthorize;
- Метод Charge;
- Метод GetState;
- Метод AddCard;
- Метод AttachCard;
- Метод GetCardList;
- Метод RemoveCard.

## Метод Init

Описание: Иницирует новый платеж.

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

Метод: GET или POST

Таблица 3. Параметры запроса

Параметр	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
IP	String	Нет	IP-адрес клиента
Description	String	Нет	Краткое описание
Currency	Number	Нет	Код валюты ISO 4217 (например, 643). Если передан Currency, и он разрешен для Продавца, то транзакция будет инициирована в переданной валюте. Иначе будет использована валюта по умолчанию для данного терминала
Token	String	Да	Подпись запроса
PayForm	String	Нет	Название шаблона формы оплаты продавца
CustomerKey	String	Нет	Идентификатор покупателя в системе Продавца. Если передается и Банком разрешена автоматическая привязка карт к терминалу, то для данного покупателя будет осуществлена привязка карты. Тогда в нотификации на AUTHORIZED будет передан параметр CardId, подробнее см. метод <i>GetCardList</i>

Параметр	Тип	Обязательный	Описание
Language	String	Нет	Язык платёжной формы. ru - форма оплаты на русском языке; en - форма оплаты на английском языке. По умолчанию (если параметр не передан) - форма оплаты на русском языке.
Recurrent	String	Нет	Если передается и установлен в Y, то регистрирует платеж как рекуррентный. В этом случае после оплаты в нотификации на AUTHORIZED будет передан параметр RebillId для использования в методе <i>Charge</i>
RedirectDueDate	Datetime	Нет	Срок жизни ссылки. В случае, если текущая дата превышает дату переданную в данном параметре, ссылка для оплаты становится недоступной и платёж выполнить нельзя. Формат даты: YYYY-MM-DDTHH24:MI:SS+GMT Пример даты: 2016-08-31T12:28:00+03:00
DATA*	Object	Нет	JSON объект содержащий дополнительные параметры в виде «ключ»:«значение». Данные параметры будут переданы на страницу оплаты (в случае ее кастомизации). Максимальная длина для каждого передаваемого параметра: Ключ – 20 знаков, Значение – 100 знаков. Максимальное количество пар «ключ-значение» не может превышать 20(*)
Receipt	Object	Нет	JSON объект с данными чека

(\*) В случае если у терминала включена опция привязки покупателя после успешной оплаты и передается параметр CustomerKey, то в передаваемых параметрах DATA могут присутствовать параметры команды AddCustomer. Если они присутствуют, они автоматически привязываются к покупателю.

Например, если указать:

`"DATA":{"Phone":"+71234567890", "Email":"a@test.com"}`

к покупателю автоматически будут привязаны данные Email и телефон, и они будут возвращаться при вызове метода *GetCustomer*.

Таблица 4. Структура объекта Receipt

Наименование	Тип	Обязательность	Описание
Items	Массив объектов	Да	Массив, содержащий в себе информацию о товарах



Наименование	Тип	Обязательность	Описание
Email	String	Да	Электронная почта
Phone	String	Нет	Телефон
Taxation	Перечисление (Enum)	Да	Система налогообложения. Перечисление со значениями: - «osn» – общая СН; - «usn_income» – упрощенная СН (доходы); - «usn_income_outcome» – упрощенная СН (доходы минус расходы); - «envd» – единый налог на вмененный доход; - «esn» – единый сельскохозяйственный налог; - «patent» – патентная СН.

Таблица 5. Структура объекта Items

Наименование	Тип	Обязательность	Описание
Name	String	Да	Наименование товара. Максимальная длина строки – 128 символов
Price	Number	Да	Цена в копейках Целочисленное значение не более 10 знаков
Quantity	Number	Да	Количество/вес: - целая часть не более 8 знаков; - дробная часть не более 3 знаков
Amount	Number	Да	Сумма в копейках. Целочисленное значение не более 10 знаков
Tax	Перечисление (Enum)	Да	Ставка налога. Перечисление со значениями: - «none» – без НДС; - «vat0» – НДС по ставке 0%; - «vat10» – НДС чека по ставке 10%; - «vat18» – НДС чека по ставке 18%;

Наименование	Тип	Обязательность	Описание
			- «vat110» – НДС чека по расчетной ставке 10/110; - «vat118» – НДС чека по расчетной ставке 18/118
Ean13	String	Нет	Штрих-код
ShopCode	String	Нет	Код магазина. Для параметра ShopCode необходимо использовать значение параметра Submerchant_ID, полученного в ответ при регистрации магазинов через xml. Если xml не используется, передавать поле не

**Пример запроса:**

```
{
  "TerminalKey": "TestB",
  "Amount": "140000",
  "OrderId": "21050",
  "Description": "Подарочная карта на 1000 рублей",
  "Token": "2ED30E046136931431B5251B7C9A1EAC68DAB082203BD42676BA14A851359DF4"
  "DATA": { "Phone": "+71234567890", "Email": "a@test.com" },
  "Receipt": {
    "Email": "a@test.ru",
    "Phone": "+79031234567",
    "Taxation": "osn",
    "Items": [
      {
        "Name": "Наименование товара 1",
        "Price": 10000,
        "Quantity": 1.00,
        "Amount": 10000,
        "Tax": "vat10",
        "Ean13": "0123456789",
        "ShopCode": "12345"
      },
      {
        "Name": "Наименование товара 2",
        "Price": 20000,
        "Quantity": 2.00,
        "Amount": 40000,
        "Tax": "vat18"
      },
      {
        "Name": "Наименование товара 3",
        "Price": 30000,
        "Quantity": 3.00,
        "Amount": 90000,
        "Tax": "vat10"
      }
    ]
  }
}
```

Таблица 6. Параметры ответа

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
Success	bool	Да	Успешность операции
Status	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «О» - если успешно
PaymentURL	String	Нет	Ссылка на страницу оплаты. Не передается, если ввод данных карты осуществляется на стороне Продавца. <i>Доступна в течении 24 часов по умолчанию.</i>
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

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

Статус платежа:

при успешном сценарии: NEW;

при неуспешном: REJECTED.

## Метод FinishAuthorize

Описание: Подтверждает инициированный платеж передачей карточных данных.

При использовании одностадийного проведения осуществляет списание денежных средств с карты покупателя.

При двухстадийном проведении осуществляет блокировку указанной суммы на карте покупателя.

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

Метод: POST

Таблица 7. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
CardData(*)	String	Да	Зашифрованные данные карты в формате: "PAN=%pan%;ExpDate=%month%%year%;CVV=%secure_code%" при оплате по полным реквизитам; "CardId=%id%;CVV=%secure code%" при оплате с сохраненной карты
DATA	Object	Нет	JSON объект содержащий дополнительные параметры в виде "ключ": "значение". Данные параметры будут переданы на страницу оплаты (в случае ее кастомизации). Максимальная длина для каждого передаваемого параметра: Ключ – 20 знаков, Значение – 100 знаков. Максимальное количество пар «ключ-значение» не может превышать 20
IP	String	Нет	IP-адрес клиента
Phone	String	Нет	Телефон клиента
SendEmail	boolean	Нет	true – отправлять клиенту информацию на почту об оплате; false – не отправлять
InfoEmail	String	Нет	Email для отправки информации об оплате
Token	String	Да	Подпись запроса

Продавец собирает поле CardData в виде списка «ключ=значение» с разделителем «;» и зашифровывает его выданным при регистрации терминала открытым ключом (X509 RSA 2048).

Таблица 8. Параметры CardData

Наименование	Тип	Обязательный?	Описание
PAN	Number	Да	Номер карты
ExpDate	Number	Да	Месяц и год срока действия карты в формате ММYY
CardHolder	String	Нет	Имя и фамилия держателя карты (как на карте)

Наименование	Тип	Обязательный?	Описание
CVV	String	Да	Код защиты (с обратной стороны карты)

Пример значения элемента формы CardData:

“PAN=4300000000000777;ExpDate=0519;CardHolder=IVAN PETROV;CVV=111”

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "PaymentId": "10063"
  "CardData": "b3tSlUYwsf3Erdv5ReB7WpWK3/NBWLlWdiSLjQG0cBxA0Mgs7ALd7edi0RbVlORsyGZEUJS1RynQ9z
  LMyHYzWP3z2sQYGA VzOqufoVPe2AozhW3pZV+dN5s7oGcpXd39NDC0Ma/Zw6oa3dJR0Zh8QYjv/sG0zU1lMjX15aHg
  Tpxk37q6OxUakxuG7euhvSN71JqxHsNEuoJELAQlq7U+3tuh9AjTuiBpmEH99maK9e7gnVXgZd1Nk8vachs97xj9cL
  /023qYmk7CMjldBfG4VOsYVqcHsKfbbJJ8CZXiJgmXhCYns1hmRD/kf3OhEZr038LghC7Iio0yxHYMhZyJoQ=="
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 9. Параметры ответа

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
OrderId	String	Да	Номер заказа в системе Продавца
Amount	Number	Да	Сумма в копейках
RebillId	String	Нет	Идентификатор рекуррентного платежа
CardId	String	Нет	Идентификатор карты в системе Банка. Передается только для сохраненной карты
Success	bool	Да	Успешность операции (true/false)
Status(*)	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{ "Success":true, "ErrorCode":"0", "TerminalKey":"TestB", "Status":"CONFIRMED", "PaymentId":"10063", "OrderId":"21050", "Amount":100000 }
```

(\*)Статус платежа:

при успешном сценарии и одностадийном проведении платежа: **CONFIRMED**

при успешном сценарии и двухстадийном проведении платежа: **AUTHORIZED**

при неуспешном: **REJECTED**

при необходимости прохождения проверки 3-D Secure: **3DS\_CHECKING**

## Метод 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. По завершении оплаты такого платежа в нотификации на **AUTHORIZED** будет передан параметр **RebillId**. В дальнейшем при совершении рекуррентного платежа Продавец должен инициировать его, вызвав метод **Init**, а затем, не осуществляя переадресации на **PaymentURL**, вызвать метод **Charge** для оплаты по тем же самым реквизитам передав параметр **RebillId**, полученный при совершении первичного платежа. Независимо от установленного типа проведения платежа, метод **Charge** всегда работает по типу одностадийного проведения. Это значит, что во время выполнения метода **Charge** на **Notification URL** будет отправлен синхронный запрос (подробнее см. Нотификация продавца об операциях), на который требуется корректный ответ.

Другими словами, для использования рекуррентных платежей необходима следующая последовательность действий:

1. Совершить родительский платеж путем вызова **Init** с указанием дополнительного параметра **Recurrent=Y**.
2. Переадресовать Покупателя на **PaymentUrl**.
3. После оплаты покупателем в нотификации на **AUTHORIZED** будет передан параметр **RebillId**, который необходимо сохранить.
4. Спустя некоторое время для совершения рекуррентного платежа необходимо вызвать метод **Init** со стандартным набором параметров (параметр **Recurrent** здесь не нужен).
5. Получить в ответ на **Init** параметр **PaymentId**, при этом переадресацию пользователя на **PaymentUrl** производить не надо.
6. Вызвать метод **Charge** с параметром **RebillId** полученным в п.3 и параметром **PaymentId** полученным в п.5.

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

Метод: POST

Таблица 10. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода <b>Init</b>

Наименование	Тип	Обязательный?	Описание
IP	String	Нет	IP-адрес клиента
RebillId	Number	Да	Идентификатор рекуррентного платежа (см. параметр Recurrent в методе Init)
Token	String	Да	Подпись запроса.
SendEmail	bool	Нет	true – если покупатель хочет получать уведомления на почту
InfoEmail	String	Нет	Адрес почты покупателя

Пример запроса:

```
{
  "TerminalKey": "TestB",
  "PaymentId": "10063",
  "RebillId": "145919",
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 11. Параметры ответа

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
OrderId	String	Да	Номер заказа в системе Продавца
Success	bool	Да	Успешность операции (true/false)
Status	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «О» - если успешно
Amount	Number	Да	Сумма списания в копейках
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Наименование	Тип	Обязательный?	Описание
CardId	String	Нет	Идентификатор карты в системе Банка
RebillId	String	Нет	Идентификатор рекуррентного платежа

Пример ответа:

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

## Метод GetState

Описание: Возвращает статус платежа

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

Метод: POST

Таблица 12. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
IP	String	Нет	IP-адрес клиента
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey":"TestB",
  "PaymentId":"10063",
  "Token":"871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 13. Параметры ответа

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
OrderId	String	Да	Номер заказа в системе Продавца
Success	bool	Да	Успешность операции (true/false)



Наименование	Тип	Обязательный?	Описание
Status	String	Да	Статус транзакции
Amount	Number	Нет	Сумма отмены в копейках
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «О» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

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

Таблица 14. Возможные статусы транзакции

Статус	Промежуточный?	Значение
AUTHORIZED	Нет	Средства заблокированы, но не списаны
3DS_CHECKING	Нет	Покупатель начал аутентификацию по протоколу 3-D Secure
CONFIRMED	Нет	Денежные средства списаны

## Метод AddCard

**Описание:** Иницирует привязку карты к покупателю.

В случае успешной привязки переадресует клиента на Success Add Card URL, в противном случае на Fail Add Card URL. Можно использовать форму банка, возможно заменить на кастомную форму.

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

Метод: POST

Таблица 15. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
CheckType	String	Нет	Возможные значения: NO – сохранить карту без проверок. Rebill ID для рекуррентных платежей не возвращается.

Наименование	Тип	Обязательный?	Описание
			<p>HOLD – при сохранении сделать списание и затем отмену на 1 руб. RebillID для рекуррентных платежей возвращается в ответе.</p> <p>3DS – при сохранении карты выполнить проверку 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. В этом случае RebillID будет только для 3DS карт. Карты, не поддерживающие 3DS, привязаны не будут.</p> <p>3DSHOLD – при привязке карты выполнить проверку поддержки картой 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. Если карта не поддерживает 3DS, выполняется списание и последующая отмена на произвольную сумму от 100 до 199 копеек. Клиент будет перенаправлен на страницу для ввода списанной суммы, где должен корректно указать случайную сумму. В случае успешного подтверждения случайной суммы карта будет привязана и возвращен Rebill ID</p>
Description	String	Нет	Описание/название карты
PayForm	String	Нет	Название шаблона формы привязки
IP	String	Нет	IP-адрес запроса
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "1201253242594",
  "CustomerKey": "Test-112",
  "Token": "2ED30E046136931431B5251B7C9A1EAC68DAB082203BD42676BA14A851359DF4"
}
```

Формат ответа: JSON

Таблица 16. Параметры ответа

Имя	Тип	Обязательный?	Описание
TerminalKey	String	Да	Платежный ключ, выдается Продавцу при заведении терминала
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
RequestKey	String	Да	Идентификатор запроса на привязку карты
PaymentURL	String	Нет	Ссылка на страницу привязки карты. На данную страницу необходимо переадресовать клиента для привязки карты
Success	bool	Да	Успешность операции
ErrorCode	String	Да	Код ошибки, «0» если успешно

Имя	Тип	Обязательный?	Описание
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success": true,
  "ErrorCode": "0",
  "TerminalKey": "1485466639730",
  "CustomerKey": "906540",
  "PaymentURL": "https://rest-api-test.tinkoff.ru/AddCard/82a31a62-6067-4ad8-b379-04bfl3e37642",
  "RequestKey": "ed989549-d3be-4758-95c7-22647e03f9ec"
}
```

## Метод AttachCard

**Описание:** Завершает привязку карты к покупателю. Метод вызывается автоматически после метода AddCard.

В случае успешной привязки переадресует клиента на Success Add Card URL в противном случае на Fail Add Card URL.

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

Метод: POST

Таблица 17. Параметры запроса

Имя	Тип	Обязательный?	Описание
TerminalKey	String	Да	Платежный ключ, выдается Продавцу при заведении терминала
RequestKey	String	Да	Идентификатор запроса на привязку карты
CardData	String	Да	Зашифрованные данные карты в формате: "PAN=%pan%;ExpDate=%month%%year%;CVV=%secure_code%"
DATA	Object	Нет	Ключ = значение дополнительных параметров через " ", например, Email = <a href="#">a@test.ru</a>  Phone = +71234567890. Если ключи или значения содержат в себе спец символы, то получившееся значение должно быть закодировано функцией urlencode. Максимальная длина для каждого передаваемого параметра: Ключ – 20 знаков, Значение – 100 знаков. Максимальное количество пар «ключ-значение» не может превышать 20
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "testRegress",
  "CardData": "U5jDbwqOVx+2vDapxe/rfACMt+rFWXzPdJ8ZXxNFViiZaLzrOW72bGe9cKZdIDnekW0nqm88YxRD4jyfa5Ru0kY5cQValU+juSlu1zpamSDtaGFeb8sRZfhj72yGw+io+qHGSBeorcfgoKStyKGuBPWfg4d0PLHuyBE6QgZyIAM1XfdmNlV0UAxOnkTGDsskLpIt3AWhw2e8KOar0vwbGCTDjznDB1/DLgOW014Aj/bXyLJoG1BkOrPBm9JURs+f+u"
```

```
yFae0hkRicNKNgXoN5pJTSQxOEauOi6ylsVJB3WK5MNYXtj6x4GlxcmTk/LD9kvHcjTeojcAlDzRZ87GdWeY8wgg==
",
"RequestKey": "13021e10-a3ed-4f14-bcd1-823b5ac37390",
"Token": "7241ac8307f349afb7bb9dda760717721bbb45950b97c67289f23d8c69cc7b96",
"DATA": {
  "Email": "a@test.com"
}
```

Формат ответа: JSON

Таблица 18. Параметры ответа

Имя	Тип	Обязательный?	Описание
TerminalKey	String	Да	Платежный ключ, выдается Продавцу при заведении терминала
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
RequestKey	String	Да	Идентификатор запроса на привязку карты
RebillId	String	Нет	Идентификатор рекуррентного платежа
CardId	String	Да	Идентификатор карты в системе Банка
Status	String	Да	Статус привязки карты
Success	bool	Да	Успешность операции
ErrorCode	String	Да	Код ошибки, «0» если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success": true,
  "ErrorCode": "0",
  "TerminalKey": "testRegress",
  "Status": "3DS_CHECKING",
  "CustomerKey": "testRegress5",
  "RequestKey": "8de92934-26c9-474c-a4ce-424f2021d24d"
  "CardId": "5555"
}
```

## Метод GetCardList

Описание: Возвращает список привязанных карт у покупателя

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

Метод: POST

Таблица 19. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца

Наименование	Тип	Обязательный?	Описание
IP	String	Нет	IP-адрес запроса
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "CustomerKey": "Customer1"
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: Массив JSON

Таблица 20. Параметры ответа

Наименование	Тип	Обязательный?	Описание
Pan	String	Да	Номер карты 411111*****1111
CardId	String	Да	Идентификатор карты в системе Банка
Status	String	Да	Статус карты: А – активная, I – не активная.
RebillId	Number	Да	Идентификатор рекуррентного платежа (см. параметр Recurrent в методе <i>Init</i> )
ExpDate	String	Нет	Срок действия карты

Пример ответа:

```
[{"CardId": "4750", "Pan": "543211*****4773", "Status": "A", "RebillId": "145919"},
{"CardId": "5100", "Pan": "411111*****1111", "Status": "I", "RebillId": "145917"}]
```

## Метод RemoveCard

Описание: Удаляет привязанную карту у покупателя

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

Метод: POST

Таблица 21. Параметры запроса

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CardId	Number	Да	Идентификатор карты в системе Банка

Наименование	Тип	Обязательный?	Описание
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
IP	String	Нет	IP-адрес запроса
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "CardId": "4750"
  "CustomerKey": "Customer1"
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6">
}
```

Формат ответа: JSON

Таблица 22. Параметры ответа

Наименование	Тип	Обязательный?	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CardId	Number	Да	Идентификатор карты в системе Банка
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
Success	bool	Да	Успешность операции
Status	String	Да	Статус карты: D – удалена
ErrorCode	String	Да	Код ошибки, «O» - успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

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

## Дополнительная информация

### AcquiringSdk

Для программиста SDK представлен 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;
- `password` - пароль от терминала;
- `publicKey` - публичный ключ продавца.

Помимо переданных параметров в конструкторе можно настроить `setLogger(Logger)`. Параметр 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`;

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`.

### 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`.

### Алгоритм формирования подписи запроса (Token)

Для формирования подписи необходимо:

1. Собрать все параметры запроса Ключ-Значение, кроме параметра `Token`.  
Например, `[["TerminalKey", "TestB"], ["PaymentId", "20150"]]`.
2. Добавить туда пару `Password-Значение`.  
`[["TerminalKey", "TestB"], ["PaymentId", "20150"], ["Password", "123456789"]]`.
3. Отсортировать по ключам.

```
[["Password","123456789"],["PaymentId","20150"],["TerminalKey","TestB"]].
```

4. Конкатенировать значения.

12345678920150TestB.

5. Вычислить SHA-256 от пункта 4.

### 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 с использованием publicKey в качестве ключа.
3. Полученная криптограмма на шаге 2 энкодится алгоритмом Base64.



## 11. Коды ошибок API и возможные исключения

Ошибки должны пробрасываться классом AcquiringSdk как исключения AcquiringApiException.

API может возвращать следующие ошибки:

Таблица 23. 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 {O} 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 {O} превышает максимально допустимый размер
208	Наименование ключа из параметра DATA превышает максимально допустимый размер
209	Значение ключа из параметра DATA превышает максимально допустимый размер
210	Поле {O} должно быть формата "{regex}"
211	Неверный формат IP
212, 213	Поле {O} должно быть формата "{regex}"

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

Таблица 24. 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
1014	Неверно введены реквизиты карты. Проверьте корректность введенных данных
1030	Повторите попытку позже
1033	Проверьте реквизиты или воспользуйтесь другой картой

Код ошибки	Описание
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

## 12. Поддержка

- Github: <https://github.com/TinkoffCreditSystems/tinkoff-asdk-android>;
- Баги и feature-реквесты можно направлять в раздел [issues](#);
- Подробное описание методов: [API методов](#).