Протокол взаимодействия с терминалом оплаты AQSI Cube

aQsi

Version 1.1, 2024

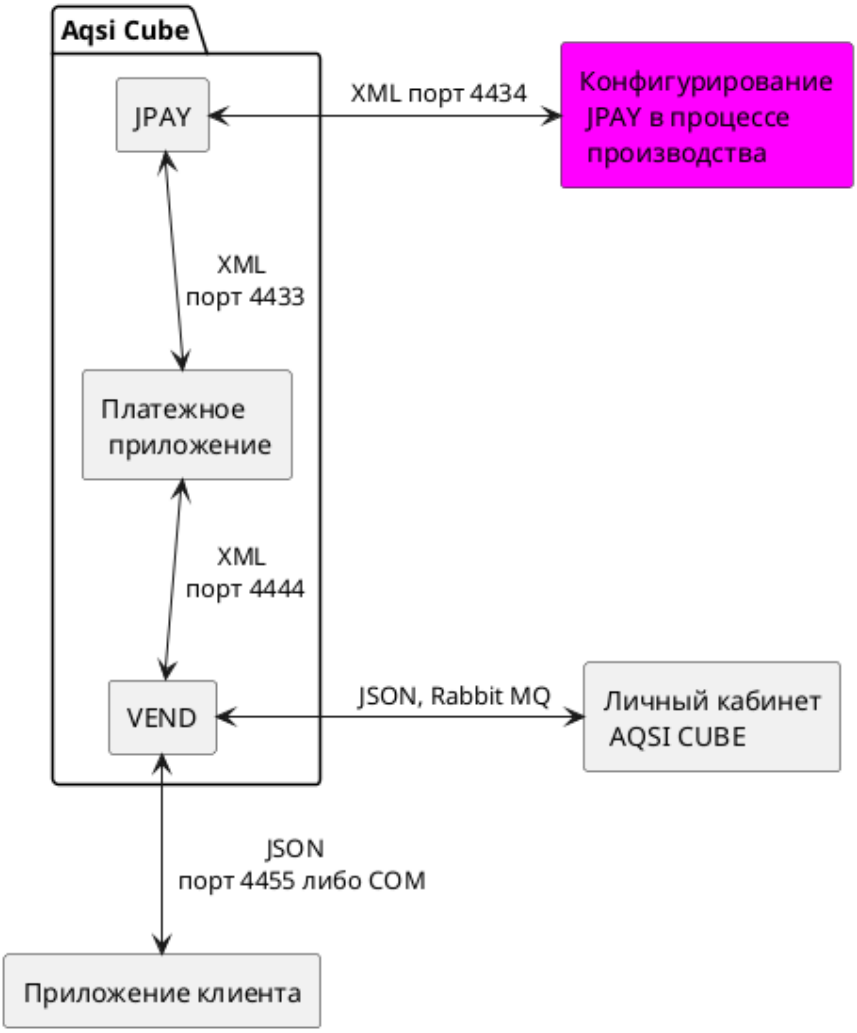
Содержание

1. Платёжный функционал на устройствах AQSI CUBE . . . 1
2. Наборы команд, поддерживаемые в приложениях JPAY, ПП, VEND . . . 3
3. Взаимодействие с программой JPAY . . . 9
   3.1. Активация JPAY . . . 9
4. Платёжное Приложение . . . 11
   4.1. Обработка команд входящих от клиентов. . . . 11
   4.2. Особенности обработки команд, требующих авторизации . . . 13
   4.3. Особенности обработки команд с промежуточными сообщениями . . . 15
   4.4. Специальная обработка команд JPAY для транзакций . . . 16
   4.5. Платёжное приложение и активации JPAY . . . 17
   4.6. Обработка ситуаций «обрыва» соединений . . . 17
5. Программа VEND . . . 19
   5.1. updateconfiguration – загрузка конфигурации с сервера . . . 19
   5.2. loadmasterkey – загрузка мастер-ключей . . . 19
   5.3. loadworkkeys – загрузка рабочих ключей . . . 20
   5.4. testconnection – проверка связи с сервером . . . 20
   5.5. getparameters – сведения о приложении и терминале . . . 20
   5.6. transaction – транзакции . . . 22
   5.7. report – отчет . . . 28
   5.8. settlement– сверка итогов . . . 34
   5.9. clear – очистка журнала . . . 35
   5.10. runreset – переинициализация . . . 36
   5.11. keep-alive – сообщение об увеличении времени ожидания . . . 36
   5.12. fiscal-receipt – фискализация чека . . . 37
   5.13. show-qr-code – показать QR-код на экране терминала . . . 43
   5.14. show-customer-screen – показать сообщение отправленное пользовательским приложением на экране терминала . . . 45
   5.15. get-work-state – запросить текущее состояние работы терминала . . . 47
   5.16. status - статус выполнения команды/операции . . . 47
6. Настройка работы с картами mifare в режиме POS . . . 49
   6.1. mifare-reader-enable – включение / отключение считывателя карт mifare . . . 49
   6.2. mifare-reader-get-last-read - получение результатов последнего считывания карты mifare . . . 49
   6.3. mifare-reader-clear-last-read - удаление данных об уже считанной карте (индикатор того, что данные были обработаны) . . . 50
7. Формат транспортного пакета . . . 52
   7.1. Взаимодействие по сетевым протоколам . . . 52
   7.2. Взаимодействие через последовательный интерфейс . . . 52

8. Параметры, необходимые для фискализации . . . 57
8.1. fiscalParams . . . 57
8.2. Содержимое чека (content) . . . 57
8.3. Позиции чека (positions) . . . 57
8.4. Параметры закрытия чека (checkClose) . . . 57
8.5. Оплаты (payments) . . . 58
8.6. Системы налогообложения . . . 58
8.7. Операционный режим . . . 58
8.8. Типы чека . . . 58
8.9. Типы НДС . . . 59
8.10. Способы расчёта . . . 59
8.11. Типы предмета расчёта . . . 59
8.12. Типы оплаты . . . 60

1. Платёжный функционал на устройствах AQSI CUBE

Схема взаимодействия приложений AQSI CUBE между собой и с внешними устройствами.



Приложение клиента может взаимодействовать с программой VEND либо через сетевой протокол либо через COM-порт.

Программа	Описание
JPAY	Данное приложение является стандартным эквайринговым ядром AQSI. Взаимодействие с этим приложением осуществляется через сетевой порт 4433 путём пересылки сообщений в формате XML (при локальной работе взаимодействие осуществляется через localhost). В приложении не предусматривается никаких диалоговых окон для взаимодействия с пользователем кроме окна ввода PIN. В процессе производства кассы предусматривается отдельный набор команд для загрузки сертификатов (сертификат клиента конфигурации, сертификат сервера конфигурации, сертификат для получения мастер-ключей и сертификат для взаимодействия с сервером процессинга). Все эти действия по загрузке сертификатов выполняются через специальный отдельный порт 4434.
Платёжное приложение (ПП)	Это специализированная программа, которая «дополняет» функционал JPAY в части пользовательстких интерфейсов. ПП (также как и JPAY) получает команды на выполнение эквайринговых операций в виде сообщений в формате XML. Формат команд ПП и JPAY близок, но не полностью идентичен в силу того, что ПП «берёт на себя» часть технических коммуникаций.

Программа	Описание
VEND	Это специализированная программа, которая выполняет следующие функции: конвертирует информацию из формата JPAY в формат транспортного протокола AQSI принимает команды от приложения клиента на выполнение эквайринговых операций выполняет эквайринговые операции при помощи Платёжного Приложения возвращает результат в приложение клиента дублирует информацию о платеже в Личный Кабинет AQSI CUBE (на сервер) в случае, если предусматривается формирование фискального чека через облачный сервис Оранж-Дата, приложение VEND получает фискальные данные из ЛК (с сервера) и осуществляет отображение QR-кода на экране AQSI CUBE в случае, если предусматривается формирование фискального чека при помощи встроенного фискального регистратора payonline, приложение VEND осуществляет взаимодействие с устройством, формирование фискального чека, передачу фискальных данных в ЛК вместе с чеком и отображение QR-кода на экране взаимодействие с шиной вендингового автомата

2. Наборы команд, поддерживаемые в приложениях JPAY, ПП, VEND

Команда / сообщение	Описание	JPAY	ПП	VEND
login	авторизация в режим администратора и получение токена	+	- авторизация должна выполняться автоматически в процессе выполнения других команд	- авторизация должна выполняться автоматически в процессе выполнения других команд

Команда / сообщение	Описание	JPAY	ПП	VEND
logout	выход из режима администратора	+	- авторизация должна выполняться автоматически в процессе выполнения других команд	- авторизация должна выполняться автоматически в процессе выполнения других команд
changepassword	смена пароля администратора	+	- пароль следует прописать как константу в настройках	- пароль следует прописать как константу в настройках
resetpassword	сброс пароля	+	- пароль следует прописать как константу в настройках	- пароль следует прописать как константу в настройках
updateconfiguration	проверка и загрузка обновлений конфигурации с сервера конфигураций	+	+	+
loadworkkeys	загрузка рабочих ключей	+	+	+
loadmasterkeys	загрузка мастер ключей	+	+	+
testconnection	проверка связи с сервером авторизации	+	+	+
getparameters	сведения о приложении и терминале	+	+	+
transaction	транзакция (purchase, refund, purchase-with-cashback, void)	+	+	+

Команда / сообщение	Описание	JPAY	ПП	VEND
keep-alive (вспомогательное сообщение при выполнении транзакции)	промежуточное сообщение от JPAY при выполнении транзакции – предписание ожидать результата обработки от сервера	+	+ сообщения keep-alive должны обрабатываться внутри ПП, но и сам ПП должен проинформировать вышестоящее приложение об увеличении времени ожидания. При этом программа ПП (в отличие от JPAY) не требует обязательного ответа на сообщение keep-alive. Ответить можно, если необходимо принудительно прервать транзакцию.	+ сообщения keep-alive должны обрабатываться внутри ПП, т.е. VEND просто «транслирует» эти сообщения чтобы проинформировать вызывающее приложение об увеличении времени ожидания. При этом программа VEND (в отличие от JPAY) не требует обязательного ответа на сообщение keep-alive. Ответить можно, если необходимо принудительно прервать транзакцию.
display (вспомогательное сообщение при выполнении транзакции)	промежуточное сообщение от JPAY при выполнении транзакции – предписание вывести определённую информацию на дисплей	+	+ сообщения display должны обрабатываться внутри ПП, однако, ПП «транслирует» эти сообщения в VEND «для информации». Ответ на сообщения display в программе ПП (в отличие от JPAY и ПП) не требуется. Если ответ всё же будет, ПП его передаст в JPAY.	+ сообщения display не обрабатываются внутри VEND, а просто «транслируются» в приложение клиента «для информации». Ответ на сообщения display в программе VEND (в отличие от JPAY) не требуется. Если ответ всё же будет – VEND его «передаст» в приложение ПП.

Команда / сообщение	Описание	JPAY	ПП	VEND
dex (вспомогательное сообщение при выполнении транзакции)	промежуточное сообщение от JPAY при выполнении транзакции – запрос значений тегов EMV	+	- необходимо запрещать в настройках вывод сообщений dex. Соответственно, сообщение dex будет вызывать ошибку, которая потребует дополнения конфигурации EMV	- необходимо запрещать в настройках вывод сообщений dex. Соответственно, сообщение dex будет вызывать ошибку, которая потребует дополнения конфигурации EMV
report	отчёт	+	+	+
settlement	сверка итогов	+	+	+
clear	очистка журнала	+	+	+
directpayment	команда для прямой работы с L2	+	+	- режим прямой работы с L2 нужен только если партнёр будет писать свой аналог JPAY и «выносить» эту команду в VEND нецелесообразно.
onlinerequest	команда для прямой работы с L2	+	+	- режим прямой работы с L2 нужен только если партнёр будет писать свой аналог JPAY и «выносить» эту команду в VEND нецелесообразно.
runreset	команда переинициализации JPAY	+	+	+

Команда / сообщение	**Описание**	**JPAY**	**ПП**	**VEND**
keep-alive (вспомогательное сообщение при выполнении инициализации)	промежуточное сообщение от JPAY при выполнении инициализации (команда runreset) – предписание ожидать результата обработки от сервера	+	+ сообщения keep-alive должны обрабатываться внутри ПП, но и сам ПП должен проинформировать вышестоящее приложение об увеличении времени ожидания. При этом программа ПП (в отличие от JPAY) не требует обязательного ответа на сообщение keep-alive. Ответить можно, если необходимо принудительно прервать текущий процесс инициализации.	+ сообщения keep-alive должны обрабатываться внутри ПП. VEND просто прозрачно «транслирует» полученные сообщения keep-alive VEND не требует обязательного ответа на сообщение keep-alive. Ответить можно, если необходимо принудительно прервать текущий процесс инициализации.
fiscal-receipt	- сформировать чек в формате JSON без фискальных признаков - передать чек в ЛК - получить ответ от ЛК с фискальными признаками - отобразить QR-код на дисплее Если возникли ошибки обработки – вернуть информацию об ошибках в вызвавшее команду приложение.	-	-	+

Команда / сообщение	**Описание**	**JPAY**	**ПП**	**VEND**
show-qr-code	- показать на экране терминала текст закодированный в виде QR-кода - запросить статус обработки предыдущей команды отображения QR-кода	-	-	+
get-work-state	- запросить текущее состояние работы терминала	-	-	+

3. Взаимодействие с программой JPAY

Соединение осуществляется в ОДНОПОЛЬЗОВАТЕЛЬСКОМ режиме через сокет. Для коммуникации используется сетевой порт 4433. Программа-клиент должна отправлять команды в формате XML и реагировать на ответы JPAY. В процессе оплаты JPAY может отправить 5-6 технических сообщений (указание вставить карту, указание забрать карту, указание ожидать ответа). Программа-клиент должна ответить на эти сообщения в обязательном порядке (отсутствие ответа или обрыв соединения сокета приводят к отмене платежа в программе JPAY).

В ответе имя корневого тега ответа JPAY должно совпадать с именем корневого тега команды. Т.е. если в JPAY передаётся команда в виде XML-объекта с корневым тегом <changepassword>, то на выходе должен быть ответ в виде XML объекта с точно таким же корневым тегом <changepassword> (при этом структура полей внутри запроса и ответа отличается – структура полей по каждой команде JPAY и ответам на команды JPAY описаны в документации JPAY). Для некоторых «сложных» команд (например, для команды <transaction>) программа JPAY может отправлять промежуточные сообщения в формате XML: «вставьте карту», «уберите карту», на которые клиентское приложение должно отреагировать. Но в конечном итоге (после выполнения всех промежуточных шагов) в ответ на команду <transaction> программа JPAY должна отправить ответ <transaction> с финальными итогами транзакции (успех или неуспех).

Следует учитывать следующие особенности JPAY

- При попытке подключиться к JPAY «параллельно» двумя приложениями на один и тот же порт 4433, во втором приложении возникнет ошибка. В архитектуре AQSI CUBE запланировано, что клиентом JPAY будет Платёжное Приложение (ПП). Соответственно, при попытке подключения к JPAY «в обход» ПП будут неизбежно возникать коллизии и ошибки. Исключение составляет только процесс конфигурирования JPAY - здесь параллельная работа через два порта 4433 т 4434 предусмотрена по документации.
- При обрыве соединения с сокетом, JPAY прекращает выполнение текущей исполняемой команды (при этом команда разработки JPAY декларирует, что финансовая транзакция не будет выполнена, однако, обрыв вполне может случиться через милисекунды после того как транзакция будет проведена на сервере и все подтверждения транзакции между JPAY и сервером будут отправлены и в этом случае прервать выполнение транзакции вряд ли получится). В связи с этим «грубый обрыв соединения с JPAY» не желателен.

3.1. Активация JPAY

Активация JPAY – это процесс «загрузки» в JPAY сертификатов, необходимых для работы JPAY:

- сертификат клиента конфигурации
- сертификат сервера конфигурации
- сертификат для получения мастер-ключей

- сертификат для взаимодействия с сервером процессинга

Все операции по «загрузки» JPAY сертификатов должны выполняться после выполнения команды

```
<runreset>
<token>DE9773A8CB888560AB0F89C07623FE03</token>
</runreset>
```

После выполнения данной команды приложение должно сохранить подключение к порту 4433, «поддерживать диалог», отвечая на промежуточные сообщения сервера <keep-alive> и ожидать завершения процесса в виде сообщения от сервера вида

```
<runreset>
<status>ok</status>
</runreset>
```

При этом параллельно к порту 4434 должна подключится программа Активатор (либо в виде Windows-приложения либо в виде серверного приложения) и передать все необходимые для работы JPAY сертификаты.

4. Платёжное Приложение

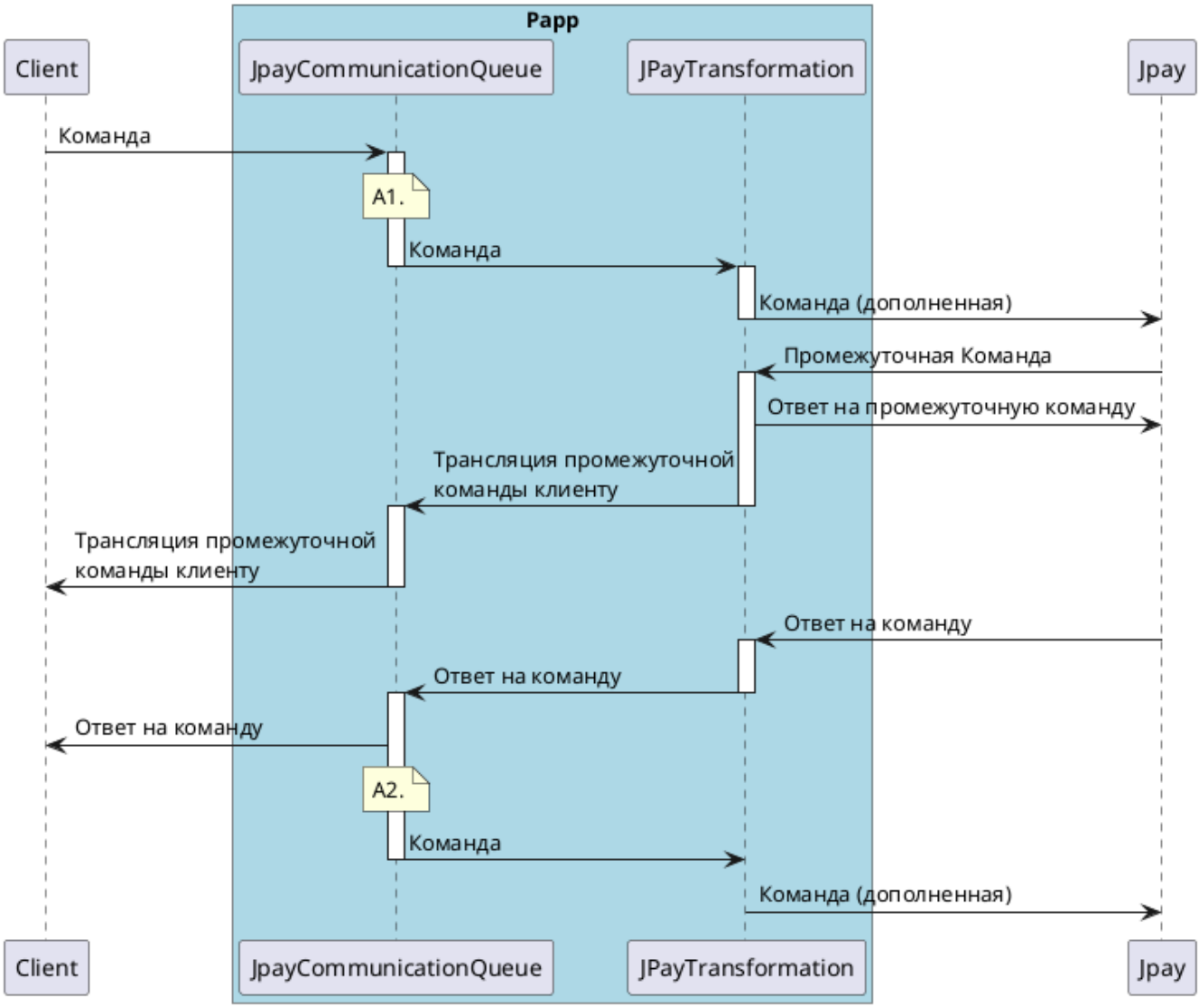
Платёжное Приложение (papp) принимает «поручения» на проведение эквайринговых операций в формате XML по аналогии с программой JPAY. Приём команд осуществляется в МНОГОПОЛЬЗОВАТЕЛЬСКОМ режиме через сокет.
Для коммуникации с приложением Vend используется сетевой порт 4444.
Для коммуникации с приложением Dm используется сетевой порт 4445.

4.1. Обработка команд входящих от клиентов.

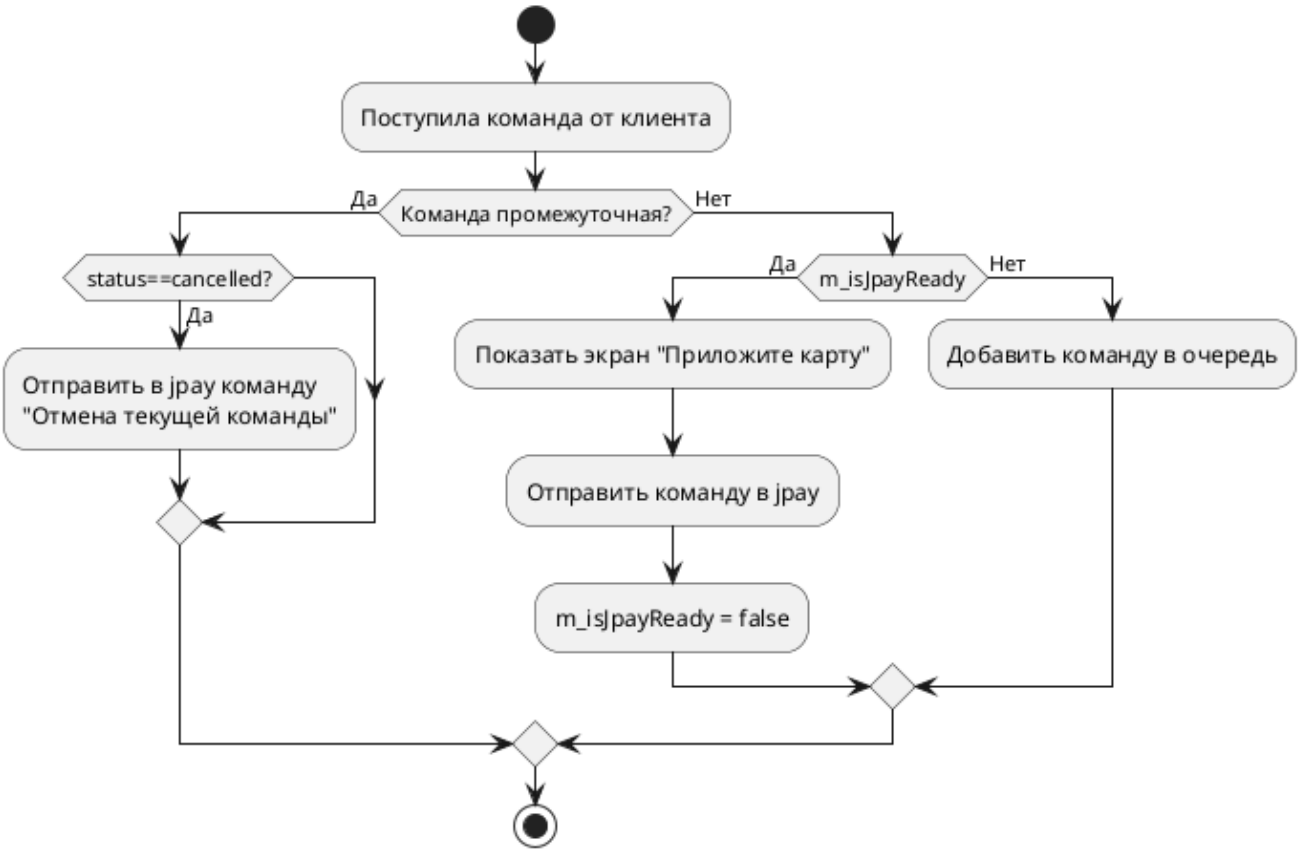
JPAY одновременно может обрабатывать лишь один запрос, а со стороны приложений vend и dm может приходить множество сообщений, до того момента как jpay закончит выполнение предыдущего.

Для решения этой задачи, алгоритм отправки команд papp→jpay реализован в виде очереди. Логика работы очереди изображена на диаграмме S1.

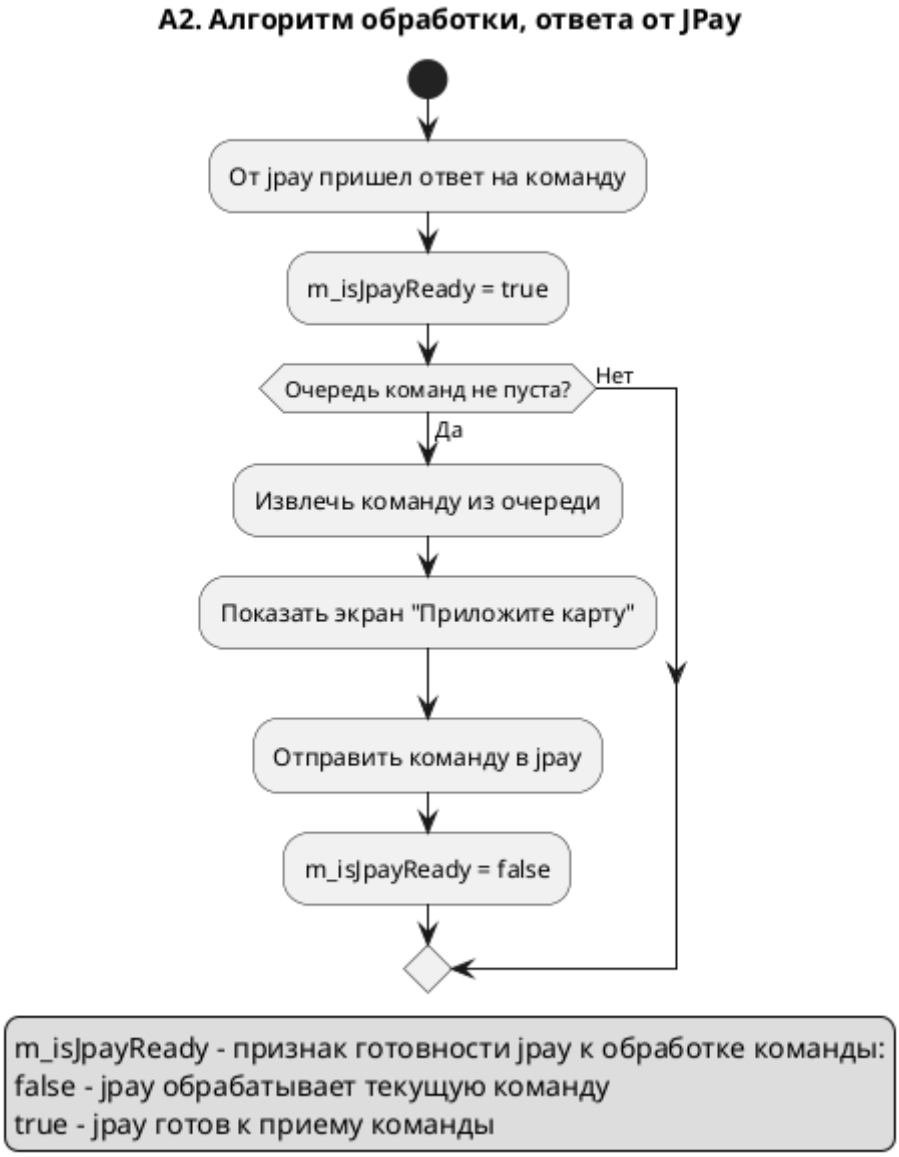
S1. Цикл обмена командами клиент <-> jpay



А1. Алгоритм обработки, входящей команды от клиента



m_isJpayReady - признак готовности jpay к обработке команды:
false - jpay обрабатывает текущую команду
true - jpay готов к приему команды



4.2. Особенности обработки команд, требующих авторизации

Платёжное Приложение должно «облегчить» работу с JPAY. В частности, во все команды, требующие авторизации, papp должен автоматически «подставлять» дополнительные теги для обеспечения авторизации. А перед началом выполнения таких команд – papp должен авторизоваться в программе JPAY и получить токен.

В частности, авторизации требуют следующие команды:

Команда	**Тип авторизации**
updateconfiguration (проверка и загрузка обновлений конфигурации с сервера конфигураций	токен
loadworkkeys (загрузка рабочих ключей)	токен
loadmasterkeys (загрузка мастер ключей)	токен

Команда	Тип авторизации
testconnection (проверка связи с сервером авторизации)	токен
getparameters (сведения о приложении и терминале)	токен
report (отчёт)	пароль
settlement (сверка итогов)	пароль
clear (очистка журнала)	пароль
runreset (команда переинициализации JPAY)	токен

В случае, если в таблице выше указан тип авторизации «пароль», программа papp должна в XML-команду добавить дополнительный тег <password>.

Пример пакета из программы VEND в программу papp:

```
<report>
  <report-type>brief</report-type>
</report>
```

Пример «дополненной» команды из программы papp в JPAY:

```
<report>
  <password>12345678</password>
  <report-type>brief</report-type>
</report>
```

В случае, если в таблице выше используется тип авторизации «токен», то программа papp должна сначала получить от программы JPAY новый токен, потом нужно передать основную команду с использованием этого токена.

Пример пакета из программы VEND в программу papp:

```
<loadmasterkeys>
</loadmasterkeys>
```

При получении такого сообщения papp должна запросить авторизационный токен:

```
<login>
<password>12345678</password>
</login>
```

Пример ответа от JPAY:

```
<login>
```

```
<status>ok</status>
<token>DE9773A8CB888560AB0F89C07623FE03</token>
</login>
```

После этого papp должна «транслировать» команду loadmasterkeys:

```
<loadmasterkeys>
<token>DE9773A8CB888560AB0F89C07623FE03</token>
</loadmasterkeys>
```

Таким образом, одна команда в программе papp превращается в три команды в программе JPAY. Можно токен не запрашивать постоянно, а хранить в какой-то переменной и использовать многократно. Если токен «устареет» с точки зрения JSON, то в какой-то момент в ответе придёт ошибка:

```
<status>notauthorized</status>
```

Тогда можно будет повторно запросить новый токен и повторить попытку обращения к JPAY.

4.3. Особенности обработки команд с промежуточными сообщениями

Протокол JPAY предполагает, что при выполнении команд transaction и runreset , могут возникать промежуточные вспомогательные сообщения <keep-alive>, <display> и <dex>.

При этом:

1. В ответ на <dex> надо возвращать сообщение <dex> со статусом “notimplemented”. Чтобы JSON даже не ждал ответа.
2. В ответ на <display> программа papp должна всегда отвечать сообщением <display> со статусом “ok” и далее – передавать полученное сообщение в программу VEND.

Логика обработки сообщения <keep-alive> должна быть следующей:

- В программе papp должна быть специальная строковая переменная keep-alive-status
- В начале выполнения любой команды <transaction> или <runreset> переменной keep-alive-status присваивается значение «ok»
- Если от программы VEND приходит сообщение <keep-alive>, то из этого сообщения считывается поле <status> и * сохраняется в переменную keep-alive-status для последующего использования
- Если от программы JPAY поступает сообщение <keep-alive> , то papp должна сразу вернуть ответ <keep-alive> с текущим статусом из переменной keep-alive-status. А сообщение от программы JPAY должно без изменений быть переслано в программу

VEND.

4.4. Специальная обработка команд JPAY для транзакций

В JPAY есть следующие типы транзакций:

- purchase (покупка)
- purchase-with-cashback (покупка с кешбеком)
- refund (возврат)
- void (отмена операции)

Платёжное приложение должно «специальным образом» обрабатывать команды **transaction**, предусмотренные протоколом JPAY: в ответ на такие команды papp должно «перехватывать» вывод на экран, отображать «на переднем плане» диалоговые окна, информирующие о статусе эквайринговой транзакции и указания для плательщика (вставить карту, вынуть карту, подождать отклика и проч.). Промежуточные ответы от JPAY типа <display>, <keepalive> и <dex> должны обрабатываться внутри papp следующим образом:

- сообщения **keep-alive** должны увеличивать тайм-аут ожидания в papp и далее передаваться в приложение VEND
- сообщения **display** должны обрабатываться внутри papp
- сообщения **dex** не желательны в приложении papp и их нужно «отключать» на уровне конфигурации EMV. Если в papp поступило сообщение dex – нужно прервать транзакцию и сохранить информацию о причине прерывания транзакции (от приложения JPAY поступило сообщение dex, которое программа papp не должна обрабатывать, необходимо изменить настройки EMV) На выходе из Платёжного Приложения клиент должен получить XML-сообщение <transaction> с итогами выполнения транзакции. В случае успешного выполнения транзакции ответ должен содержать блок <receipt>. При неуспешном выполнении транзакции в ответном сообщении <transaction> должно содержаться сообщение об ошибке.

4.4.1. Silent режим обработки транзакций

Если в команду `<transaction>` передать тег `<mode>silent</mode>`, то будет активирован Silent режим:

```
<transaction>
    ....
    <mode>silent</mode>
    ...
</transaction>
```

В данном режиме платёжное приложение не «перехватывает» вывод на экран, не

отображает «на переднем плане» диалоговые окна, информирующие о статусе эквайринговой транзакции и указания для плательщика (вставить карту, вынуть карту, подождать отклика и проч.).

4.4.2. Reader режим обработки транзакций

Если в команду `<transaction>` передать тег `<mode>reader</mode>`, то будет активирован Reader режим:

```
<transaction>
    ....
    <mode>reader</mode>
    ...
</transaction>
```

В данном режиме платёжное приложение ведёт себя аналогично тому, как вело бы себя в режиме Silent, однако выполнить транзакцию не получится. В этом режиме платёжное приложение лишь информирует о поднесённых картах, передавая сообщения, содержащие идентификатор карт.

4.5. Платёжное приложение и активации JPAY

Процесс инициализации JPAY осуществляется не через стандартный порт 4433, а через специальный порт активации 4434. Для этого порта предусмотрен специальный набор команд, которые можно передавать только на порт 4434. При активации JPAY пользователь должен «напрямую» взаимодействовать с JPAY, не используя Платёжное Приложение. Платёжное приложение не должно никак передавать команды активации JPAY и не должно осуществлять взаимодействие с портом 4434 программы JPAY. Такое решение связано с тем, что при активации более целесообразно именно «прямое» подключение к JPAY и никакие «посредники» не требуются.

4.6. Обработка ситуаций «обрыва» соединений

В случае, если клиент «оборвал» подключение к сокету Платёжного Приложения на порту 4444 в процессе выполнения транзакции, Платёжное приложение на очередное сообщение <keep-alive> от JPAY должно ответить сообщением

```
<keepalive>
    <status>cancelled</status>
</keepalive>
```

и удалить из очереди команд все сообщения, отправителем которых является этот клиент.

При этом papp **не должно** «симметрично обрывать соединение» с JPAY.

В случае, если клиент papp «оборвал» соединение с papp и сразу после этого в papp пришло

сообщение об успешном или не успешном завершении текущей выполняемой транзакции, у papp не будет возможности передать полученный результат в «вышестоящее» приложение VEND. В этом случае papp просто игнорирует результат и никуда его не передаёт. Приложение VEND самостоятельно должно «восстановить пробел» после любого обрыва при помощи отчетов (команда <report>).

5. Программа VEND

Программа VEND должна принимать через сетевой сокет 4455 запросы и ответы от клиентов в ОДНОПОЛЬЗОВАТЕЛЬСКОМ режиме. Перечень поддерживаемых команд приложения VEND и сравнение с командами JPAY приведён в разделе #2.

5.1. updateconfiguration – загрузка конфигурации с сервера

В программе VEND для обновления конфигурации следует использовать команду:

```
{
"command":"updateconfiguration"
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
"reply":"updateconfiguration",
"status":"ok"
}
```

5.2. loadmasterkey – загрузка мастер-ключей

В программе VEND для загрузки мастер-ключей следует использовать команду

```
{
"command":"loadmasterkey"
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
"reply":"loadmasterkey",
"status":"ok",
"receipt":
{  "pkcv":"123456",
     "mkcs":"789ABC"
   }
}
```

Где: pkcv - KCV мастер ключа PIN mkcv - KCV мастер ключа MAC

5.3. loadworkkeys – загрузка рабочих ключей

В программе VEND для загрузки рабочих-ключей следует использовать команду

```
{
"command":"loadworkkeys"
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
"reply":"loadworkkeys",
"status":"ok",
"mac-change-receipt": { "rrn":"123456789123" },
"net-change-receipt": { "rrn":"123456789123" },
}
```

В ответе передаются два чека содержащие rrn операций загрузки ключа MAC и NET (PEK), если это предусмотрено протоколом сервера авторизации.

5.4. testconnection – проверка связи с сервером

В программе VEND для проверки соединения с сервером следует использовать команду

```
{
"command":"testconnection"
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
"reply":"testconnection",
"status":"ok"
}
```

5.5. getparameters – сведения о приложении и терминале

В программе VEND для получения сведений о приложении и терминале следует использовать команду:

```
{
"command":"getparameters"
```

```
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
"reply":"getparameters",
"status":"ok",
"parameters": {
    "sn": "000000000009",
    "app": "1.0.67.6",
    "firmware-mcu": "1.5.3",
    "firmware-boot": "2.0.1",
    "os": "CS10_V1.07_181127PK",
    "sdk": "1.0.4",
    "tid": "1000000001",
    "mid": "243423434122313",
    "tconf": "19-04-01.01",
    "ntconf": "cname",
    "cconf": "18-10-25.06",
    "ncconf": "cname_test",
    "econf": "19-04-13.02",
    "neconf": "2can_jibe_emv",
    "kconf": "18-08-30.04",
    "nkconf": "combined_ca_database",
    "acqid": "twocan",
    "cccert": "24.03.2027",
    "csca": "20.11.2037",
    "cshost": "192.168.0.185",
    "accert": "24.03.2027",
    "asca": "12.10.2020",
    "ashost": "192.168.0.2",
    "kccert": "24.03.2027",
    "ksca": "none",
    "devid": "M2100-0000005164"
  }
}
```

Где:

- **sn** - серийный номер терминала
- **app** - версия приложения
- **firmware-mcu** - версия защищенного кернела терминала
- **firmware-boot** - версия загрузчика
- **os** - версия операционной системы
- **sdk** - версия SDK (aar)
- **tid** - идентификатор (номер) терминала

- **mid** - идентификатор мерчанта
- **tconf** - версия конфигурации терминала
- **ntconf** - имя конфигурации терминала
- **cconf** - версия общей конфигурации
- **ncconf** - имя общей конфигурации
- **econf** - версия EMV конфигурации
- **neconf** - имя EMV конфигурации
- **kconf** - версия списка ключей платежных систем конфигурации
- **nkconf** - имя списка ключей платежных систем конфигурации
- **acqid** - идентификатор эквайера
- **mccert** - дата окончания действия клиентского сертификата для подключения к серверу онфигурации
- **csca** - дата окончания действия CA сервера конфигурации
- **cshost** - имя хоста или IP сервера конфигурации
- **accert** - дата окончания действия клиентского сертификата для подключения к серверу эквайера
- **asca** - дата окончания действия CA сервера эквайера
- **ashost** - имя хоста или IP сервера эвайера
- **kccert** - дата окончания действия клиентского сертификата для подключения к серверу загрузки ключей
- **ksca** - дата окончания действия CA сервера загрузки ключей
- **devid** - Идентификатор устройства для эквайинга

5.6. transaction – транзакции

Предусмотрены 4 типа транзакций: purchase (покупка) purchase-with-cashback (покупка с кешбеком) refund (возврат) void (отмена) В зависимости от типа транзакции передаются разные параметры в запросе:

```
{
    "command":"transaction",
    "type": "purchase",
    "currency": 643,
    "amount": 123.00
}

{
   "command":"transaction",
    "type": "refund",
    "amount": 123.00,
    "rrn": "120157645346"
```

```
}

{
    "command":"transaction",
    "type": "purchase-with-cashback",
    "amount": 123.00,
    "amount-other": "000000000100"
}

{
      "command":"transaction",
      "type": "void",
      "number": "000008",
      "amount": 1.00
}

{
    "command":"transaction",
    "type": "purchase",
    "amount": 123.00,
    "pan": "4000000010000001",
    "expired": "1805",
    "cardholder": "JOHN SMITH"
}
```

В поле fiscalParams транзакции типа purchase можно поместить все параметры необходимые для фискализации после оплаты. Массив payments при этом не нужно заполнять. Поле автоматически будет заполнено после успешного приёма платежа. Поле payments может отсутствовать в запросе (см. Параметры, необходимые для фискализации).

```
{
    "command": "transaction",
    "type": "purchase",
    "currency": 643,
    "amount": 123.00,
    "fiscalParams": {
        "amount": 123,
        "content": {
            "automatNumber": "777",
            "checkClose": {
                "payments": [
                ],
                "taxationSystem": 4
            },
            "positions": [
                {
                    "paymentMethodType": 4,
                    "paymentSubjectType": 10,
                    "price": 123,
                    "quantity": 1,
```

```
                    "slotInfo": {
                        "slotId": -1
                    },
                    "tax": 1,
                    "text": "Тестовый товар"
                }
            ],
            "settlementAddress": "Московская обл, г Одинцово, поселок Барвиха",
            "settlementPlace": "За углом",
            "type": 1
        }
    }
}
```

Параметр	Тип	Значение
type	string	Тип операции
amount	float	Сумма число float. Копейки отделяются десятичной точкой. Всего – не более 12 знаков, включая копейки. Для транзакции void сумма может отсутствовать. В этом случае отмена производится на всю сумму отменяемой транзакции.
amount-other	float	Сумма кэшбэка для операции **purchasewithcashback**. Требования аналогичны к полю amount.
currency	integer	Код валюты, если он отличается от кода валюты по умолчанию, указанного в конфигурации.
rrn	string	retrieval refernce number (12 символов; необязательный параметр операции refund)
number	string	Номер чека (6 цифр)
pan	string	Номер карты. Указывается при вводе данных карты вручную.
expired	string	Окончание срока действия карты (4 цифры, ггмм). Указывается при вводе данных карты вручную.
password	string	Пароль администратора (8 цифр). Указывается при вводе данных карты вручную.
cardholder	string	Имя владельца карты. Указывается при вводе данных карты вручную

```
{
   "reply":"transaction",
   "status": "ok",
   "acquiringData":
    {
```

```
      "id":"49678657-3d7a-4f2c-1ea8-508d8112345a",
      "type":"purchase",
      "date":1580985696,
      "automatonId":"19178557-6d7a-4f2c-8ea5-308d8166061a",
      "amount":123.00,
      "acquirerBankName":"TINKOFF",
      "transactionId":"1000000001",
      "slipNumber":"000009",
      "approvalCode":"AFK045",
      "terminalId":"1000000001",
      "maskPan":"************1234",
      "AID":"A00000000031010",
      "TVR":"2040300000",
      "TSI":"EF00",
      "APN":"VISA Classic",
      "IIN":"VISA Classic",
      "RRN":" 120157645346",
      "signNeeded":false,
      "expirationDate":"2112",
      "companyName":null,
      "customerContact": null,
      "calculationAddress": null,
      "errorCode":"000",
      "errorMessage":"",
      "text": null,
      "cardholder":"IVAN IVANOV",
      "acquirerAgentName":"tinkoff",
      "cashlessType":2
    }
   "error-stack": ["message1", "message2"]
  }

}
```


Для всех 4 видов транзакций набор полей первого уровня – идентичен. Объект acquiringData формируется в соответствии с форматом эквайрингового слипа в транспортном формате.

Программа VEND должна выполнять преобразование эквайринговых слипов из формата JPAY к формату транспортной модели. Преобразование осуществляется в соответствии со следующей таблицей:

Эквайринговый слип

Поле транспортной модели	**Тип**	**Поле модели JPAY**	**Пояснение**
id	UUID(v4)	-	Уникальный идентификатор. При конвертации необходимо сгенерировать уникальное значение UUID.

Поле транспортной модели	Тип	Поле модели JPAY	Пояснение
type	enum	type	Тип слипа. Возможные значения: purchase, void, refund, purchasewithcashback.
date	timestamp with time zone	datetime	Дата операции. В JPAY дата передаётся в формате yymmddHHMMSS. При конвертации в транспортную модель необходимо конвертировать это значение в datetime
cashierId	UUID(v4)	из настроек VEND	Идентификатор кассира. Поле заполняется на основнаии настроек программы VEND. В модели JSON никаких данных о кассире нет.
automatonId	UUID(v4)	из настроек VEND	Идентификатор автомата. Поле заполняется на основнаии настроек программы VEND. В модели JSON никаких данных об автомате нет.
amount	float8	amount-authorized	Сумма. Следует учитывать, что в JPAY фигурируют суммы в копейках, а в транспортной модели используются рубли с дробной частью.
acquirerBankName	varchar(64)	bank-name	Наименование банка. Это значение JPAY возвращает на основании данных конфигурации common_config. Т.е. значение задаёт отдел эквайринга при формировании типового файла конфигурации.
transactionId	varchar(64)	tid	ID операции в банке. В настоящий момент данное поле заполняется из параметра tid. Однако tid также используется и для terminalId. Необходимо уточнить.
slipNumber	varchar(64)	seq	Номер слипа
approvalCode	varchar(64)	auth-number	Код подтверждения
terminalId	varchar(20)	tid	Id терминала В настоящий момент данное поле заполняется из параметра tid. Однако tid также используется и для transacrionId. Необходимо уточнить.
maskPan	varchar(19)	pan	Маскированный номер карты
AID	varchar(18)	aid	AID - номер авторизации в сети
TVR	varchar(16)	tvr	TVR - аттрибут слипа
TSI	varchar(16)	tsi	TSI - аттрибут слипа

Поле транспортной модели	Тип	Поле модели JPAY	Пояснение
APN	varchar(64)	appname	APN - аттрибут слипа. Тип сети авторизации(Тип карты)
IIN	varchar(64)	либо appname либо alabel	IIN - аттрибут слипа. Тип карты MasterCard/Visa В кассах это поле заполняется из поля appname. Если поле пустое – используется alabel
RRN	varchar(64)	rrn	RRN - аттрибут слипа. Номер операции для возвратов
signNeeded	boolean	signature	Требуется ли подпись клиента
expirationDate	varchar(4)	aed	Дата окончания срока действия карты
companyName	varchar(255)	из настроек VEND	Название организации. Поле заполняется на основнаии настроек программы VEND.
customerContact	varchar(64)	из настроек VEND	Email или телефон клиента. Поле заполняется на основнаии настроек программы VEND.
calculationAddress	varchar(128)	из настроек VEND	Адрес установки ККТ для проведения расчетов. Поле заполняется на основнаии настроек программы VEND.
errorCode	text	error-code	Этот тег содержит мнемонику кода ошибки команды. Там может быть, например, notauthorized или ok. Это служебная информация.
transactionResult	text	status	Код ошибки при оплате по которой мы можем понять успешна она или нет (approved или declined соответственно).
errorMessage	varchar(1024)	decline-reason	Описание ошибки при оплате
respCode	varchar(64)	resp-code	Код ошибки при оплате.
text	text	-	Текст слипа
cardholder	varchar(255)	cardholder	Имя держателя карты
acquirerAgentName	varchar(255)	acquirer-tag	Посредник до банка эквайера

Поле транспортной модели	Тип	Поле модели JPAY	Пояснение
cashlessType	enum	При оплате картой указывается значение “2”	Тип безналичной оплаты (1=Безналичные, 2=картой, 3=QR код)

Следует отметить, что для корректного формирования эквайрингового слипа в соответствии с транспортной моделью необходимо сделать следующие настройки в программе VEND:

Поле транспортной модели	Тип	Пояснение
automatonId	UUID(v4)	Идентификатор автомата. Поле заполняется на основании настроек программы VEND, которые необходимо передать из Личного кабинета.
automatNumber	varchar(20)	Номер автомата (поле 1036 ФФД). В качестве номера автомата следует передавать инвентарный номер автомата.
companyName	varchar(255)	Название организации. Поле заполняется на основании настроек в личном кабинете. Нужно эти данные передавать вместе с настройками из ЛК.
customerContact	varchar(64)	Email или телефон клиента. Поле заполняется в том случае, если во время оплаты клиент каким-либо образом в интерфейсе ПО AQSI CUBE указал свой телефон или адрес электронной почты. По умолчанию данное поле пустое.
calculationAddress	varchar(128)	Адрес установки ККТ для проведения расчетов. Поле заполняется на основнаии настроек программы VEND, которые должны передаваться из Личного Кабинета

5.7. report – отчет

В программе VEND для получения отчета следует использовать команду

```
{
    "command":"report",
    "report-type":"brief"
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:


```
{
    "reply":"report",
    "status":"ok",
    "xreport": {
    "header": {
        "mid": "M123456789012345",
        "tid": "1000000001",
        "title": { "line": "Header line 1" },
        "cur": "643",
        "time": "189329120002"
        },
    "transactions": {
          {
        "status": "approved",
        "state": "active",
        "type": "refund",
        "seq": "000009",
        "aa": "000000001000",
        "ao": "000000000100",
        "rrn": "647394847384",
        "time": "180322121408",
        "apn": "AFJ879",
        "arc": "000",
        "cur": "643",
        "pan": "************1234",
        "aex": "2112",
        "aid": "A0000000031010",
        "tvr": "0000000000",
        "tsi": "000000"
           },
           {
        "status": "approved",
        "state": "active",
        "type": "refund",
        "seq": "000010",
        "aa": "000000001000",
        "ao": "000000000100",
        "rrn": "647394847373",
        "time": "180322128383",
        "apn": "AFJ879",
        "arc": "000",
        "cur": "643",
        "pan": "************1234",
        "aex": "2112",
        "aid": "A0000000031010",
        "tvr": "0000000000",
        "tsi": "000000"
          },
```

```
        ...
    },
    "totals": {
          {
        "rid": "A000000003",
        "name": "VISA",
        "purchase-count": 100,
        "purchase-sum": 100.00,
        "refund-count": 92,
        "refund-sum": 82832.88,
        "purchase-with-cashback-count": 92,
        "purchase-with-cashback-sum": 922.45,
        "purchase-reversal-count": 0,
        "purchase-reversal-sum": 0.00,
        "refund-reversal-count": 0,
        "refund-reversal-sum": 0.00,
        "total-card-count": 0,
        "total-card-sum": 0.00
          },
          {
        "rid": "A0000000032010",
        "name": "Visa Electron",
        "purchase-count": 100,
        "purchase-sum": 100.00,
        "refund-count": 92,
        "refund-sum": 82832.88,
        "purchase-with-cashback-count": 92,
        "purchase-with-cashback-sum": 922.45,
        "purchase-reversal-count": 0,
        "purchase-reversal-sum": 0.00,
        "refund-reversal-count": 0,
        "refund-reversal-sum": 0.00,
        "total-card-count": 0,
        "total-card-sum": 0.00
          }
          ...

    },
    "grand-totals": {
        "purchase-total-count": 100,
        "purchase-total-sum":   100.00,
        "refund-total-count": 92,
        "refund-total-sum": 82832.88,
        "purchase-with-cashback-total-count": 92,
        "purchase-with-cashback-total-sum": 922.45,
        "purchase-reversal-total-count": 0,
        "purchase-reversal-total-sum": 0.00,
        "refund-reversal-total-count": 0,
        "refund-reversal-total-sum": 0.00,
        "total-card-total-count": 0,
        "total-card-total-sum": 0.00
```

```
            }
        }

    }
```

- status - код ошибки
- tid - идентификатор (номер) терминала
- mid - идентификатор владельца терминала (Merchant ID)
- title - строки заголовка чека
- cur - код валюты
- time - дата и время составления отчета (yymmddHHMMSS)
- transactions - список транзакций в полном (full) отчете. Для каждой транзакции указывается:
- status - код ошибки
- state - значение этого поля в чеке всегда active
- type - тип транзакции
- seq - номер чека
- aa - сумма транзакции в копейках
- ao - сумма кэшбека (для транзакции purchase-with-cashback)
- rrn - retrieval reference number
- time - дата и время проведения транзакции (yymmddHHMMSS)
- apn - код авторизации.
- arc - код ответа сервера авторизации
- cur - код валюты
- pan - последние 4 цифры номера карты
- aex - дата окончания срока действия карты (ггмм)
- aid - идентификатор приложение EMV карты (AID)
- tvr - значение тэга terminal verification result
- tsi - значение тэга transaction status information
- totals - итоговые данные отчета сгруппированные по типу карты. Для каждого типа карты указывается:
    - rid - registered application provider identifier (RID) карты
    - name - название карты
    - purchase-count - количество операций оплата
    - purchase-sum - общая сумма всех операций оплата
    - refund-count - количество операций возврат

    - refund-sum - общая сумма всех операций возврат
    - purchase-with-cashback-count - количество операций оплата с кэшбэком
    - purchase-with-cashback-sum - общая сумма всех операций оплата с кэшбэком
    - purchase-reversal-count - количество операций отмены оплаты
    - purchase-reversal-sum - общая сумма всех операций отмены оплаты
    - refund-reversal-count - количество операций отмены возврата
    - refund-reversal-sum - общая сумма всех операций отмены возврата
    - total-card-count - общее количество операций по карте
    - total-card-sum - баланс всех операций по карте. Баланс это 12 цифр со знаком.
- grand-totals - итоговые данные очета:
    - purchase-total-count - количество операций оплата
    - purchase-total-sum - общая сумма всех операций оплата
    - refund-total-count - количество операций возврат
    - refund-total-sum - общая сумма всех операций возврат
    - purchase-with-cashback-total-count - количество операций оплата с кэшбэком
    - purchase-with-cashback-total-sum - общая сумма всех операций оплата с кэшбэком
    - purchase-reversal-total-count - количество операций отмены оплаты
    - purchase-reversal-total-sum - общая сумма всех операций отмены оплаты
    - refund-reversal-total-count - количество операций отмены возврата
    - refund-reversal-total-sum - общая сумма всех операций отмены возврата
    - total-count - общее количество операций
    - total-sum - баланс всех операций.

5.8. settlement– сверка итогов

В программе VEND для получения сверки итогов следует использовать команду

```
{
    "command":"settlement",
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
    "reply":"settlement",
    "status":"ok",
    "sreport": {
        "resp-code": "000",
        "approval-number": "ASD002",
```

```
        "rrn": "837495759322",
        "orig-amount": "D003030001000",
        "amount": "D003030001000",
        "datetime": "180322102354",
        "tid": "1000000001",
        "mid": "123456789012345",
        "cur": "643",
        "from": "190101100000",
        "to": "190102110000",
        "tnum": "100",
        "batch-upload": "passed",
        "art-resp-code": "00",
        "cov-resp-code": "00",
        "cov-rrn": "123456789012",
        "settlement-result": "passed"
    }
}
```

- **status** - код ошибки
- **resp-code** - код ответа сервера авторизации
- **approval-number** - код авторизации.
- **rrn** - retrieval reference number
- **orig-amount** - итоговая сумма подсчитанная на терминале
- **amount** - итоговая сумма переданная сервером авторизации
- **datetime** - дата и время проведения транзакции (yymmddHHMMSS)
- **tid** - идентификатор (номер) терминала
- **mid** - идентификатор владельца терминала (Merchant ID)
- **cur** - код валюты
- **from** - дата и время первой транзакции в сверке
- **to** - дата и время последней транзакции в сверке
- **tnum** - количество транзакций в сверке
- **batch-upload** - в случае если суммы при сверке не совпали требуется загрузка транзакций. В этом теге указывается результат такой загрузки.passed - загрузка выполнена успешно, failed - загрузка транзакций не удалась. Этот тег может отсутствовать.
- **art-resp-code** - Код ответа на запрос завершающий загрузку транзакций. Этот тег может отсутствовать.
- **cov-resp-code** - код возврата операции очистки журнала. Тег включается в ответ, если после сверки итогов выполняется операция очистки журнала (cutover). Для этого в конфигурации надо указать настройку settlement cutover=“true” [4].
- **cov-rrn** - RRN операции очистки журнала. Присутствует в случае если операция очистки журнала успешно выполнена

- **settlement-result** - результат операции сверки итогов. passed - сверка завершена успешно. failed - операция не выполнена.

5.8. settlement– сверка итогов

В программе VEND для получения сверки итогов следует использовать команду

```
{
    "command":"settlement",
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
    "reply":"settlement",
    "status":"ok",
    "sreport": {
        "resp-code": "000",
        "approval-number": "ASD002",
        "rrn": "837495759322",
        "orig-amount": "D003030001000",
        "amount": "D003030001000",
        "datetime": "180322102354",
        "tid": "1000000001",
        "mid": "123456789012345",
        "cur": "643",
        "from": "190101100000",
        "to": "190102110000",
        "tnum": "100",
        "batch-upload": "passed",
        "art-resp-code": "00",
        "cov-resp-code": "00",
        "cov-rrn": "123456789012",
        "settlement-result": "passed"
    }
}
```

- **status** - код ошибки
- **resp-code** - код ответа сервера авторизации
- **approval-number** - код авторизации.
- **rrn** - retrieval reference number
- **orig-amount** - итоговая сумма подсчитанная на терминале
- **amount** - итоговая сумма переданная сервером авторизации
- **datetime** - дата и время проведения транзакции (yymmddHHMMSS)

- **tid** - идентификатор (номер) терминала
- **mid** - идентификатор владельца терминала (Merchant ID)
- **cur** - код валюты
- **from** - дата и время первой транзакции в сверке
- **to** - дата и время последней транзакции в сверке
- **tnum** - количество транзакций в сверке
- **batch-upload** - в случае если суммы при сверке не совпали требуется загрузка транзакций. В этом теге указывается результат такой загрузки.passed - загрузка выполнена успешно, failed - загрузка транзакций не удалась. Этот тег может отсутствовать.
- **art-resp-code** - Код ответа на запрос завершающий загрузку транзакций. Этот тег может отсутствовать.
- **cov-resp-code** - код возврата операции очистки журнала. Тег включается в ответ, если после сверки итогов выполняется операция очистки журнала (cutover). Для этого в конфигурации надо указать настройку **settlement cutover**=“true” [4].
- **cov-rrn** - RRN операции очистки журнала. Присутствует в случае если операция очистки журнала успешно выполнена
- **settlement-result** - результат операции сверки итогов. passed - сверка завершена успешно. failed - операция не выполнена.

5.9. clear – очистка журнала

В программе VEND для очистки журнала следует использовать команду

```
{
    "command":"clear",
}
```

В ответ программа VEND должна вернуть JSON со следующей структурой:

```
{
    "reply":"clear",
    "status":"ok",
    "sreport": {
        "resp-code": "000",
        "approval-number": "ASD002",
        "rrn": "837495759322",
        "orig-amount": "D003030001000",
        "amount": "D003030001000",
        "datetime": "180322102354",
        "tid": "1000000001",
        "mid": "123456789012345"
    }
```

```
}
```

- **status** - код ошибки
- **resp-code** - код ответа сервера авторизации
- **approval-number** - код авторизации.
- **rrn** - retrieval reference number
- **orig-amount** - итоговая сумма подсчитанная на терминале
- **amount** - итоговая сумма переданная сервером авторизации
- **datetime** - дата и время проведения транзакции (yymmddHHMMSS)
- **tid** - идентификатор (номер) терминала
- **mid** - идентификатор владельца терминала (Merchant ID)

5.10. runreset – переинициализация

В программе VEND для запуска инициализации следует использовать команду

```
{
    "command":"runreset",
}
```

Сразу после запуска нужно подключаться к порту 4434 и проводить инициализацию JPAY в соответствии с руководством программы JPAY.

По окончании инициализации программа VEND должна вернуть JSON со следующей структурой:

```
{
    "reply":"runreset",
    "status":"ok",
}
```

5.11. keep-alive – сообщение об увеличении времени ожидания

В отличие от команд, которые отправляет клиент в программу VEND, источником сообщений **keep-alive** является эквайринговое ядро. В соответствии с разделом [4.2. Особенности обработки команд с промежуточными сообщениями] программа ПП должна сразу же отвечать на это сообщение, передавая статус, который хранится в переменной keep-alive-status.

Для того чтобы изменить статус переменной keep-alive-status необходимо в программу VEND передать сообщение

```
{
    "keep-alive-status":"cancelled"
}
```

Программа VEND должна преобразовать это сообщение в формат протокола JPAY следующего вида:

```
<keep-alive>
    <status>cancelled</status>
</keep-alive>
```

В свою очередь, ПП при получении такого сообщения должен изменить статус переменной **keep-alive-status** и при получении очередного сообщения <keep-alive> от эквайрингового ядра даст команду на отмену текущей выполняемой транзакции.

Программа VEND при получении сообщения keep-alive-status от клиента не направляет никакого ответа и отмена транзакции не гарантируется - если транзакция уже прошла и отменять поздно – отмена не возможна. Если отмена возможна – транзакция будет отменена и эквайринговое ядро вернёт это в ответном сообщении на команду транзакции. Реальный результат транзакции клиент должен определять по ответу эквайрингового ядра в сообщениях вида:

```
{ "reply": "transaction", ··· }
```

5.12. fiscal-receipt – фискализация чека

Для фискализации чека нужно отправить в терминал команду **fiscal-receipt** с типом **create**. В поле fiscalParams необходимо поместить все параметры необходимые для фискализации (см. Параметры, необходимые для фискализации).

Пример команды **create**:

```
{
    "command": "fiscal-receipt",
    "type": "create",
    "fiscalParams": {
        "amount": 15,
        "content": {
            "automatNumber": "777",
            "checkClose": {
                "payments": [
                    {
                        "acquiringData": {
                            "AID": "A0000000041010",
                            "APN": "MC Credit/Debit",
```

```
                            "IIN": "MC Credit/Debit",
                            "RRN": "111554544569",
                            "TSI": "",
                            "TVR": "0000008001",
                            "acquirerAgentName": null,
                            "acquirerBankName": "ДЕМО БАНК (ПАО)\r\n",
                            "amount": 15,
                            "approvalCode": "53494D554C41",
                            "cardholder": null,
                            "cashlessType": 2,
                            "date": "2022-01-12T19:05:52.000+03:00",
                            "errorCode": "00",
                            "errorMessage": null,
                            "expirationDate": "2611",
                            "id": "df3816c4-d970-458c-b500-4ee8e31c42ec",
                            "maskPan": "************6105",
                            "signNeeded": false,
                            "slipNumber": "000009",
                            "terminalId": "04206819",
                            "transactionId": "04206819",
                            "transactionResult": "approved",
                            "type": "purchase"
                        },
                        "amount": 15,
                        "type": 1
                    }
                ],
                "taxationSystem": 4
            },
            "positions": [
                {
                    "paymentMethodType": 4,
                    "paymentSubjectType": 10,
                    "price": 15,
                    "quantity": 1,
                    "slotInfo": {
                        "slotId": -1
                    },
                    "tax": 1,
                    "text": "Вода 5 л"
                }
            ],
            "settlementAddress": "Московская обл, г Одинцово, поселок Барвиха",
            "settlementPlace": "За углом",
            "type": 1
        }
    }
}
```


Если терминал успешно принял команду формирования чека, то он присылает ответ со

статусом «ok» и возвращает **id** чека.

Пример успешного ответа на команду **create**:


```
{
    "reply": "fiscal-receipt",
    "type": "create",
    "status": "ok",
    "id": "257879c2-105e-4c6b-8462-00d8e837b9a4"
    "connection-status": "connected"
}
```


Далее происходит процесс обработки и фискализации чека, который может занять некоторое время. Если процесс фискализации прошел успешно, то чек сохраняется в памяти терминала.

Для того, чтобы получить результат фискализации необходимо запросить чек по его **id** с помощью команды **fiscal-receipt** с типом **get**.

Пример команды get:


```
{
    "command": "fiscal-receipt",
    "type": "get",
    "id": "257879c2-105e-4c6b-8462-00d8e837b9a4",
    "delete": true
}
```


В случае, когда чек успешно фискализирован и находится в памяти устройства, то терминал возвращает его в ответе. Если поле **delete** было установлено как true, то после ответа чек удаляется из памяти.

Пример успешного ответа на команду **get**:


```
{
    "reply": "fiscal-receipt",
    "type": "get",
    "status": "ok",
    "receipt": {
        "accountId": 85,
        "amount": 15,
        "automatId": 1129,
        "calculationAddress": null,
        "calculationPlace": null,
        "callbackUrl": "http://10.125.0.129:3000/api/receipts/orange-data/257879c2-
105e-4c6b-8462-00d8e837b9a4",
        "change": null,
        "companyINN": "5260380716",
```

```
        "companyName": "ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ \"СИТИКАРД\"",
        "content": {
            "automatNumber": "777",
            "checkClose": {
                "payments": [
                    {
                        "acquiringData": {
                            "AID": "A0000000041010",
                            "APN": "MC Credit/Debit",
                            "IIN": "MC Credit/Debit",
                            "RRN": "111554544569",
                            "TSI": "",
                            "TVR": "0000008001",
                            "acquirerAgentName": null,
                            "acquirerBankName": "ДЕМО БАНК (ПАО)\r\n",
                            "amount": 15,
                            "approvalCode": "53494D554C41",
                            "cardholder": null,
                            "cashlessType": 2,
                            "date": "2022-01-12T19:05:52.000+03:00",
                            "errorCode": "00",
                            "errorMessage": null,
                            "expirationDate": "2611",
                            "id": "df3816c4-d970-458c-b500-4ee8e31c42ec",
                            "maskPan": "************6105",
                            "signNeeded": false,
                            "slipNumber": "000009",
                            "terminalId": "04206819",
                            "transactionId": "04206819",
                            "transactionResult": "approved",
                            "type": "purchase"
                        },
                        "amount": 15,
                        "type": 1
                    }
                ],
                "taxationSystem": 4
            },
            "positions": [
                {
                    "paymentMethodType": 4,
                    "paymentSubjectType": 10,
                    "price": 15,
                    "quantity": 1,
                    "slotInfo": {
                        "slotId": -1
                    },
                    "tax": 1,
                    "text": "Вода 5 л"
                }
            ],
```

```
            "settlementAddress": "Московская обл, г Одинцово, поселок Барвиха",
            "settlementPlace": "За углом",
            "type": 1
        },
        "createdAt": "2022-01-12T16:06:03.399Z",
        "deletedAt": null,
        "deviceId": 115,
        "deviceRN": "0000000000038978",
        "deviceSN": "0471840028000201",
        "documentIndex": 1,
        "documentNumber": 583,
        "fiscalizationError": null,
        "fiscalizationStatus": 3,
        "fnsWebsite": "www.nalog.ru",
        "fp": "3309352446",
        "fsNumber": "9999078900006906",
        "id": "257879c2-105e-4c6b-8462-00d8e837b9a4",
        "isNonFiscal": false,
        "isOrangeDataReceipt": true,
        "ofdInn": null,
        "ofdName": "Такском ТЕСТ",
        "ofdWebsite": "www.taxcom.ru",
        "operationMode": 0,
        "operationNumber": null,
        "processedAt": "2022-01-12 19:06:03.134",
        "processedAtTz": "2022-01-12T16:06:03.134Z",
        "returnCheckId": null,
        "type": 0,
        "updatedAt": "2022-01-12T16:06:04.189Z"
    }
}
```


Пример ответа на команду **get**, если в памяти терминала отсутствует чек с запрошенным **id**:


```
{
    "reply": "fiscal-receipt",
    "type": "get",
    "status": "failed",
    "id": "257879c2-105e-4c6b-8462-00d8e837b9a4"
}
```


Для получения списка чеков, находящихся в памяти терминала, нужно отправить в терминал команду **fiscal-receipt** с типом **get-list**.

Пример команды **get-list**:


```
{
    "command": "fiscal-receipt",
    "type": "get-list"
```

```
}
```

В ответ на команду **get-list** терминал передает массив **id-list**, сохраненных в памяти чеков (может быть пустым).

Пример ответа на команду **get-list**:

```
{
    "reply": "fiscal-receipt",
    "type": "get-list",
    "status": "ok",
    "id-list": [
        {
            "createdAt": "2022-06-01T08:09:57.520Z",
            "id": "866f57f8-9c8d-4b45-834e-412417502a05"
        },
        {
            "createdAt": "2022-06-01T08:08:49.376Z",
            "id": "2358e847-a777-409e-bf4c-011577d0b105"
        },
        {
            "createdAt": "2022-06-01T08:08:38.325Z",
            "id": "a98c7192-e775-403e-b8b7-6ad5a8a10c2d"
        }
    ]
}
```

Удаление чека по его **id** осуществляется с помощью команды **fiscal-receipt** с типом **delete**.

Пример команды delete:

```
{
    "command": "fiscal-receipt",
    "type": "delete",
    "id": "257879c2-105e-4c6b-8462-00d8e837b9a4"
}
```

Если чек успешно удален или отсутствует в памяти, терминал отвечает на команду статусом «ok» и возвращает **id** удаленного/отсутствующего чека.

Пример ответа на команду **delete**:

```
{
    "reply": "fiscal-receipt",
    "type": "delete",
    "status": "ok",
    "id": "257879c2-105e-4c6b-8462-00d8e837b9a4"
```

```
}
```

5.13. show-qr-code – показать QR-код на экране терминала

Чтобы показать QR-код нужно отправить в терминал команду **show-qr-code**. В поле **text** необходимо поместить текст, который будет закодирован в виде QR-кода. Дополнительно в поле **timeout** можно передать время в секундах в течение которого будет показываться QR-код. Если поле **timeout** не передаётся, то используются настройки конфигурации устройства.

Пример команды **show-qr-code**:

```
{
    "command": "show-qr-code",
    "text": "Здесь могла быть ваша реклама!",
    "timeout": 20
}
```

Если терминал успешно принял команду и выполнил показ QR-кода, то он присылает ответ со статусом **ok**.

Пример успешного ответа на команду **show-qr-code**:

```
{
    "reply": "show-qr-code",
    "status": "ok"
}
```

Если была отправлена команда **show-qr-code**, а предыдущий QR-код ещё отображается или находится в очереди на показ, то он будет обновлён на новый. При этом будет возвращён статус ответа **updated**.

```
{
    "reply": "show-qr-code",
    "status": "updated"
}
```

Если терминал успешно принял команду и не может в данный момент показать QR-код, то он присылает ответ со статусом **queued** (в очереди на показ). Как только терминал сменит своё состояние на то, в котором может быть показан QR-код, то он будет показан.

Пример ответа на команду **show-qr-code**, если терминал не может показать его в данный момент:

```
{
    "reply": "show-qr-code",
    "status": "queued"
}
```

Если была отправлена команда **show-qr-code** с пустым полем **text**, то показ текущего или находящегося в очереди на показ QR-кода немедленно прекращается.

```
{
    "command": "show-qr-code",
    "text": ""
}
```

Пример успешного выполнения операции:

```
{
    "reply": "show-qr-code",
    "status": "ok"
}
```

Если в момент вызова команды не показывается QR-код или его нет в очереди, то операция отменяется:

```
{
    "reply": "show-qr-code",
    "status": "cancelled"
}
```

Для запроса статуса обработки показа предыдущего QR-кода можно воспользоваться командой **show-qr-code** с полем **type** со значением **get-status**. Возможные варианты ответа: **ok** (показан), **queued** (в очереди на показ), **cancelled** (команды на показ QR-кода ещё не поступало).

```
{
    "command": "show-qr-code",
    "type": "get-status"
}
```

Пример ответа, если команды на показ ещё не поступало:

```
{
    "reply": "show-qr-code",
    "type": "get-status",
    "status": "cancelled"
```

```
}
```

5.14. show-customer-screen – показать сообщение отправленное пользовательским приложением на экране терминала

Чтобы показать сообщение отправленное пользовательским приложением нужно отправить в терминал команду **show-customer-screen** (реализовано в 1.0.6-rc43). В поле **text** необходимо поместить текст, который будет отобаражаться на экране. Дополнительно в поле **timeout** можно передать время в секундах в течение которого будет отображаться сообщение. Если поле **timeout** не передаётся, то используются настройки конфигурации устройства.

Пример команды **show-customer-screen**:

```
{
    "command": "show-customer-screen",
    "text": "Ваш запрос находиться в обработке",
    "timeout": 20
}
```

Если терминал успешно принял команду и отобразил сообщение, то он присылает ответ со статусом **ok**.

Пример успешного ответа на команду **show-customer-screen**:

```
{
    "reply": "show-customer-screen",
    "status": "ok"
}
```

Если была отправлена команда **show-customer-screen**, а предыдущее сообщение ещё отображается или находится в очереди на показ, то он будет обновлён на новый. При этом будет возвращён статус ответа **updated**.

```
{
    "reply": "show-customer-screen",
    "status": "updated"
}
```

Если терминал успешно принял команду, но не может в данный момент отобразить сообщение, команда ставиться в очередь и возвращается ответ со статусом **queued** (помещен в очередь). Как только терминал освободиться от всех команд находящихся в очереди, сообщение будет отображено на экране.

Пример ответа на команду **show-customer-screen**, если терминал не может показать его в данный момент:

```
{
    "reply": "show-customer-screen",
    "status": "queued"
}
```

Если была отправлена команда **show-customer-screen** с пустым полем **text**, то показ текущего или находящегося в очереди сообщения немедленно прекращается.

```
{
    "command": "show-customer-screen",
    "text": ""
}
```

Пример успешного выполнения операции:

```
{
    "reply": "show-customer-screen",
    "status": "ok"
}
```

Если в момент вызова команды не показывается сообщение от пользовательского приложения или его нет в очереди, то операция отменяется:

```
{
    "reply": "show-customer-screen",
    "status": "cancelled"
}
```

Для запроса статуса обработки показа предыдущего ообщения от пользовательского приложения можно воспользоваться командой **show-customer-screen** с полем **type** со значением **get-status**. Возможные варианты ответа: **ok** (показан), **queued** (в очереди на показ), **cancelled** (команды на показ QR-кода ещё не поступало).

```
{
    "command": "show-customer-screen",
    "type": "get-status"
}
```

Пример ответа, если команды на показ ещё не поступало:

```
{
```

```
    "reply": "show-customer-screen",
    "type": "get-status",
    "status": "cancelled"
}
```

5.15. get-work-state – запросить текущее состояние работы терминала

Для запроса текущего состояния работы терминала необходимо отправить команду **get-work-state**.

Пример команды **get-work-state**:

```
{
    "command": "get-work-state"
}
```

Пример ответа:

```
{
    "reply": "get-work-state",
    "status": "ok",
    "state": 0
}
```

Возможные значения **state** (тип `int`):

- `0` - ожидание команды от ККМ (основное рабочее состояние);
- `1` - идёт показ QR-кода;
- `2` - идёт обмен с сервером (отображается экран "Продажа временно недоступна");
- `3` - идёт обмен с платёжным приложением (отображаются экраны платёжного приложения);
- `4` - открыто сервисное меню.

Показ QR-кода командой **show-qr-code** начинается немедленно, если терминал находится в состояниях `0` или `1`. В остальных случаях показ QR-кода откладывается.

5.16. status - статус выполнения команды/операции

- **ok** - операция завершена успешно;
- **failed** - операция завершена с ошибкой;
- **cancelled** - операция отменена;
- **queued** - операция поставлена в очередь на выполнение;

- **updated** - операция обновила параметры предыдущей команды;
- **timeout** - превышено время ожидания завершения операции;
- **notimplemented** - запрашиваемая функция не реализована;
- **invalidarguments** - указаны недопустимые значения параметров операции;
- **communicationerror** - ошибка связи в т.ч. ошибка установки соединения по сети;
- **formaterror** - ошибка представления данных;
- **notauthorized** - недопустимый логин или пароль;
- **fileioerror** - ошибка доступа к файлу;
- **verificationfailed** - ошибка проверки контрольной суммы или сертификата
- **missingdata** - отсутствуют данные необходимые для выполнения операции;
- **systemerror** - системная ошибка;
- **transactionnotfound** - отменяемая транзакция отсутствует в логе;
- **notreversible** - транзакция не может быть отменена;
- **alreadyreversed** - транзакция уже отменена;
- **notapproved** - отменемая транзакция не одобрена;
- **emverror** - ошибка ядра EMV. В т.ч. ошибка инициализации;
- **notdetected** - ошибка обнаружения карты;
- **notallowedinterface** - использование интерфейса запрещено для данной операции;
- **tryagain** - ошибка чтения карты; повторить;
- **usechip** - требуется провести контактную emv транзакцию;
- **batchuploadfailed** - ошибка загрузки лога транзакций;
- **readerdisabled** - устройство чтения карт отключено;
- **preprocessingfailed** - ошибка предварительной обработки транзакции в ядре EMV;
- **readernotavailable** - устройство чтения карт отсутствует;
- **readererror** - ошибка устройства чтения карт;
- **tryanotherinterface** - ошибка чтения карты; используйте другой интерфейс.

В дополнение к статусу может передаваться его описание в поле **status-description**, например:

```
{
    "reply": "foo-bar",
    "status": "notimplemented",
    "status-description": "Неизвестная команда"
}
```

Внимание! Текстовое описание статусов **status-description** может меняться в разных версиях протокола и не имеет стабильного API.

6. Настройка работы с картами mifare в режиме POS

Для правильной работы считывателя карт `mifare` необходимо выполнение следующих требований:

- приложение `jpay` должно поддерживать работу с `mifare` картами (приложение должно быть собрано после 05.2024)
- в конфигурации `jpay` должен быть включён соответсвующий интерфейс `<interfaces><mifare enable="true"/>...</interfaces>`.
- для параметра `card-detect-timeout` должно быть установлено значение `inf`.

6.1. mifare-reader-enable – включение / отключение считывателя карт mifare

Для запуска / отключения считывателя карт mifare необходимо отправить команду:

```
{
    "command":"mifare-reader-enable",
    "enable":true/false
}
```

Пример ответа:

```
{
    "command":"mifare-reader-enable",
    "error":0,
    "errorMsg":""
}
```

Если не удалось запустить / отключить считыватель карт mifare, поле `error` будет иметь ненулевой код ошибки. Также будет добавлено опциональное поле `errorMsg`, которое будет содержать описание ошибки.

6.2. mifare-reader-get-last-read - получение результатов последнего считывания карты mifare

Для получения результатов последнего считывания карт mifare необходимо отправить команду:

```
{
    "command":"mifare-reader-get-last-read"
```

```
}
```

Примеры отправляемых ответов на команду `mifare-reader-get-last-read`:

- При успешном считывании `UID` карты `mifare`

```
{
    "command":"mifare-reader-get-last-read",
    "UID":"0011AABB",
    "datetime":"2023-11-16T17:34:08.689+03:00"
}
```

- При отсутствии считывания карты с момента запуска считывателя ответ не будет содержать данных о карте и времени считывания:

```
{
    "command":"mifare-reader-get-last-read"
}
```

6.3. mifare-reader-clear-last-read - удаление данных об уже считанной карте (индикатор того, что данные были обработаны)

Для удаления данных считывания карты `mifare` необходимо отправить команду:

```
{
    "command":"mifare-reader-clear-last-read",
    "UID":"0011AABB",
    "datetime":"2023-11-16T17:34:08.689+03:00"
}
```

- Пример ответа, если данные последнего считывания карты равны отправленным:

```
{
    "command":"mifare-reader-clear-last-read",
    "error": 0,
    "errorMsg": ""
}
```

- Пример ответа, если отправленные данные отличаются от результатов последнего считывания:

```
{
```

```
    "command":"mifare-reader-clear-last-read",
    "error":0,
    "errorMsg":""
}
```

Поле `errorMsg` содержит описание ошибки. Это поле опционально и будет добавлено в случае ошибки.

7. Формат транспортного пакета

7.1. Взаимодействие по сетевым протоколам

При взаимодействии по сети между всеми программами (VEND, ПП, JPAY) используется формат пакета, изображенный на Рисунке А.

Рисунок А. Формат пакета при взаимодействии по сетевым портам.

размер пакета (4 байта)
тело пакета (переменная длина)

Каждый пакет начинается с 4 двоичных байтов в которых указана длина пакета без учета этих 4х байтов. Первый из байтов длины старший.

Тело пакета – это основной текст сообщения. Для программ ПП и JPAY тело пакета – это сообщение в формате XML. Для программы VEND тело пакета – это сообщение в формате JPAY.

7.2. Взаимодействие через последовательный интерфейс

Программа VEND предусматривает взаимодействие через последовательные интерфейсы:

- либо RS-232 (com-порт)
- либо UART

Взаимодействие через последовательные интерфейсы предусматривается только для программы VEND (для программ ПП и JPAY – только сетевое подключение). При этом перечень и структура передаваемых пакетов отличаются от тех, которые используются при работе с программой VEND по сети в связи с необходимостью обеспечивать целостность принимаемых и передаваемых пакетов. Структура пакета при передаче через последовательный порт изображена на Рисунке 3 (см. далее).

Предусмотрено 2 типа пакетов:

- Пакет типа **«сообщение»** - в рамках этого пакета передаются команды и результаты исполнения команд. Пакеты такого типа всегда содержат тело сообщения с данными.
- Пакеты типа **«запрос статуса»**. Такие пакеты состоят из одного байта с кодом 05 и не содержат ничего кроме этого одного байта. Пакеты **«запрос статуса»** может отправлять только приложение клиента в адрес программы VEND. Программа VEND в ответ должна вернуть статус последней исполняемой команды.

7.2.1. Параметры передачи данных через последовательный интерфейс

При передаче информации через последовательный интерфейс по умолчанию используются следующие параметры:

Baud rate	38400
Data	8 bits
Parity	None
Stop	1 bit
Flow control	None

7.2.2. Пакеты типа «сообщение» при передаче через последовательный интерфейс

Структура пакета типа «Сообщение» изображена на Рисунке В. Пакеты данного типа могут отправляться как из программы-клиента, так и из программы VEND. Клиент будет отправлять команды на исполнение. Программа VEND использует пакеты типа «сообщение» для передачи результатов последней команды.

Рисунок В. Структура пакета типа «сообщение» при передаче через последовательный интерфейс.

признак начала пакета (1 байт)
hex(размер пакета) (8 байт)
hex(идентификатор команды) (16 байт)
тело пакета (переменная длина)
hex(CRC) (4 байта)

Пакет состоит из 5 блоков:

- **Признак начала пакета**
  В качестве признака начала пакета используется символ с кодом 01 – «начало передачи». Символ с кодом 01 допускается передавать только в блоке «начало пакета». Если система принимает символ с кодом 01 (или 05 для сообщений типа «запрос статуса»), приём предыдущего пакета прекращается и начинается приём нового пакета.
- **Размер пакета**
  Размер пакета кодируется числом unsigned integer (4 байта). Однако, при передаче байты

преобразуются в HEX-формат и строка увеличивается вдвое. Таким образом, для передачи размера пакета используется 8 байт. В начале передаются старшие байты, затем – младшие.

- **Идентификатор команды**
  В качестве идентификатора команды рекомендуется использовать timestamp с микросекундами. Само значение идентификатора записывается в 8 байт. Однако, при преобразовании в HEX идентификатор преобразуется в 16 байт.
  Идентификатор команды формирует клиент когда отправляет какую-то команду в адрес программы VEND (например, команда **«loadmasterkeys»**). Все ответы от программы VEND на данную команду будут содержать тот же самый идентификатор команды. Т.е. пока от клиента не поступит новая команда – идентификатор будет сохраняться. До момента, пока от клиента не поступила какая-либо команда для ПП, ответ на запрос статуса будет иметь идентификатор "0000000000000000".
- **Тело пакета**
  Это основной текст сообщения. Для программы VEND тело пакета – это сообщение в формате JSON. Внутри JSON не допускается использовать символы с кодами 01 и 05. Эти поля должны передаваться в виде HEX-строк.
- **Контрольная сумма**
  Контрольная сумма, вычисляемая на основе алгоритма CRC-16-CCITT. При вычислении контрольной суммы учитываются значения блоков «Признак начала пакета», «Размер пакета», «Идентификатор команды», «Тело пакета». При этом при вычислении контрольной суммы учитываются именно те байты, которые фактически были переданы через COM-порт. Т.е. если поле передаётся в формате HEX, то и контрольная сумма считается от значения в формате HEX.

7.2.3. Пакеты типа «запрос статуса» при передаче через последовательный интерфейс

В отличие от соединения по сети (через сокет), программа VEND не должна автоматически пересылать каждое сообщение от программы ПП в «вышестоящую» программу-клиента. Все сообщения со стороны VEND в сторону программы-клиента осуществляются по запросу – после получения сообщения «запрос статуса».

Структура сообщения «Запрос статуса» изображена на Рисунке С. Фактически, это сообщение состоит из одного единственного байта с кодом 05. Никаких контрольных сумм и тела сообщения не предусматривается.

Рисунок С. Структура пакета типа «запрос статуса» при взаимодействии через последовательный интерфейс.

байт с кодом 05 (1 байт)

В ответ на сообщение «запрос статуса» программа VEND должна прислать:

- **Результат выполнения последней команды**, если от программы ПП ранее был получен ответ на команду.
- Последнее сообщение **«display»**, если от программы ПП поступило сообщение **«display»**, но не пришел результат команды. Т.е. результат исполнения команды – более приоритетен и если он уже поступил, то никакие сообщения **«display»** уже не нужны.
- Сообщение **«keep-alive»** если от ПП не пришло никаких сообщений **«display»** и никаких результатов исполнения команды (при коммуникации через com-порт сообщение **«keep-alive»** отправляется для любых команд, не только для транзакций).

Все ответы должны содержать идентификатор команды, который поступил вместе с текущей исполняемой командой. Пока исполняется команда – идентификатор команды во всех ответах будет один и тот же. Когда приходит новая команда от клиента – тогда меняется и идентификатор команды.

7.2.4. Процесс передачи данных через последовательный интерфейс

Процесс работы через последовательный интерфейс выглядит следующим образом:

1. Клиент формирует команду, которую необходимо исполнить и отправляет в программу VEND через последовательный интерфейс.
2. Программа VEND получает команду и начинает её исполнять. При этом все служебные сообщения «поднесите карту», «уберите карту» обрабатываются на уровне программы ПП. В то же время все служебные сообщения для информации транслируются в программу VEND.
3. Клиент последовательный интерфейс может в цикле отправлять запросы текущего статуса исполнения последней команды. Частота опросов – по усмотрению клиента.
4. Если клиент хочет прервать текущую исполняемую транзакцию, необходимо передать сообщение { "keep-alive-status"="cancelled" } с указанием идентификатора текущей исполняемой команды в заголовке пакета. При этом транзакция прерывается не моментально и реальный результат исполнения транзакции (прервалась или не прервалась) всё равно нужно запрашивать через сообщения «запросы статуса».
5. Результаты исполнения последней команды будут храниться в программе VEND до тех пор, пока клиент не инициирует новую команду. Т.е. клиент должен получить информацию о результатах команды до того как инициирует очередную команду.

Такой подход более удобен в том смысле, что клиенту не нужно реализовать постоянный мониторинг и анализ всех входящих пакетов, поступающих через последовательный порт. Приложение клиента может в любой момент запросить статус последней «порученной к исполнению» команды и получить ответ из которого понятно – завершилась команда или нет и результат исполнения команды (успешно или не успешно).

Наличие двух «специальных символов» с кодами 01 и 05, являющихся признаками «начала пакета» облегчает процесс «корректировки ситуации» в случае возникновения сбоев при передаче.

8. Параметры, необходимые для фискализации

8.1. fiscalParams

Название	Тип данных	Примечание
amount	float	Сумма чека
content	JSON object	Содержимое чека
operationMode	int	Операционный режим, в котором был пробит чек. Битовая маска, по умолчанию 0

8.2. Содержимое чека (content)

Название	Тип данных	Примечание
type	int	Типы чека
checkClose	JSON object	Параметры закрытия чека
positions	JSON array	Позиции чека (массив)
automatNumber	string	Номер автомата
settlementAddress	string	Адрес установки
settlementPlace	string	Место расчётов

8.3. Позиции чека (positions)

Название	Тип данных	Примечание
quantity	float	Количество предмета расчёта (максимум 3 знака после запятой)
price	float	Стоимость единицы предмета расчёта
tax	int	Типы НДС
text	string	Наименование предмета расчёта(товара)
paymentMethodType	int	Способы расчёта
paymentSubjectType	int	Типы предмета расчёта

8.4. Параметры закрытия чека (checkClose)

Название	Тип данных	Примечание
payments	JSON array	Оплаты (массив)
taxationSystem	int	Системы налогообложения

8.5. Оплаты (payments)

Название	Тип данных	Примечание
type	int	Типы оплаты
amount	float	Сумма платежа типом оплаты
acquiringData	JSON object	Эквайринговый слип

8.6. Системы налогообложения

Значение	Название
1	Общая, ОСН
2	Упрощенная доход, УСН доход
4	Упрощенная доход минус расход, УСН доход - расход
8	Единый налог на вмененный доход, ЕНВД
16	Единый сельскохозяйственный налог, ЕСН
32	Патентная система налогообложения, Патент

8.7. Операционный режим

Номер бита	Признак
0	Признак шифрование данных
1	Признак автономного режима
2	Признак автоматического режима
3	Признак расчетов за услуги
4	Признак формирования только БСО
5	Признак ККТ для расчетов только в Интернет

8.8. Типы чека

Значение	Описание
1	Приход
2	Возврат прихода
3	Расход
4	Возврат расхода

8.9. Типы НДС

Значение	Название
1	20%
2	10%
3	Ставка расч. 20/120
4	Ставка расч. 10/110
5	0%
6	Без НДС

8.10. Способы расчёта

Значение	Название
1	Предоплата 100%
2	Частичная предоплата
3	Аванс
4	Полный расчёт
5	Частичный расчёт и кредит
6	Передача в кредит
7	Оплата кредита

8.11. Типы предмета расчёта

Значение	Название
1	Товар
2	Подакцизный товар
3	Работа
4	Услуга
5	Ставка азартной игры
6	Выигрыш азартной игры
7	Лотерейный билет
8	Выигрыш лотереи
9	Предоставление РИД
10	Платёж
11	Агентское вознаграждение
12	Выплата

Значение	Название
13	Иной предмет расчёта
14	Имущественное право
15	Внереализационный доход
16	Иные платежи и взносы
17	Торговый сбор
18	Курортный сбор
19	Залог
20	Расход
21	Взносы на обязательное пенсионное страхование ИП
22	Взносы на обязательное пенсионное страхование
23	Взносы на обязательное медицинское страхование ИП
24	Взносы на обязательное медицинское страхование
25	Взносы на обязательное социальное страхование
26	Платёж казино

8.12. Типы оплаты

Значение	Название
0	Наличные
1	Электронные
13	Аванс
14	Кредит
15	БСО (Бланк строгой отчётности)