PRINTEC POSAPI – Версия 1.23
Интерфейс интеграции с ПОС терминалом

Руководство разработчика

ИСТОРИЯ ИЗМЕНЕНИЙ

Версия	Дата	Описание
1.5	22.06.2011	В метод `pos_open()` добавлен параметр *`log`* включающий журналирование вызовов библиотеки.
1.5	22.06.2011	Отныне параметры *`POS_CARD_PAN`* и *`POS_CARD_EXPIRY`* могут быть установлены с помощью `pos_set()` для подмены ручного ввода информации о карте клиента.
1.6	06.12.2011	Класс интеграции с JAVA помещен в пакет `com.cardpay.pos`
1.8	07.02.2013	В метод `pos_get()` добавлен параметр *`POS_STATUS`*, который возвращает код причины, по которой операция не была подтверждена.
1.9	14.01.2014	Добавлены печать отчетов, копий квитанций и закрытия дня. Получение статуса последней финансовой операции. Добавлено описание механизма восстановления.
1.10	24.01.2014	Добавлено описание алгоритма использования библиотеки.
1.11	17.03.2014	Добавлена возможность проводить операции оплаты со скидкой. Добавлена операция чтения карты (*`ACTION_READ_CARD)`*. Добавлены параметры: *`POS_CARD_TRACK1`*, *`POS_CARD_TRACK2`*, *`POS_CARD_TRACK3`*, *`POS_CARD_PAYMENT`*, *`POS_ORIG_AMOUNT`*. **Поддерживается в USO v3.107C и выше.**
1.12	16.01.2015	Добавлена частичная отмена (*`ACTION_PARTIAL_REVERSAL`*) для операций: преавторизация и оплата.
1.13	12.02.2015	Добавлена операция ввода кода проверки для дисконтных карт (*`ACTION_GET_VERIF_CODE`*).
1.13.1	09.04.2015	В операцию закрытия дня (*`ACTION_CLOSE_DAY`*) добавлен необязательный параметр код профиля (*`POS_PROFILE`*). Вызов закрытия дня с указанием профиля позволяет выполнить эту операцию только для соответствующего хоста. **Поддерживается в USO v3.108D и выше.**
1.14	06.08.2015	В API добавлены два новых метода `pos_get_length()` и `pos_get_max_length()`. Добавлено описание класса интеграции с платформой .NET
1.15	25.09.2015	Добавлена операция «Идентификация карты» `(ACTION_CARD_VERIFICATION)` для протокола OpenWay (Oschadniy).
1.16	14.12.2015	Изменена функция pos_open(). Вместо одной попытки установить связь с терминалом (в старой версии), осуществляется несколько попыток в течении 15 сек (в новой версии).
1.17	02.02.2016	Добавлена операция `ACTION_CREDIT_VOUCHER` для протокола OpenWay. **Поддерживается в USO v3.109C и выше.**
1.18	11.02.2016	Добавлена операция смены кода проверки (*`ACTION_CHANGE_VERIF_CODE`*). В операции ввода кода проверки (*`ACTION_GET_VERIF_CODE`*) и смены кода проверки (*`ACTION_CHANGE_VERIF_CODE`*) добавлен режим передачи введенных данных в зашифрованном виде. Этот режим включается параметром: *`POS_ENCRIPTION_SIGN`*. **Поддерживается в USO v3.109D и выше.**

1.19	16.02.2016	Добавлены операции оплаты топлива купонами (описание в «posapi_OKKO»). Добавлен параметр: CVV2 код (*POS_CARD_CVV2)*. **Поддерживается в USO v3.109E и выше.**
1.20	09.06.2016	В сообщение *RESP_IDENTIFIER* добавлено поле *POS_CARD_LOYALTY_CODE*. Это поле может использоваться кассой для определения размера скидки.
1.21	16.09.2016	Добавлены параметры: *POS_CARD_PAN_SHA256* и *POS_TIPS*. **Поддерживается в USO v3.109I и выше.**
1.22	19.01.2017	- Добавлена операция проверки связи с хостом – *ACTION_HOST_ECHO_TEST*. - В сообщения (ответы, получаемых с ПОС терминала) *RESP_CONFIRM* и RESP_DECLINE добавлены параметры: *POS_MERCHANT_ID, POS_TERMINAL_ID*
1.23	09.06.2017	- Добавлена операция показа QR кода – *ACTION_SHOW_QR_CODE*. - В сообщение (посылаемое от кассы в ПОС терминал) добавлены параметры: *POS_DATA, POS_TIMEOUT и POS_OPTION*

СОДЕРЖАНИЕ

1 ОБЩАЯ ИНФОРМАЦИЯ

Интерфейс POSAPI предназначен для глубокой интеграции ПОС терминалов под управлением USO v.3.100+ с программными комплексами банковских систем, кассовых аппаратов, электронных киосков, автоматов самообслуживания и прочих систем.

Для использования данного интерфейса необходимо физическое подключение ПОС терминала к соответствующей системе через последовательный порт (с помощью специального кабеля). Пожалуйста, уточните наличие такой возможности у поставщика данного решения.

Интерфейс реализован на языке программирования C и может поставляться в виде статических и динамических библиотек. Это позволяет интегрировать его с большинством промышленных средств разработки и популярных языков программирования.

На данный момент существуют реализации для платформ Win32/64 и Linux, при этом библиотеки могут быть доступны в следующих вариантах поставки:

Платформа	Файл	Описание
Win32/64	POS.LIB	Статическая библиотека MSVC-2015
	POSAPI.DLL	Динамическая библиотека Windows
	POSAPI2.DLL	Спец. реализация в виде объекта COM
	POSJAVA.DLL	Спец. библиотека для интеграции с JAVA (через JNI)
	POSNET.DLL	Спец. библиотека для интеграции с .NET
Linux	LIBPOS.A	Статическая библиотека GCC
	LIBPOS.SO	Динамическая библиотека Linux
	LIBPOSJAVA.SO	Спец. библиотека для интеграции с JAVA (через JNI)

В комплект поставки также входит простое тестовое приложение, основанное на статической библиотеке. Оно позволяет разработчику проверить корректность подключения ПОС терминала к ПК и его работоспособность.

Исходный код библиотек не поставляется и не распространяется, при необходимости их сборки под конкретный дистрибутив Linux или переноса на другую платформу уточните наличие такой возможности и ее условия у разработчика.

Далее в данном документе приводится описание интерфейса POSAPI и особенностей его различных реализаций, а также примеры их использования.

2 ОПИСАНИЕ МЕТОДОВ ИНТЕРФЕЙСА

В данном разделе описываются базовые реализации интерфейса, доступные в виде статических и динамических библиотек. При использовании прочих, специализированных реализаций, синтаксис вызовов может несколько отличаться (полная информация обо всех отличиях содержится в соответствующих разделах), однако принципы их использования полностью идентичны базовым реализациям.

В самом общем виде корректный алгоритм взаимодействия с интерфейсом может быть представлен в виде следующей последовательности вызовов:

1. Открытие сеанса взаимодействия с терминалом с помощью `pos_open()`.
2. Установка параметров запроса с помощью `pos_set()`.
3. Отсылка запроса с помощью `pos_send()`.
4. Считывание ответа с помощью `pos_receive()`.
5. Получение параметров ответа с помощью `pos_get()` или же с помощью пары методов `pos_get_first()` / `pos_get_next()`. Для получения размеров буферов,

необходимых для хранения значений параметров можно воспользоваться методами `pos_get_length()` и `pos_get_max_length()`.

6. Закрытие сеанса взаимодействия с терминалом с помощью `pos_close()`.

Более подробно особенности использования методов интерфейса раскрываются ниже.

2.1 Открытие сеанса: pos_open()

Данный метод предназначен для открытия сессии взаимодействия с ПОС терминалом и получения ее дескриптора, т.е. для начала любой операции. После завершения операции открытая сессия должна быть закрыта с помощью `pos_close()`.

Внимание:

- в течение каждой сессии может быть выполнено не более одного запроса на проведение новой операции с помощью `pos_send()`;
- в каждый определенный момент времени может быть открыто не более одной сессии взаимодействия с определенным ПОС терминалом;
- при нарушении данных правил поведение библиотеки и ПОС терминала может быть непредсказуемым.

Синтаксис:

```
bool pos_open(POS_HANDLE *handle_p, const char *name,
              const char *log)
```

Параметры:

handle_p Указатель на дескриптор сессии, значение которого устанавливается библиотекой в случае успешного завершения операции. Данный дескриптор должен использоваться при всех последующих вызовах методов в рамках сессии. В случае неудачи значение данного параметра устанавливается в *POS_NONE*.

name Назначенное операционной системой имя последовательного порта, к которому подключен ПОС терминал.

log Файл журнала. Для отключения журналирования вызовов библиотеки следует установить данный параметр в *NULL* или передать указатель на пустую строку. Для включения журналирования следует передать указатель на строку с полным именем файла, корректным для данной платформы. В случае неудачи открытия или создания данного файла параметр будет проигнорирован.

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

2.2 Закрытие сеанса: pos_close()

Данный метод предназначен для закрытия сессии, открытой ранее с помощью `pos_open()`, и освобождения ресурсов системы и ПОС терминала.

Синтаксис:

```
bool pos_close(POS_HANDLE *handle_p)
```

Параметры:

handle_p Указатель на дескриптор открытой сессии. В случае успешного завершения операции дескриптору присваивается значение *POS_NONE*.

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

2.3 Отсылка запроса: pos_send()

Данный метод отсылает ПОС терминалу запрос на выполнение операции с заданным кодом. При этом терминалу также отсылаются все параметры, установленные с помощью `pos_set()` перед вызовом данного метода.

Синтаксис:

```
bool pos_send(POS_HANDLE handle, int action)
```

Параметры:

handle Дескриптор сессии.

action Код операции. Допустимые значения:

ACTION_CASH	–	Запрос на выдачу наличных.
ACTION_DEPOSIT	–	Запрос на внесение наличных на счёт.
ACTION_PAYMENT	–	Запрос на оплату товара или услуги.
ACTION_RETURN	–	Запрос на возврат товара.
ACTION_PREAUTH	–	Запрос на преавторизацию.
ACTION_COMPLETE	–	Запрос на завершение преавторизации, выполненной ранее.
ACTION_REVERSAL	–	Запрос на отмену финансовой операции, выполненной ранее (в успешно завершенной сессии в течение текущего финансового дня).
ACTION_PARTIAL_REVERSAL		Запрос на частичную отмену операции преавторизации или оплаты, выполненной ранее (в успешно завершенной сессии в течение текущего финансового дня).
ACTION_BALANCE	–	Запрос баланса карточного счета (сумма баланса не возвращается, только статус выполнения).
ACTION_CLOSE_DAY	–	Закрытие текущего финансового дня.
ACTION_BREAK	–	Запрос на прерывание операции, находящейся в процессе выполнения ПОС терминалом. Операция может быть прервана не всегда, в случае успеха код ответа, полученный с помощью последующего вызова `pos_receive()`, будет равен *RESP_BREAK*.
ACTION_TEST	–	Проверка соединения с ПОС терминалом. ***Начиная с версии 1.9 использовать эту команду не рекомендуется. Проверка соединения с ПОС терминалом осуществляется на этапе выполнения функции pos_open().***
ACTION_REPORT	–	Запрос на печать отчета
ACTION_COPY_RECEIPT	–	Запрос на печать копии квитанции: По номеру, если в запросе задан номер квитанции (*POS_TRANS_RECEIPT*), или последней, если номер квитанции не задан.
ACTION_COPY_CLOSE_DAY	–	Запрос на печать копии отчета закрытия дня
ACTION_STATUS	–	Запрос статуса последней финансовой

операции

ACTION_REVISION	–	Запрос на смену суммы транзакции после считывания номера карты (дисконт/комиссия)
ACTION_READ_CARD	–	Запрос на чтение карты
ACTION_GET_VERIF_CODE	–	Запрос на ввод кода проверки
ACTION_CHANGE_VERIF_CODE	–	Запрос на смену кода проверки
ACTION_CARD_VERIFICATION	–	Запрос на проведение операции «Идентификация карты» по протоколу OpenWay (Oschadniy).
ACTION_HOST_ECHO_TEST	–	Запрос на проверку связи с удаленным хостом
ACTION_SHOW_QR_CODE	–	Запрос на отображение QR кода

Замечание: ПОС терминалы, интегрируемые в автоматы самообслуживания, поддерживают лишь ограниченный набор операций: *ACTION_PAYMENT*, *ACTION_REVERSAL*, *ACTION_CLOSE_DAY*.

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

2.4 Считывание ответа: pos_receive()

Данный метод позволяет считать ответ ПОС терминала на запрос, отосланный ранее. ПО обязано вызвать данный метод, если перед этим был успешно вызван метод `pos_send()`.

Синтаксис:

```
int pos_receive(POS_HANDLE handle, int timeout)
```

Параметры:

handle	Дескриптор сессии.
timeout	Время ожидания ответа, в миллисекундах.

Возвращаемое значение:

Целочисленное, представляет из себя код ответа ПОС терминала на ранее отосланный запрос. Допустимые значения:

RESP_TIMEOUT	–	Истекло время ожидания ответа, метод следует вызвать еще раз.
RESP_BREAK	–	Операция прервана ПОС терминалом.
RESP_CONFIRM	–	Подтверждение успешного выполнения запрошенной операции.
RESP_DECLINE	–	Запрошенная операция отклонена терминалом или процессинговым центром.
RESP_MESSAGE	–	Уведомление о текущем этапе выполнения ПОС терминалом запрошенной операции. Информация, содержащаяся в данном ответе, может использоваться вызывающим ПО для вывода на большой экран с целью более комфортного уведомления кассира или клиента. Данный ответ может также игнорироваться вызывающим ПО. В случае получения данного ответа метод `pos_receive()` следует вызвать еще раз.
RESP_IDENTIFIER	–	Уведомление о присвоенном идентификаторе транзакции.
RESP_KEEPALIVE	–	Уведомление о том, что ПОС терминал продолжает выполнять запрос кассы, и требуется дополнительное время для получения

ответа. Это уведомление ПОС терминал периодически посылает на кассу в процессе выполнения длительных операций. ***В данной версии протокола не поддерживается. Зарезервировано для дальнейшего использования.***

Прочие значения зарезервированы для будущего использования; в случае их получения ПО должно прервать текущую операцию, отправив запрос *ACTION_BREAK*.

2.5 Установка параметра: pos_set()

Данный метод устанавливает значение параметра, которое будет отослано ПОС терминалу в ближайшем последующем вызове метода `pos_send()`.

Синтаксис:

```
bool pos_set(POS_HANDLE handle, const char *param, const char *val)
```

Параметры:

handle	Дескриптор сессии.
param	Имя параметра (С-строка).
val	Значение параметра (С-строка).

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

Допустимые значения:

Имя параметра	Описание	Значение параметра
POS_AMOUNT	Сумма операции. Обязательна в: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH*. Может использоваться в: *ACTION_REVERSAL*, *ACTION_COMPLETE*.	Строковое представление целого числа длиной от 1 до 12 символов, соответствующее сумме в минимальных единицах заданной валюты.
POS_TIPS	Сумма чаевых. Может использоваться в: *ACTION_PAYMENT*	Строковое представление целого числа длиной от 1 до 12 символов, соответствующее сумме чаевых в минимальных единицах заданной валюты.
POS_CURRENCY	Код валюты операции. Обязателен в: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH* Может использоваться в: *ACTION_REVERSAL*, *ACTION_COMPLETE*, *ACTION_BALANCE*.	Строковое представление числа длиной 3 символа. Наиболее часто используемые коды: 980 (UAH), 840 (USD), 948 (EUR), 810 (RUR).
POS_TRANS_ID	Идентификатор транзакции. Обязателен в:	Строковое представление целого числа. Возвращается

	ACTION_STATUS. Может использоваться в: *ACTION_REVERSAL*	при проведении любой финансовой операции. Служит для однозначной идентификации транзакций в терминале при их отмене, а также получения статуса завершенной операции.
POS_PROFILE	Код профиля авторизации Может использоваться в: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH*, *ACTION_COMPLETE*, *ACTION_REVERSAL*, *ACTION_BALANCE*.	Строковое значение. Может устанавливаться, если приложение терминала настроено на использование нескольких профилей авторизации. Если установлен, заменяет собой выбор профиля, выполняемый кассиром вручную на терминале. При интеграции терминала с несколькими профилями в автоматы самообслуживания является ОБЯЗАТЕЛЬНЫМ параметром во всех указанных командах, т.к. кассир отсутствует. Используемые значения должны соответствовать фактическим настройкам терминала. На данный момент коды профилей соответствуют идентификаторам терминала (Terminal ID), назначенным процессинговым центром.
POS_CARD_PAN	Номер карты клиента. Может использоваться в: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH*, *ACTION_COMPLETE*, *ACTION_REVERSAL*, *ACTION_BALANCE*.	Строковое представление числа длиной от 8 до 19 цифр. Может устанавливать-ся, если ПОС терминал разрешает ручной ввод информации о карте для данной операции. При этом также обязательно должен быть установлен параметр *POS_CARD_EXPIRY* (срок годности карты клиента).
POS_CARD_EXPIRY	Срок годности карты. Может использоваться в: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH*, *ACTION_COMPLETE*, *ACTION_REVERSAL*,	Строка длиной 4 символа в формате "YYMM". Может устанавливаться, если ПОС терминал разрешает ручной ввод информации о карте для данной операции. При этом также обязательно должен быть установлен параметр *POS_CARD_PAN* (номер карты

	ACTION_BALANCE.	клиента).
POS_CARD_CVV2	CVV2 код карты.	Строковое представление числа длиной до 4 цифр.
POS_TRANS_CODE	Код оригинальной транзакции, он же – Reference Number. Может использоваться в: *ACTION_RETURN*, *ACTION_REVERSAL*, *ACTION_COMPLETE*	Строка, содержимое которой зависит от типа процессингового центра. Чаще всего – строковое представление числа длиной 12 символов. Идентифицирует транзакцию в процессинговом центре. Устанавливаемое значение должно точно совпадать со значением, возвращенным терминалом в подтверждении оригинальной операции. Параметр необходим при подключении к процессингам типа WAY4.
POS_TRANS_APPROVAL	Код оригинальной авторизации. Может использоваться в: *ACTION_COMPLETE*, *ACTION_REVERSAL*	Строка, содержимое которой зависит от типа процессингового центра. Чаще всего – длиной 6 символов. Устанавливаемое значение должно точно совпадать со значением, возвращенным терминалом в подтверждении оригинальной опреации. Параметр необходим при подключении к процессингам типа WAY4.
POS_TRANS_ACTION	Код оригинальной операции. Может использоваться в: *ACTION_REVERSAL*	Строка длиной 4 символа – шестнадцатеричное представление числового кода оригинальной операции. Допустимые коды операций: *ACTION_CASH*, *ACTION_DEPOSIT*, *ACTION_PAYMENT*, *ACTION_RETURN*, *ACTION_PREAUTH*, *ACTION_COMPLETE*. Параметр необходим при подключении к процессингам типа WAY4.
POS_TRANS_MSGCODE	Код оригинального сообщения. Может использоваться в: *ACTION_REVERSAL*	Строка, содержимое которой зависит от типа процессингового центра. Чаще всего – строковое представление числа длиной 4 символа (MTID). Устанавливаемое значение должно точно совпадать со

		значением, возвращенным терминалом в подтверждении оригинальной операции. Параметр необходим при подключении к процессингам типа WAY4.
POS_TRANS_RECEIPT	Номер чека. Может использоваться в: *ACTION_COPY_RECEIPT*, *ACTION_REVERSAL*	Строковое представление целого числа длиной 6 символов.
POS_REVISION_SIGN	Признак изменения суммы. Может использоваться в: *ACTION_PAYMENT*, *ACTION_CASH*	Признак изменения суммы операции после считывания карты (дисконт/комиссия) 1 – После считывания карты ожидать измененную сумму операции.
POS_REPORT	Идентификатор отчета. Обязателен в: *ACTION_REPORT*	Идентификатор отчета: 1 – Суммарный отчет. 2 – Отчет по типам карт. 3 – Отчет по квитанциям.
POS_ENCRYPTION_SIGN	Признак шифрования данных. Может использоваться в: *ACTION_GET_VERIF_CODE*, *ACTION_CHANGE_VERIF_CODE*	Признак передачи данных в зашифрованном виде: 1 – Шифровать данные. 0 – Передавать в нешифрованном виде.
POS_PRINT_RECEIPT	Признак печати чека. Обязателен в: *ACTION_FUEL_COUPON_PAYMENT*.	Признак печати чека: 0 – Чек не печатать. 1 – Чек печатать.
POS_DATA	Строка с данными. Может использоваться в: *ACTION_SHOW_QR_CODE*	Строковые данные. Для *ACTION_SHOW_QR_CODE строка длиной до 197 символов для отображения QR кода.*
POS_TIMEOUT	Тайм-аут для выполнения операции. Не обязателен в: *ACTION_SHOW_QR_CODE*	Для *ACTION_SHOW_QR_CODE задает тайм-аут отображения QR кода на терминале(в сек.)*
POS_OPTION	Зарезервирован. Не обязателен в: *ACTION_SHOW_QR_CODE*	Для *ACTION_SHOW_QR_CODE зарезервирован.*

Точный набор обязательных параметров и их значения зависят от типа процессингового центра и набора используемых операций. В будущем список устанавливаемых параметров может быть расширен.

2.6 Считывание значения параметра: pos_get()

Данный метод считывает значение параметра, возвращенное ПОС терминалом в последнем вызове метода `pos_receive()`, по его имени. Если параметр отсутствует, устанавливает значение в пустую строку и возвращает ложь.

Синтаксис:

```
bool pos_get(POS_HANDLE handle, const char *param,
             char *val, int val_size)
```

Параметры:

handle	Дескриптор сессии.
param	Имя параметра (С-строка).
val	Буфер для считывания значения параметра (С-строки).
val_size	Размер буфера *val*, в байтах.

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

Допустимые значения:

Имя параметра	Описание	Значение параметра
POS_AMOUNT	Сумма операции. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строковое представление числа длиной от 1 до 12 символов, соответствующее сумме в минимальных единицах заданной валюты.
POS_CURRENCY	Код валюты операции. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строковое представление числа длиной 3 символа. Наиболее часто используемые коды: 980 (UAH), 840 (USD), 948 (EUR), 810 (RUR).
POS_TRANS_ID	Идентификатор транзакции. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*, *RESP_IDENTIFIER*.	Строковое представление целого числа. Возвращается при проведении любой финансовой операции. Служит для однозначной идентификации транзакций в терминале при их отмене, а также получения статуса завершенной операции.
POS_TRANS_RECEIPT	Номер чека. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строковое представление целого числа длиной 6 символов.
POS_PROFILE	Код профиля авторизации.	Строка. Может возвращаться терминалом, содержащим приложение с несколькими профилями. На данный момент коды профилей соответствуют идентификаторам терминала (Terminal ID), назначенным процессинговым центром.
POS_TRANS_CODE	Код транзакции, он же – Reference Number. Может возвращаться в *RESP_CONFIRM*.	Строка, содержимое которой зависит от типа процессингового центра. Чаще всего – строковое представление числа длиной 12 символов. Идентифицирует транзакцию в процессинговом центре.
POS_TRANS_APPROVAL	Код авторизации. Может возвращаться в	Зависит от типа процессингового центра. Чаще всего – строка длиной

	RESP_CONFIRM.	6 символов.
POS_TRANS_STATUS	Сообщение об отказе. Может возвращаться в *RESP_DECLINE*.	Сообщение длиной до 50 символов с расшифровкой отказа при проведении финансовой операции (для вывода на экран).
POS_DATE_TIME	Дата и время операции. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строка длиной 14 символов в формате "YYYYMMDDhhmmss".
POS_CARD_PAN	Номер карты клиента. Может возвращаться в *RESP_IDENTIFIER*, *RESP_CONFIRM*, *RESP_DECLINE*.	Строковое представление числа длиной от 8 до 19 цифр. Как правило, частично маскируется символами звездочки '*' в соответствии с требованиями безопасности платежных систем и банка.
POS_CARD_PAN_SHA256	SHA256 код номера карты клиента. Может возвращаться в *RESP_IDENTIFIER*, *RESP_CONFIRM*, *RESP_DECLINE*.	32-байтное хэш-значение номера карты клиента, рассчитанное по алгоритму SHA256. Передается в виде 64 HEX символов.
POS_CARD_EXPIRY	Срок годности карты. Может возвращаться в *RESP_IDENTIFIER*, *RESP_CONFIRM*, *RESP_DECLINE*.	Строка длиной 4 символа в формате "YYMM". Может не возвращаться в соответствии с требованиями безопасности платежных систем и банка.
POS_CARD_HOLDER	Имя владельца карты. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строка длиной до 26 символов содержит имя, считанное ПОС терминалом с карты.
POS_MSG_TITLE	Заголовок сообщения. Может возвращаться в *RESP_MESSAGE*.	Заголовок сообщения длиной до 50 символов (для вывода на экран).
POS_MSG_BODY	Тело сообщения. Может возвращаться в *RESP_MESSAGE*.	Тело сообщения длиной до 200 символов (для вывода на экран).
POS_MSG_BREAK	Признак прерывания. Может возвращаться в *RESP_MESSAGE*, *RESP_IDENTIFIER*.	Строковое представление чисел 1 или 0. Признак того, что на данном этапе запрошенную операцию еще можно прервать командой *ACTION_BREAK*.
POS_PRINT	Текст для печати. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Если ПОС терминал не имеет встроенного принтера, в данном параметре возвращается текст квитанций и отчетов, которые при получении обязано распечатать вызывающее приложение. Формат данных: обычный текст, строки текста разделяются символами конца строки '\n', чеки (если их несколько)

		– символами конца страницы '\f'.
POS_STATUS	Код отказа. Может возвращаться в *RESP_DECLINE*.	Может принимать такие значения: • 1 - в авторизации отказано, без уточнения причин; • 2 - ошибка коммуникаций; • 3 - ошибка криптографии; • 4 - сбой процессингового центра; • 5 - мало денег; • 6 - издатель недоступен;
POS_TRANS_MSGCODE	Код оригинального сообщения. Может возвращаться в *RESP_CONFIRM*	Строка, содержимое которой зависит от типа процессингового центра. Чаще всего – строковое представление числа длиной 4 символа (MTID). Устанавливаемое значение должно точно совпадать со значением, возвращенным терминалом в подтверждении оригинальной операции. Параметр необходим при подключении к процессингам типа WAY4.
POS_CARD_TRACK1	Track1 дисконтной карты. Может возвращаться в *RESP_CONFIRM* в ответе на *ACTION_READ_CARD*	Track1 дисконтной карты.
POS_CARD_TRACK2	Track2 дисконтной карты. Может возвращаться в *RESP_CONFIRM* в ответе на *ACTION_READ_CARD*	Track2 дисконтной карты.
POS_CARD_TRACK3	Track3 дисконтной карты. Может возвращаться в *RESP_CONFIRM* в ответе на *ACTION_READ_CARD*	Track3 дисконтной карты.
POS_CARD_PAYMENT	Признак платежной карты. Может возвращаться в *RESP_CONFIRM* в ответе на *ACTION_READ_CARD*	Строковое представление чисел 1 или 0. Признак того, что считанная карта является платежной = «1», или дисконтной = «0».
POS_ORIG_AMOUNT	Оригинальная сумма операции. Может возвращаться в *RESP_CONFIRM*, *RESP_DECLINE*.	Строковое представление числа длиной от 1 до 12 символов, соответствующее оригинальной сумме (без скидки) в минимальных единицах заданной валюты.
POS_CARD_VERIF_CODE	Код проверки дисконтной карты. Может возвращаться в *RESP_CONFIRM*	Строковое представление целого числа длиной от 4 до 12 символов.
POS_CARD_NEW_VERIF_CODE	Новый код проверки	Строковое представление целого

	дисконтной карты. Может возвращаться в *RESP_CONFIRM*	числа длиной от 4 до 12 символов.
POS_CARD_LOYALTY_CODE	Код лояльности карты. Может возвращаться в *RESP_IDENTIFIER*	Дополнительные данные, записанные на магнитной полосе или в чипе карты, позволяющие определить размер скидки для данной карты. Для разных типов карт эта информация может считываться из разных тэгов.

В будущем список возвращаемых параметров может быть расширен.

2.7 Считывание всех параметров: pos_get_first() и pos_get_next()

Данные методы предназначены для последовательного считывания имен и значений всех параметров, возвращенных ПОС терминалом в последнем вызове метода `pos_receive()`. Используется путем предварительного вызова `pos_get_first()` и последующего циклического вызова `pos_get_next()` до тех пор, пока его возвращаемое значение будет оставаться истинным.

Синтаксис:

```
bool pos_get_first(POS_HANDLE handle, char *param, int param_size,
                   char *val, int val_size)
bool pos_get_next(POS_HANDLE handle, char *param, int param_size,
                   char *val, int val_size)
```

Параметры:

handle	Дескриптор сессии.
param	Буфер для считывания имени параметра (С-строки).
param_size	Размер буфера *param*, в байтах.
val	Буфер для считывания значения параметра (С-строки).
val_size	Размер буфера *val*, в байтах.

Возвращаемое значение:

Логическое, сигнализирует об успешности выполнения операции.

Набор считываемых данными методами параметров совпадает с приведенным в описании метода `pos_get()`. Однако, в будущем список возможных параметров может быть расширен, поэтому реализация вызывающего приложения должна учитывать этот факт и корректно обрабатывать ситуации, когда имя считанного параметра неизвестно.

2.8 Функции получения размера параметров: pos_get_length() и pos_get_max_length()

Эти функции позволяют определить необходимый размер буфера для хранения значений параметров, возвращенных ПОС терминалом в последнем вызове метода pos_receive().

Синтаксис:

```
int pos_get_length(POS_HANDLE handle, const char *param)
```

Параметры:

handle	Дескриптор сессии.

param Имя параметра (C-строка).

Возвращаемое значение:

Возвращает размер буфера, необходимый для сохранения значения параметра *param*.

Синтаксис:

```
int pos_get_max_length(POS_HANDLE handle)
```

Параметры:

handle Дескриптор сессии.

Возвращаемое значение:

Возвращает размер буфера, необходимый для сохранения максимально длинного из значений всех параметров.

2.9 Список запросов, отсылаемых на ПОС терминал

Описание каждого поля содержит «Признак наличия», который указывает должно ли поле присутствовать в запросе/ответе: M – поле обязательно должно быть, O – может присутствовать, но наличие его не обязательно, WAY4 – поле должно быть, если используется протокол OpenWay.

2.9.1 ACTION_CASH - Запрос на выдачу наличных

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.2 ACTION_DEPOSIT - Запрос на внесение наличных на счёт

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.3 ACTION_PAYMENT - Запрос на оплату товара или услуги

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_TIPS	Сумма чаевых	O
POS_CURRENCY	Код валюты операции	M
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.4 ACTION_RETURN - Запрос на возврат товара

Имя параметра	Описание	Признак наличия

POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_TRANS_CODE	Код оригинальной транзакции (Reference Number)	O
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.5 ACTION_PREAUTH - Запрос на преавторизацию

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.6 ACTION_COMPLETE - Запрос на завершение преавторизации, выполненной ранее

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_TRANS_CODE	Код оригинальной транзакции (Reference Number)	M
POS_TRANS_APPROVAL	Код оригинальной авторизации	O
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.7 ACTION_REVERSAL - Запрос на отмену финансовой операции, выполненной ранее

Имя параметра	Описание	Признак наличия
POS_TRANS_ID	Идентификатор транзакции	M[1]
POS_TRANS_RECEIPT	Номер чека	M[1]
POS_AMOUNT	Сумма операции	WAY4
POS_CURRENCY	Код валюты операции	O
POS_TRANS_CODE	Код оригинальной транзакции (Reference Number)	WAY4
POS_TRANS_APPROVAL	Код оригинальной авторизации	WAY4
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O
POS_TRANS_ACTION	Код оригинальной операции	WAY4
POS_TRANS_MSGCODE	Код оригинального сообщения	WAY4

Примечание: M[1] - для выполнения команды ACTION_REVERSAL необходим один (любой) из 2 параметров: *POS_TRANS_ID* или *POS_TRANS_RECEIPT.*

2.9.8 ACTION_PARTIAL_REVERSAL - Запрос на частичную отмену преавторизации или оплаты, выполненной ранее

Имя параметра	Описание	Признак наличия
POS_TRANS_ID	Идентификатор транзакции	M[1]
POS_TRANS_RECEIPT	Номер чека	M[1]
POS_AMOUNT	Сумма частичной отмены. Не должна превышать сумму отменяемой операции.	M
POS_CURRENCY	Код валюты операции	O
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

Примечание: M[1] - для выполнения команды ACTION_PARTIAL_REVERSAL необходим один (любой) из 2 параметров: *POS_TRANS_ID* или *POS_TRANS_RECEIPT.*

2.9.9 ACTION_CREDIT_VOUCHER – Запрос на проведение операции “credit voucher” протокола OpenWay

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_CURRENCY	Код валюты операции	M
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.10 ACTION_BALANCE - Запрос баланса карточного счета

Имя параметра	Описание	Признак наличия
POS_CURRENCY	Код валюты операции	O
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.11 ACTION_CLOSE_DAY - Закрытие текущего финансового дня

Имя параметра	Описание	Признак наличия
POS_PROFILE	Код профиля авторизации	O

Примечание:

При вызове этой операции без параметров, терминал последовательно выполняет «закрытие дня» для всех хостов, на которые он настроен. Независимо от успешности выполнения этой операции по каждому хосту, терминал **всегда** возвращает ответ *RESP_CONFIRM.* В этом случае контроль над выполнением операции возложен на кассира.

Для автоматизации контроля над результатами выполнения «закрытия дня», эту операцию нужно вызывать с параметром POS_PROFILE, для каждого хоста отдельно.

2.9.12 ACTION_BREAK - Запрос на прерывание операции, находящейся в процессе выполнения POS терминалом

Без параметров.

2.9.13 ACTION_REPORT - Запрос на печать отчета

Имя параметра	Описание	Признак наличия
POS_REPORT	Идентификатор отчета	M

2.9.14 ACTION_COPY_RECEIPT - Запрос на печать копии квитанции

Имя параметра	Описание	Признак наличия
POS_TRANS_RECEIPT	Номер чека	O

2.9.15 ACTION_COPY_CLOSE_DAY - Запрос на печать копии отчета закрытия дня

Без параметров.

2.9.16 ACTION_STATUS - Запрос статуса последней финансовой операции

Имя параметра	Описание	Признак наличия
POS_TRANS_ID	Идентификатор транзакции	M

2.9.17 ACTION_READ_CARD - Запрос на чтение карты

Без параметров.

2.9.18 ACTION_GET_VERIF_CODE - Запрос на ввод кода проверки

Имя параметра	Описание	Признак наличия
POS_ENCRYPTION_SIGN	Признак передачи данных в шифрованном виде	O

2.9.19 ACTION_CHANGE_VERIF_CODE - Запрос на смену кода проверки

Имя параметра	Описание	Признак наличия
POS_ENCRYPTION_SIGN	Признак передачи данных в шифрованном виде	O

2.9.20 ACTION_CARD_VERIFICATION - Запрос на проведение операции «Идентификация карты» по протоколу OpenWay (Oschadniy).

Имя параметра	Описание	Признак наличия
POS_CURRENCY	Код валюты операции	O
POS_PROFILE	Код профиля авторизации	O
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_EXPIRY	Срок годности карты	O

2.9.21 ACTION_HOST_ECHO_TEST – Проверка связи с удаленным хостом

Имя параметра	Описание	Признак наличия
POS_PROFILE	Код профиля авторизации	O

Примечание:

При успешности выполнения этой операции терминал возвращает ответ *RESP_CONFIRM*, иначе - *RESP_DECLINE*. Если передан неверный код профиля («Tetminal ID») терминал возвращает - *RESP_BREAK*.

Если **не указан** код профиля, а в терминале прописано несколько записей в хостовой таблице (несколько профилей), терминал выдает сообщение (wrong profile) и возвращает - *RESP_BREAK*.

2.9.22 ACTION_SHOW_QR_CODE- Запрос печати QR кода

Имя параметра	Описание	Признак наличия
POS_DATA	Строка с данными для печати QR кода (длиной до 197 символов, вкючительно)	M
POS_TIMEOUT	Задержка (в сек) для показа QR кода	O
POS_OPTION	Резерв	O

Примечание:

При успешности выполнения этой операции терминал возвращает ответ *RESP_CONFIRM (без параметров)*, иначе терминал возвращает - *RESP_BREAK*.

2.10 Список ответов, получаемых с ПОС терминала

2.10.1 RESP_TIMEOUT - Истекло время ожидания ответа

Без параметров.

2.10.2 RESP_BREAK - Операция прервана ПОС терминалом

Без параметров.

2.10.3 RESP_CONFIRM - Подтверждение успешного выполнения запрошенной финансовой операции (ACTION_CASH, ACTION_DEPOSIT, ACTION_PAYMENT, ACTION_RETURN, ACTION_PREAUTH, ACTION_COMPLETE, ACTION_REVERSAL, ACTION_PARTIAL_REVERSAL, ACTION_BALANCE, ACTION_STATUS, ACTION_CARD_VERIFICATION)

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_TIPS	Сумма чаевых	O
POS_CURRENCY	Код валюты операции	M
POS_TRANS_ID	Идентификатор транзакции	M
POS_TRANS_RECEIPT	Номер чека	M
POS_TRANS_CODE	Код транзакции (Reference Number)	M
POS_TRANS_APPROVAL	Код авторизации	M
POS_DATE_TIME	Дата и время операции	M
POS_CARD_PAN	Номер карты клиента	M
POS_CARD_PAN_SHA256	SHA256 номера карты клиента	M
POS_CARD_EXPIRY	Срок годности карты	M
POS_MERCHANT_ID	Идентификатор торговца	M
POS_TERMINAL_ID	Идентификатор терминала	M
POS_CARD_HOLDER	Имя владельца карты	O
POS_PRINT	Текст для печати	O

POS_TRANS_MSGCODE	Код оригинального сообщения	WAY4
POS_CARD_ID_NUMBER	Код проверки карты	WAY4

Примечание:
Для топлива (ОККО) в ответе *RESP_CONFIRM* следующие параметры **не** выдаются:
POS_CARD_PAN_SHA256, POS_MERCHANT_ID, POS_TERMINAL_ID

2.10.4 RESP_CONFIRM - Подтверждение успешного выполнения запрошенной операции (ACTION_CLOSE_DAY, ACTION_REPORT, ACTION_COPY_RECEIPT, ACTION_COPY_CLOSE_DAY)

Имя параметра	Описание	Признак наличия
POS_PRINT	Текст для печати	O

2.10.5 RESP_DECLINE - Запрошенная операция отклонена терминалом или процессинговым центром

Имя параметра	Описание	Признак наличия
POS_AMOUNT	Сумма операции	M
POS_TIPS	Сумма чаевых	O
POS_CURRENCY	Код валюты операции	M
POS_TRANS_ID	Идентификатор транзакции	M
POS_TRANS_RECEIPT	Номер чека	M
POS_TRANS_STATUS	Сообщение об отказе	M
POS_DATE_TIME	Дата и время операции	M
POS_CARD_PAN	Номер карты клиента	M
POS_CARD_PAN_SHA256	SHA256 номера карты клиента	M
POS_CARD_EXPIRY	Срок годности карты	M
POS_CARD_HOLDER	Имя владельца карты	M
POS_STATUS	Код отказа	M
POS_MERCHANT_ID	Идентификатор торговца	M
POS_TERMINAL_ID	Идентификатор терминала	M
POS_PRINT	Текст для печати	O

Примечание:
Для топлива (ОККО) в ответе *RESP_DECLINE* следующие параметры **не** выдаются:
POS_CARD_PAN_SHA256, POS_MERCHANT_ID, POS_TERMINAL_ID

2.10.6 RESP_MESSAGE - Уведомление о текущем этапе выполнения ПОС терминалом запрошенной операции

Имя параметра	Описание	Признак наличия
POS_MSG_TITLE	Заголовок сообщения	M
POS_MSG_BODY	Тело сообщения	M
POS_MSG_BREAK	Признак прерывания	M

2.10.7 RESP_IDENTIFIER - Уведомление о присвоенном идентификаторе транзакции

Имя параметра	Описание	Признак наличия

POS_TRANS_ID	Идентификатор транзакции	M
POS_MSG_BREAK	Признак прерывания	M
POS_CARD_PAN	Номер карты клиента	O
POS_CARD_PAN_SHA256	SHA256 номера карты клиента	M
POS_CARD_EXPIRY	Срок годности карты	O
POS_CARD_LOYALTY_CODE	Код лояльности карты	O

2.10.8 RESP_CONFIRM - Подтверждение успешного выполнения чтения карты (ACTION_READ_CARD), если считанная карта – платежная

Имя параметра	Описание	Признак наличия
POS_CARD_PAN	Номер карты клиента	M
POS_CARD_PAN_SHA256	SHA256 номера карты клиента	M
POS_CARD_EXPIRY	Срок годности карты	M
POS_CARD_HOLDER	Имя владельца карты	M
POS_CARD_PAYMENT	Признак платежной карты (1)	M

2.10.9 RESP_CONFIRM - Подтверждение успешного выполнения чтения карты (ACTION_READ_CARD), если считанная карта – дисконтная

Имя параметра	Описание	Признак наличия
POS_CARD_TRACK1	Track1	M
POS_CARD_TRACK2	Track2	M
POS_CARD_TRACK3	Track3	M
POS_CARD_PAYMENT	Признак платежной карты (0)	M

2.10.10 RESP_CONFIRM - Подтверждение успешного выполнения ввода кода проверки дисконтной карты (ACTION_GET_VERIF_CODE)

Имя параметра	Описание	Признак наличия
POS_CARD_VERIF_CODE	Код проверки карты	M

2.10.11 RESP_CONFIRM - Подтверждение успешного выполнения смены кода проверки дисконтной карты (ACTION_CHANGE_VERIF_CODE)

Имя параметра	Описание	Признак наличия
POS_CARD_VERIF_CODE	Старый код проверки карты	M
POS_CARD_NEW_VERIF_CODE	Новый код проверки карты	M

2.10.12 RESP_CONFIRM - Подтверждение успешного выполнения запрошенной операции (ACTION_PRINT_DATA и ACTION_SHOW_QR_CODE).

Без параметров.

3 МЕХАНИЗМ ВОССТАНОВЛЕНИЯ ПОСЛЕ СБОЕВ

При подготовке к проведению финансовой операции, сразу после считывания карты, терминал передает на кассу сообщение RESP_IDENTIFIER. В этом сообщении передается параметр POS_TRANS_ID – уникальный идентификатор транзакции, присвоенный терминалом. Если сообщение RESP_IDENTIFIER не будет успешно доставлено на кассу, терминал финансовую операцию с процессингом не начнет и вернет на кассу RESP_BREAK.

При успешном завершении финансовой операции, терминал возвращает на кассу сообщение RESP_CONFIRM, и записывает всю информацию во внутренний журнал транзакций. В журнале транзакций записываются только успешные операции. При выполнении операции «Закрытие дня» (ACTION_CLOSE_DAY) – журнал транзакций очищается.

Таким образом, в течении рабочего дня, в журнале транзакций хранятся все успешные финансовые операции. И при возникновении сбоя, касса может запросить статус любой предыдущей финансовой операции. Для этого касса должна открыть новую сессию (pos_open()) и послать на терминал запрос ACTION_STATUS с параметром POS_TRANS_ID транзакции, состояние которой нужно определить. Если транзакция с указанным идентификатором будет найдена в журнале транзакций терминала, на кассу будет возвращено сообщение RESP_CONFIRM со всеми параметрами успешно завершенной операции. Если транзакции с таким идентификатором нет – терминал вернет RESP_BREAK.

Примечание: успешная операция отмены (ACTION_REVERSAL) замещает собой финансовую операцию, которую она отменяет. И статус операции (ACTION_STATUS) можно будет определить только для операции отмены.

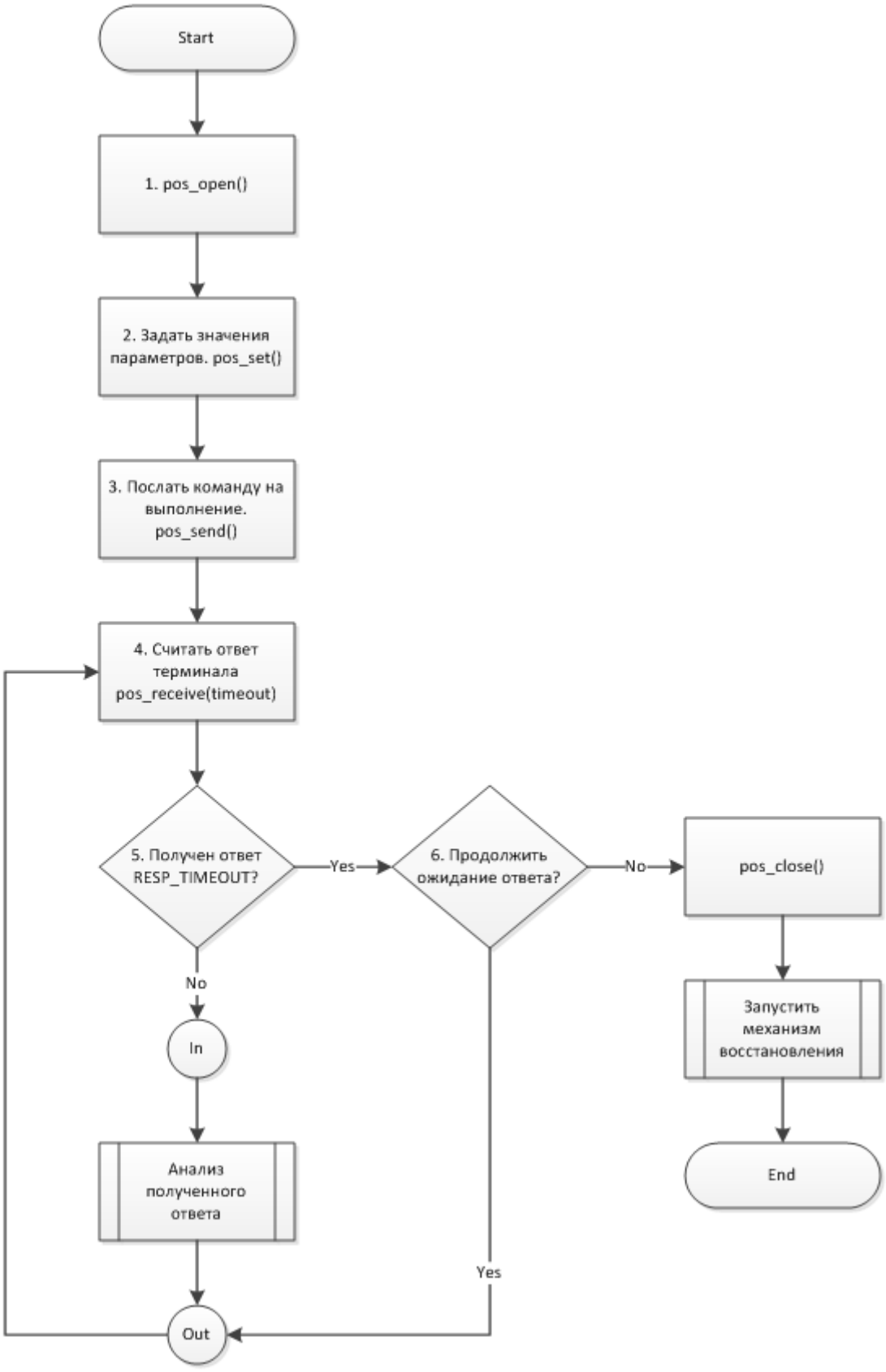
4 АЛГОРИТМ УПРАВЛЕНИЯ РАЗМЕРОМ СКИДКИ ПРИ ПРОВЕДЕНИИ ОПЕРАЦИИ «ОПЛАТА»

1. Касса передает на терминал запрос на проведение операции «Оплата» с параметром POS_REVISION_SIGN равным 1 и суммой (POS_ AMOUNT) без скидки.
2. Терминал предлагает кассиру считать карточку и высвечивает полученную сумму без скидки.
3. Получив номер карты и срок годности, терминал передает на кассу сообщение RESP_IDENTIFIER с параметрами POS_TRANS_ID, POS_MSG_BREAK, POS_CARD_PAN, POS_CARD_EXPIRY, POS_CARD_LOYALTY_CODE.
4. Терминал высвечивает сообщение: «Ожидание информации» и ожидает запрос от кассы.
5. Касса, на основании номера карты, рассчитывает скидку и передает на терминал запрос ACTION_REVISION с параметром POS_ AMOUNT – новая сумма транзакции со скидкой.
6. Терминал проводит финансовую операцию.
7. Терминал возвращает ответ RESP_CONFIRM со следующими параметрами: POS_AMOUNT – авторизованная сумма операции со скидкой, POS_ORIG_AMOUNT – сумма операции без скидки (возвращается только в том случае, если сумма была изменена).

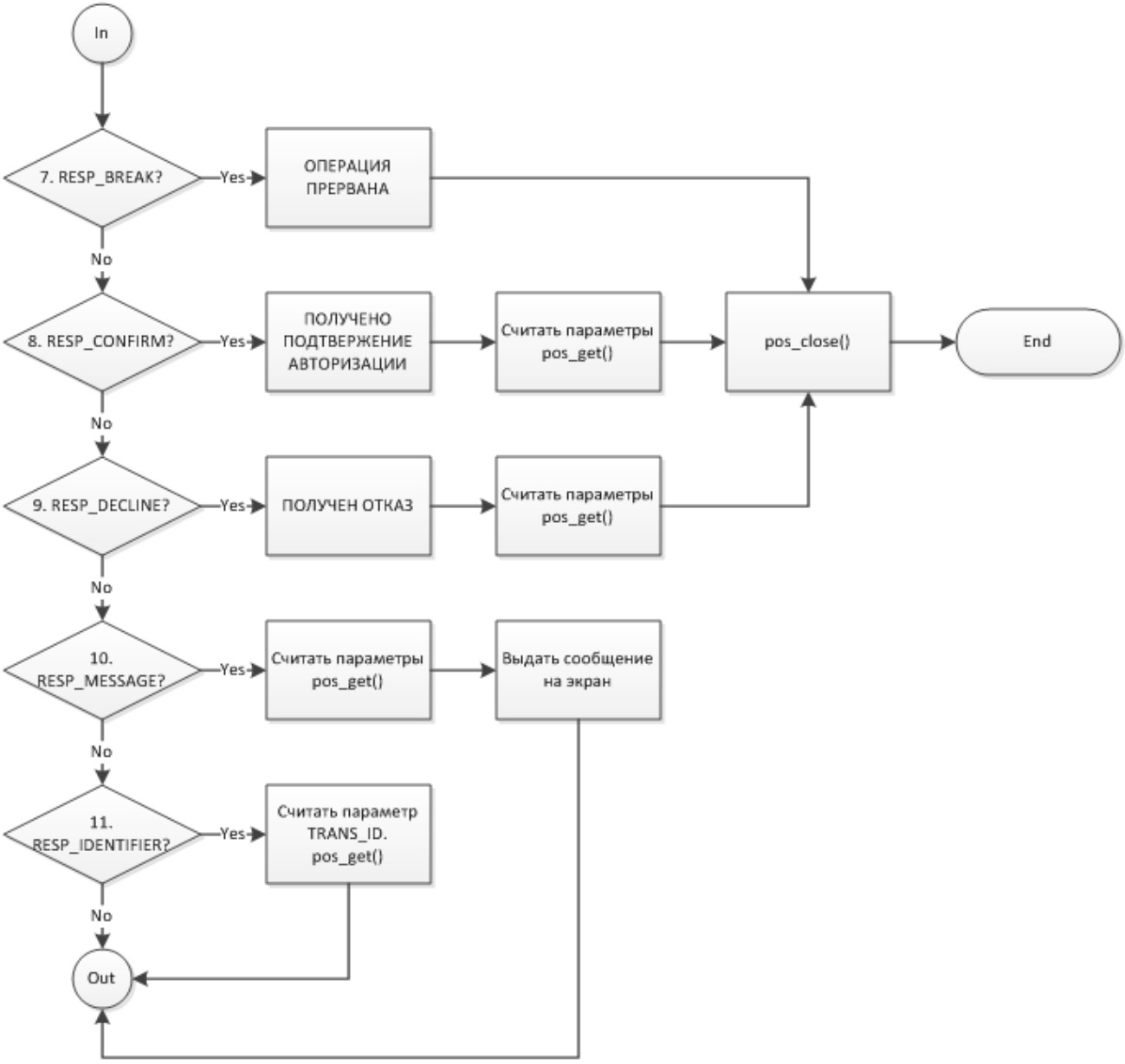
Примечание: Сумма со скидкой не должна быть больше чем оригинальная сумма операции, иначе терминал вернет RESP_BREAK.

5 АЛГОРИТМ ИСПОЛЬЗОВАНИЯ БИБЛИОТЕКИ

1. Вызвать функцию pos_open()
2. Задать значения каждого из параметров, необходимых для выполнения заданной операции (см. главу 2.7) при помощи функции pos_set()
3. Послать команду на выполнение заданной операции при помощи функции pos_send()
4. Считать ответ от терминала при помощи функции pos_receive() с заданным значением таймаута, установленным в зависимости от типа коммуникаций (Ориентировочные рекомендуемые значения составляют: 1 мин — для Ethernet, 2-3 мин — для модема, 4-5 мин — для GPRS)
5. Если получен ответ RESP_TIMEOUT, то перейти к пункту 6.
6. Если нужно продолжить ожидание ответа – перейти к пункту 4, иначе вызвать pos_close(), после чего запустить механизм восстановления (см. главу 3), и закончить.
7. Если получен ответ RESP_BREAK (операция прервана) вызвать функцию pos_close() и закончить.
8. Если получен ответ RESP_CONFIRM (получено подтверждение авторизации) считать параметры (см. главу 2.8) при помощи функции pos_get(), вызвать функцию pos_close() и закончить.
9. Если получен ответ RESP_DECLINE (получен отказ) считать параметры при помощи функции pos_get(), вызвать функцию pos_close() и закончить.
10. Если получен ответ RESP_MESSAGE, считать параметры при помощи функции pos_get(), выдать полученное сообщение на экран и перейти к п.4
11. Если получен ответ RESP_IDENTIFIER считать параметр TRANS_ID при помощи функции pos_get() и перейти к п.4
12. Перейти к п.4


Start
1. pos_open()
2. Задать значения параметров. pos_set()
3. Послать команду на выполнение. pos_send()
4. Считать ответ терминала pos_receive(timeout)
5. Получен ответ RESP_TIMEOUT?
Yes
6. Продолжить ожидание ответа?
No
pos_close()
No
In
Запустить механизм восстановления
Анализ полученного ответа
End
Yes
Out

Анализ полученного ответа



6 ОСОБЕННОСТИ РАЗЛИЧНЫХ РЕАЛИЗАЦИЙ

В данном разделе описываются отличительные особенности некоторых реализаций интерфейса POSAPI и специфика их использования.

6.1 Особенности интеграции с Delphi

Оптимальным выбором для интеграции ПОС терминала с программным продуктом, разработанным в среде Delphi, является динамическая библиотека Windows. Однако, ее непосредственное подключение является проблематичным вследствие неполной поддержки механизма динамических библиотек данной средой разработки.

Решением является специальный стыковочный модуль uPosApi, реализованный на Object Pascal, который может поставляться вместе с динамической библиотекой Windows. Интерфейс стыковочного модуля полностью идентичен интерфейсу самой библиотеки, описанному в данном документе.

Синтаксис:

```
function pos_open(var handle_p: POS_HANDLE; const name: PChar;
              const log: PChar = Nil): ByteBool; cdecl; external
              posapi;
function pos_close(var handle_p: POS_HANDLE): ByteBool; cdecl;
              external posapi;

function pos_send(handle: POS_HANDLE; action: Integer): ByteBool;
              cdecl; external posapi;
function pos_receive(handle: POS_HANDLE; timeout: Integer):
              Integer; cdecl; external posapi;

function pos_set(handle: POS_HANDLE; const param: PChar; const val:
              PChar): ByteBool; cdecl; external posapi;
function pos_get(handle: POS_HANDLE; const param: PChar; val:
              PChar; val_size: Integer): ByteBool; cdecl; external
              posapi;

function pos_get_first(handle: POS_HANDLE; param: PChar;
              param_size: Integer; val: PChar; val_size: Integer):
              ByteBool; cdecl; external posapi;
function pos_get_next(handle: POS_HANDLE; param: PChar; param_size:
              Integer; val: PChar; val_size: Integer): ByteBool;
              cdecl; external posapi;
```

Пример использования модуля uPosApi приведен ниже, в соответствующем разделе.

6.2 Особенности интеграции с использованием COM

Реализация интерфейса в виде объекта COM позволяет интегрировать ПОС терминал с различными языками сценариев и виртуальными средами выполнения, поддерживающими данную технологию. Однако, вследствие особых требований, выдвигаемых к таким объектам, интерфейс в данной реализации имеет ряд отличий по сравнению с описанием, приведенным выше.

Следует обратить внимание на следующие специфические особенности данной реализации:

- Все методы интерфейса реализованы в виде методов COM-объекта.
- Все константы интерфейса реализованы в виде свойств COM-объекта.

- Параметр дескриптора сессии отсутствует во всех методах, т.к. соответствующая информация хранится в самом COM-объекте.
- Все строковые параметры методов имеют тип VARIANT. Поскольку при этом осуществляется автоматическое управление памятью, все параметры, в которых задается максимальная длина данных, отсутствуют.
- Возвращаемые значения всех методов являются целочисленными. Если выше возвращаемое значение метода описано как логическое, в данной реализации он возвращает 1 в случае успешного и 0 в случае неуспешного завершения.
- Для создания данного COM-объекта следует воспользоваться следующим идентификатором: `"CardPay.PosApi"`.

Синтаксис:

```
long pos_open(VARIANTARG *name, VARIANTARG *log)
long pos_close()

long pos_send(long action)
long pos_receive(long timeout)

long pos_set(VARIANTARG *param, VARIANTARG *val)
long pos_get(VARIANTARG *param, VARIANTARG *val)
long pos_get_first(VARIANTARG *param, VARIANTARG *val)
long pos_get_next(VARIANTARG *param, VARIANTARG *val)
```

Замечание:

Перед использованием данной библиотеки следует зарегистрировать ее в операционной системе. В частности, это можно сделать в консоли Windows с помощью следующей команды:

```
regsvr32.exe posapi2.dll
```

Пример использования данной реализации интерфейса приведен в соответствующем разделе данного документа.

6.3 Особенности интеграции с JAVA

Для подключения платформенно-зависимых библиотек к Java необходима их доработка в соответствии с требованиями спецификации JNI (Java Native Interface). Чтобы облегчить процесс интеграции, данная работа уже выполнена поставщиком интерфейса.

Реализация интерфейса, предназначенная для интеграции с Java, включает в себя две компоненты:

- Специальный вариант платформенно-зависимой библиотеки, соответствующий требованиям JNI.
- Стыковочный модуль на Java, позволяющий пользовательскому приложению обращаться к данной библиотеке.

Из-за ряда фундаментальных особенностей языка Java интерфейс в данной реализации имеет ряд отличий от общего описания, приведенного выше.

- Интерфейс представлен в виде Java-класса `PosApi`, а все его методы и константы реализованы в виде соответствующих членов данного класса.
- Именование методов интерфейса заменено принятым в Java. Например, метод `pos_open()` в данной реализации будет именоваться `posOpen()`.
- Параметр дескриптора сессии отсутствует во всех методах, т.к. соответствующая информация хранится в экземпляре класса.

- Все строковые параметры методов имеют тип `String`. Поскольку при этом осуществляется автоматическое управление памятью, все параметры, в которых задается максимальная длина данных, отсутствуют.
- Неизменяемость строк в Java вынудила скорректировать часть интерфейса, ответственную за считывание параметров. В данной реализации метод `posGet()` получает единственный аргумент (имя параметра) и возвращает его значение (в случае неудачи возвращает `null`), а пара методов `pos_get_first()` и `pos_get_next()` заменена единственным методом `posGetAll()`, не требующим никаких аргументов, и в случае успеха возвращающим массив, содержащий все пары имен и значений параметров.

Синтаксис:

```
package com.cardpay.pos;

public class PosApi {

  public native boolean posOpen(String name);
  public native boolean posClose();

  public native boolean posSend(int action);
  public native int posReceive(int timeout);

  public native boolean posSet(String param, String val);
  public native String posGet(String param);

  public native Param[] posGetAll();

  public static class Param {
    public String name;
    public String value;
  }
}
```

Пример использования данной реализации интерфейса приведен в соответствующем разделе данного документа.

6.4 Особенности использования компоненты для .NET

Работать с данной библиотекой можно из любого языка, поддерживаемого платформой .NET. Однако приведенное здесь описание и примеры относятся к использованию в языке C#.

Из-за ряда фундаментальных особенностей языка C# интерфейс в данной реализации имеет ряд отличий от общего описания, приведенного выше.

- Интерфейс представлен в виде C# класса `PosApi`, а все его методы и константы реализованы в виде соответствующих членов данного класса.
- Именование методов интерфейса заменено принятым в C#. Например, метод `pos_open()` в данной реализации будет именоваться `posOpen()`.
- Параметр дескриптора сессии отсутствует во всех методах, т.к. соответствующая информация хранится в экземпляре класса.
- Все строковые параметры методов имеют тип `String`. Поскольку при этом осуществляется автоматическое управление памятью, все параметры, в которых задается максимальная длина данных, отсутствуют.

Неизменяемость строк в C# вынудила скорректировать часть интерфейса, ответственную за считывание параметров. В данной реализации метод `posGet()` получает

единственный аргумент (имя параметра) и возвращает его значение (в случае неудачи возвращает `nullptr`), а пара методов `pos_get_first()` и `pos_get_next()` заменена единственным методом `posGetAll()`, не требующим никаких аргументов, и в случае успеха возвращающим коллекцию, содержащий все пары имен и значений параметров в структуре `Param`.

Класс `PosApi` содержит следующий набор методов:

```
bool posOpen(String^ name, String^ log);
bool posClose();

bool posSend(Action action);
Response posReceive(int timeout);

bool posSet(String^ param, String^ val);
String^ posGet(String^ param);
Parameters^ posGetAll();
```

Классы `Action` и `Response` определены как перечисления в классе `PosApi`. Например, для того, чтобы указать операцию оплаты (`ACTION_PAYMENT`) используется синтаксис `PosApi.Action.PAYMENT`, а для спецификации кода ответа `RESP_CONFIRM` указывается `PosApi.Response.CONFIRM`. В классе `PosApi` также определены строковые константы для всех параметров. Например, параметр `POS_CURRENCY` специфицируется как `PosApi.POS_CURRENCY`.

Функция `posGetAll` возвращает коллекцию `Parameters`, допускающую использование в цикле `foreach`. Элементами этой коллекции являются структуры типа `Param`, содержащие пару строк имя-значение.

Пример использования данной реализации интерфейса приведен в соответствующем разделе данного документа.

7 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

В данном разделе приведены примеры исходного кода простых приложений, демонстрирующих использование различных реализаций данного интерфейса из различных языков программирования. Все тестовые приложения выполнены по единому алгоритму:

1. Открыть сеанс связи с терминалом, подключенным к порту COM1.
2. Установить параметры – сумму и код валюты операции – соответствующие 1.00 UAH.
3. Отослать запрос на оплату товаров или услуг – *`ACTION_PAYMENT`*.
4. Ожидать в цикле ответ от терминала.
5. Проанализировать ответ и вывести на экран имена и значения всех возвращенных параметров.
6. Завершить сеанс связи с терминалом.

Замечание:

Данные примеры не претендуют на функциональную полноту и могут использоваться исключительно как вспомогательный материал для разработчика, поясняющий принципы работы интерфейса POSAPI.

7.1 Пример интеграции с приложением C / C++

```
#include <stdio.h>
#include "pos.h"

void print_response(POS_HANDLE handle)
```

```
{
  char par[20];
  char val[10240];

  if ( pos_get_first(handle, par, sizeof(par), val, sizeof(val)) )
  {
    do {
      printf("%s = \"%s\"\n", par, val);
    } while ( pos_get_next(handle, par, sizeof(par), val, sizeof(val)) );
  }
  printf("\n");
}

int main()
{
  POS_HANDLE handle;
  int timeout = 180000; // 3 min

  if ( !pos_open(&handle, "COM1", 0) )
     return -1;

  pos_set(handle, POS_AMOUNT, "100");   // 1.00
  pos_set(handle, POS_CURRENCY, "980"); // UAH

  if ( !pos_send(handle, ACTION_PAYMENT) )
  {
    pos_close(&handle);
    return -2;
  }

  bool receive_loop = true;

  while ( receive_loop )
  {
    int resp = pos_receive(handle, timeout);
    switch(resp)
    {
    case RESP_TIMEOUT:
      printf("TIMEOUT!\n");
      receive_loop = false;
      break;

    case RESP_BREAK:
      printf("BREAK!\n");
      receive_loop = false;
      break;

    case RESP_CONFIRM:
      printf("CONFIRMED\n");
      print_response(handle);
      receive_loop = false;
      break;

    case RESP_DECLINE:
      printf("DECLINED\n");
      print_response(handle);
      receive_loop = false;
      break;

    case RESP_MESSAGE:
      printf("MESSAGE\n");
      print_response(handle);
      break;
```

```
    case RESP_IDENTIFIER:
      printf("IDENTIFIER\n");
      print_response(handle);
      break;
    }
  }

  pos_close(&handle);
  return 0;
}
```

7.2 Пример интеграции с Delphi

```
program Test;

{$APPTYPE CONSOLE}

uses
  uPosAPI, Windows, SysUtils;

procedure PrintResponse(handle: POS_HANDLE);
var
  par: array[0..20] of char;
  val: array[0..10240] of char;
  res: ByteBool;
begin
  if (pos_get_first(handle, @par[0], sizeof(par), @val[0], sizeof(val)) = true)
  then begin
    repeat
      CharToOem(val, val); { Decode ANS->OEM before console output. }
      Writeln(Format('%s = "%s"', [string(par), string(val)]));
      res := pos_get_next(handle, @par[0], sizeof(par), @val[0], sizeof(val));
    until (res = false);
  end;
  Writeln('');
end;

var
  handle: POS_HANDLE;
  loop: ByteBool;
  resp: integer;
  timeout: integer;

begin
  handle := POS_NONE;

  If not pos_open(handle, PChar('COM1')) then begin
    Writeln('Can not open port!');
    Exit;
  end;

  pos_set(handle, POS_AMOUNT, PChar('100'));     { 1.00 }
  pos_set(handle, POS_CURRENCY, PChar('980'));   { UAH  }

  if not pos_send(handle, ACTION_PAYMENT) then begin
    Writeln('Can not send request!');
    Exit;
  end;

  loop := true;
  timeout := 180000; { 3 min }
```

```
  while (loop) do begin
    resp := pos_receive(handle, timeout);

    case (resp) of
    RESP_TIMEOUT:
      begin
        Writeln('TIMEOUT');
      end;
    RESP_BREAK:
      begin
        Writeln('BREAK!');
        loop := false;
      end;
    RESP CONFIRM:
      begin
        Writeln('CONFIRMED');
        PrintResponse(handle);
        loop := false;
      end;
    RESP_DECLINE:
      begin
        Writeln('DECLINED');
        PrintResponse(handle);
        loop := false;
      end;
    RESP_MESSAGE:
      begin
        Writeln('MESSAGE');
        PrintResponse(handle);
      end;
    RESP_IDENTIFIER:
      begin
        Writeln('IDENTIFIER');
        PrintResponse(handle);
      end;
    end;
  end;
end.
```

7.3 Пример интеграции с VBScript

```
Sub DispResult(info, obj)
      Dim msg, result, param, val
      msg = info

      If obj.pos_get_first(param, val) Then
            Do
                  msg = msg + "; " + param + "='" + val + "'"
            Loop While obj.pos_get_next(param, val)
      End If

      MsgBox msg, 64
End Sub

Dim obj
Set obj = CreateObject("CardPay.PosApi")

If IsEmpty(Obj) Or IsNull(Obj) Then
      MsgBox "Can't access CardPay.PosApi!", 16
      WScript.Quit
End If

Dim result
```

```
result = obj.pos_open("COM1", "")
If result = 0 Then
      MsgBox "Can't open POS session!", 16
      WScript.Quit
End If

obj.pos_set(obj.POS_AMOUNT, "100")
obj.pos_set(obj.POS_CURRENCY, "980")

result = obj.pos_send(obj.ACTION_PAYMENT)
If result = 0 Then
      MsgBox "Can't send request to POS!", 16
      obj.pos_close()
      WScript.Quit
End If

Dim recvLoop
recvLoop = True

Do While recvLoop
      Dim resp
      resp = obj.pos_receive(180000)

      Select Case resp
      Case obj.RESP_TIMEOUT
            Call DispResult("TIMEOUT", obj)
      Case obj.RESP_BREAK
            recvLoop = False
      Case obj.RESP_CONFIRM
            Call DispResult("CONFIRMED", obj)
            recvLoop = False
      Case obj.RESP_DECLINE
            Call DispResult("DECLINED", obj)
            recvLoop = False
      Case obj.RESP_MESSAGE
            Call DispResult("MESSAGE", obj)
      Case obj.RESP_IDENTIFIER
            Call DispResult("IDENTIFIER", obj)
      End Select
Loop

obj.pos_close()

MsgBox "Done!"
```

7.4 Пример интеграции с JAVA

```
import com.cardpay.pos.PosApi;

public class Test {

  public static void main(String[] args) {
    PosApi obj = new PosApi();

    if ( !obj.posOpen( "COM1" ) ) {
      System.out.println("Can't open port!");
      return;
    }

    obj.posSet(PosApi.POS_AMOUNT, "100");   // 1.00
    obj.posSet(PosApi.POS_CURRENCY, "980"); // UAH

    if ( !obj.posSend(PosApi.ACTION_PAYMENT) )
```

```
    {
      System.out.println("Can't send a request!");
      obj.posClose();
      return;
    }

    boolean receiveLoop = true;
    while ( receiveLoop )
    {
      PosApi.Param[] param = null;
      int resp = obj.posReceive(180000); // 3 min

      switch(resp) {

      case PosApi.RESP_TIMEOUT:
        System.out.println("TIMEOUT!");
        break;

      case PosApi.RESP_BREAK:
        System.out.println("BREAK!");
        receiveLoop = false;
        break;

      case PosApi.RESP_CONFIRM:
        System.out.println("CONFIRMED");
        param = obj.posGetAll();
        receiveLoop = false;
        break;

      case PosApi.RESP_DECLINE:
        System.out.println("DECLINED");
        param = obj.posGetAll();
        receiveLoop = false;
        break;

      case PosApi.RESP_MESSAGE:
        System.out.println("MESSAGE");
        param = obj.posGetAll();
        break;

      case PosApi.RESP_IDENTIFIER:
        System.out.println("IDENTIFIER");
        param = obj.posGetAll();
        break;
      }

      if ( param != null ) {
        for(int i = 0; i < param.length; i++)
          System.out.printf("%s=%s\n", param[i].name, param[i].value);
        System.out.println("");
      }
    }

    obj.posClose();
  }
}
```

7.5 Пример интеграции с C#

```
using System;
using CardPay;
```

```
namespace CardPay.PosApiTest
{
    class Program
    {
        static PosApi Api;

        static void Main(string[] args)
        {
            Api = new PosApi();

            bool result = Api.posOpen("COM1", ""); // port name
            if (!result)
            {
                Console.WriteLine("Can't open port.");
                return;
            }

            Api.posSet(PosApi.POS_AMOUNT, "100");   // 1.00
            Api.posSet(PosApi.POS_CURRENCY, "980"); // UAH

            result = Api.posSend(PosApi.Action.PAYMENT);
            if (!result)
            {
                Console.WriteLine("Can't send a request to POS.");
                Api.posClose();
                return;
            }

            PosApi.Response response;
            do
            {
                response = Api.posReceive(30000);

                if (response > 0 && response != PosApi.Response.TIMEOUT)
                {
                    Console.WriteLine("response = {0:X}", response);
                    foreach (Param param in Api.posGetAll())
                        Console.WriteLine("{0} = \"{1}\"", param.name,
                                                           param.value);
                    Console.WriteLine("\n");
                }
                else if (response == 0)
                    Console.WriteLine("ERROR: Zero response code!\n");

            } while (response == PosApi.Response.MESSAGE ||
                     response == PosApi.Response.TIMEOUT ||
                     response == PosApi.Response.IDENTIFIER);

            result = Api.posClose();

            return;
        }
    }
}
```