Проект

Протокол взаимодействия с терминалом оплаты AQSI Cube

Оглавление

[§1. Платёжный функционал на устройствах AQSI CUBE 3](#_Toc32676478)

[§2. Наборы команд, поддерживаемые в приложениях JPAY, ПП, VEND 5](#_Toc32676479)

[§3. JPAY 7](#_Toc32676480)

[3.1. Активация JPAY 8](#_Toc32676481)

[§4 Платёжное Приложение 9](#_Toc32676482)

[4.1. Особенности обработки команд, требующих авторизации 9](#_Toc32676483)

[4.2. Особенности обработки команд с промежуточными сообщениями 10](#_Toc32676484)

[4.3. Специальная обработка команд JPAY для транзакций 11](#_Toc32676485)

[4.3. Платёжное приложение и активации JPAY 12](#_Toc32676486)

[4.4. Обработка ситуаций «обрыва» соединений 12](#_Toc32676487)

[§5. Программа VEND 13](#_Toc32676488)

[5.1. updateconfiguration – загрузка конфигурации с сервера 13](#_Toc32676489)

[5.2. loadmasterkey – загрузка мастер-ключей 14](#_Toc32676490)

[5.3. loadworkkeys – загрузка рабочих ключей 14](#_Toc32676491)

[5.4. testconnection – проверка связи с сервером 15](#_Toc32676492)

[5.5. getparameters – сведения о приложении и терминале 15](#_Toc32676493)

[5.6. transaction – транзакции 16](#_Toc32676494)

[5.7. report – отчет 21](#_Toc32676495)

[5.8. settlement– сверка итогов 24](#_Toc32676496)

[5.9. clear– очистка журнала 25](#_Toc32676497)

[5.10. runreset– переинициализация 26](#_Toc32676498)

[5.11. keep-alive– сообщение об увеличении времени ожидания 26](#_Toc32676499)

[5.12. Коды ошибок 27](#_Toc32676500)

[§6. Формат транспортного пакета 29](#_Toc32676501)

[6.1. Взаимодействие по сетевым протоколам 29](#_Toc32676502)

[6.2. Взаимодействие через последовательный интерфейс 30](#_Toc32676503)

[6.2.1 Параметры передачи данных через последовательный интерфейс 30](#_Toc32676504)

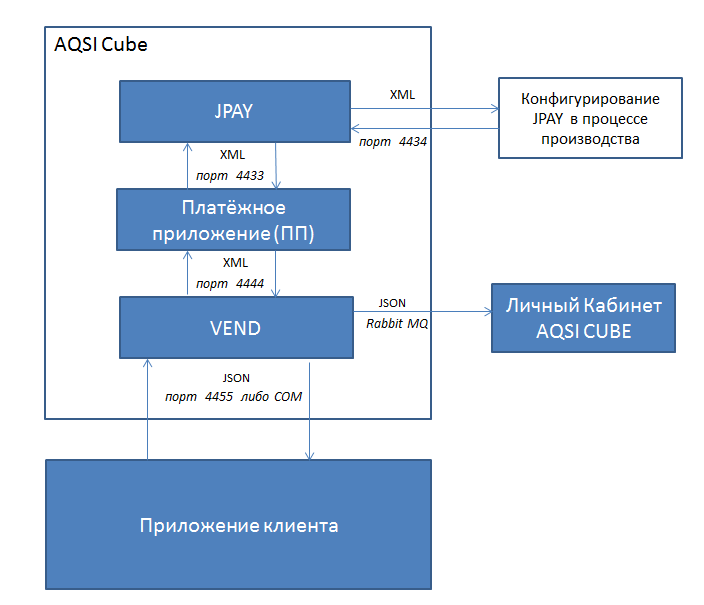
[6.2.2 Пакеты типа «сообщение» при передаче через последовательный интерфейс 30](#_Toc32676505)

[6.2.3 Пакеты типа «запрос статуса» при передаче через последовательный интерфейс 32](#_Toc32676506)

[6.2.4 Процесс передачи данных через последовательный интерфейс 33](#_Toc32676507)

# §1. Платёжный функционал на устройствах AQSI CUBE

На **Рисунке 1** изображена схема взаимодействия приложений AQSI CUBE между собой и с внешними устройствами.

**Рисунок 1. Схема взаимодействия платёжных приложений при выполнении платёжных операций.**

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

В процессе выполнения платёжных операций фигурируют следующие приложения:

|  |  |
| --- | --- |
| **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. |
| **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* |
| **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) | + | + | + |
| **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.* |
| **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 | + | + | + |
| **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 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.* |
| **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 приложение. | - | - | + |

# §3. JPAY

Взаимодействие с программой JPAY осуществляется при помощи специализированного протокола JPAY, описанного в документе <https://gitlab.aqsi.ru/aqsi-acquiring/doc/blob/master/jpay/readme.md>

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 (ПП) принимает «поручения» на проведение эквайринговых операций в формате XML по аналогии с программой JPAY. Приём команд осуществляется в ОДНОПОЛЬЗОВАТЕЛЬСКОМ режиме через сокет. Для коммуникации используется сетевой порт 4444.

## 4.1. Особенности обработки команд, требующих авторизации

Платёжное Приложение должно «облегчить» работу с JPAY. В частности, во все команды, требующие авторизации, ПП должен автоматически «подставлять» дополнительные теги для обеспечения авторизации. А перед началом выполнения таких команд – ПП 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** |
| **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 ПП должна в XML-команду добавить дополнительный тег <password>.

**Пример пакета из программы VEND в программу ПП:**

<report>

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

</report>

**Пример «дополненной» команды из программы ПП в JPAY:**

<report>

<password>12345678</password>

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

</report>

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

**Пример пакета из программы VEND в программу ПП:**

<loadmasterkeys>

*в программе ПП*

</loadmasterkeys>

**При получении такого сообщения ПП должна**

**запросить авторизационный токен:**

<login>

<password>12345678</password>

</login>

**Пример ответа от JPAY:**

<login>

*в программе JPAY*

<status>ok</status>

<token>DE9773A8CB888560AB0F89C07623FE03</token>

</login>

**После этого ПП должна «транслировать» команду loadmasterkeys:**

<loadmasterkeys>

<token>DE9773A8CB888560AB0F89C07623FE03</token>

</loadmasterkeys>

Таким образом, одна команда в программе ПП превращается в три команды в программе JPAY.

Можно токен не запрашивать постоянно, а хранить в какой-то переменной и использовать многогратно. 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.

## 4.2. 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 ПП 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 ПП 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 ПП 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.

## 4.3. 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 ПП 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 ПП следующим образом:

* сообщения **keep-alive** должны увеличивать тайм-аут ожидания в ПП и далее передаваться в приложение VEND
* сообщения **display** должны обрабатываться внутри ПП
* сообщения **dex** не желательны в приложении ПП и их нужно «отключать» на уровне конфигурации EMV. Если в ПП поступило сообщение dex – нужно прервать транзакцию и сохранить информацию о причине прерывания транзакции (от приложения JPAY поступило сообщение dex, которое программа ПП 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.

## 4.3. 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.

## 4.4. 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>

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

В случае, если клиент ПП «оборвал» соединение с ПП и сразу после этого в ПП пришло сообщение об успешном или не успешном завершении текущей выполняемой транзакции, у ПП не будет возможности передать полученный результат в «вышестоящее» приложение VEND. В этом случае ПП 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
* **ccert** - 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"

}

| **Пераметр** | **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. |

Пример ответа:

{

"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 экваринговых слипов из формата JPAY к формату транспортной модели. Преобразование осуществляется в соответствии со следующей таблицей:

| **Поле транспортной модели** | **Тип** | **Поле модели**  **JPAY** | **Пояснение** |
| --- | --- | --- | --- |
| id | UUID(v4) | - | Уникальный идентификатор. При конвертации необходимо сгенерировать уникальное значение UUID. |
| 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 |
| 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 | varchar(64) | ???? | Код ошибки при оплате.  В документации нет информации о структуре XML при ошибках, нужно смотреть что там возвращается. |
| errorMessage | varchar(1024) | ???? | Описание ошибки при оплате  В документации нет информации о структуре XML при ошибках, нужно смотреть что там возвращается. |
| text | text | - | Текст слипа |
| cardholder | varchar(255) | cardholder | Имя держателя карты |
| acquirerAgentName | varchar(255) | acquirer-tag | Посредник до банка эквайера |
| cashlessType | enum | При оплате картой указывается значение “2” | [Тип безналичной оплаты (1=Безналичные, 2=картой, 3=QR код)](https://gitlab.aqsi.ru/aqsi/documentation/blob/test/references.md) |

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.

## 5.9. 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. Коды ошибок

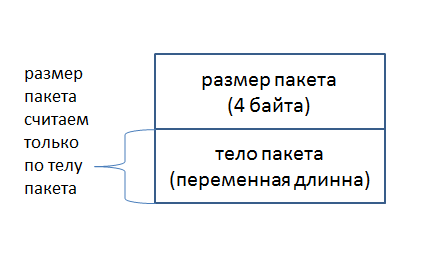
* **ok** - операция завершена успешно;
* **failed** - операция завершена с ошибкой;
* **communicationerror** - ошибка связи в т.ч. ошибка установки соединения по сети;
* **notimplemented** - запрашиваемая функция не реализована;
* **ormaterror** - ошибка представления данных;
* **notauthorized** - неопустимый 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;
* **timeout** - превышено время ожидания завершения операции;
* **invalidarguments** - указаны недопустимые значения параметров операции;
* **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;
* **cancelled** - операция отменена ;
* **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.

# §6. Формат транспортного пакета

## 6.1. Взаимодействие по сетевым протоколам

При взаимодействии по сети между всеми программами (VEND, ПП, JPAY) используется формат пакета, изображенный на Рисунке 2.

**Рисунок 2. Формат пакета при взаимодействии по сетевым портам.**



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.

## 6.2. 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.

### 6.2.1 Параметры передачи данных через последовательный интерфейс

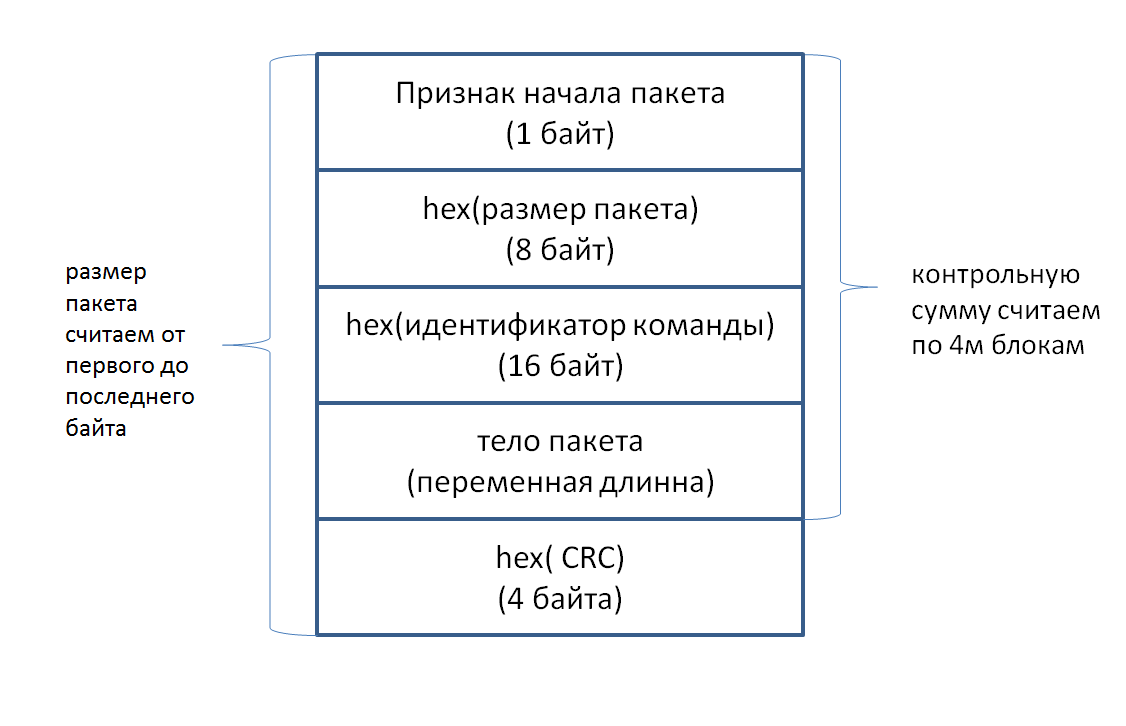
При передаче информации через последовательный интерфейс по умолчанию используются следующие параметры:

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

### 6.2.2 Пакеты типа «сообщение» при передаче через последовательный интерфейс

Структура пакета типа «Сообщение» изображена на **Рисунке 3**. 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 3. Структура пакета типа «сообщение» при передаче через последовательный интерфейс.**



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.

* **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.

### 6.2.3 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 4. Фактически, это сообщение состоит из одного единственного байта с кодом 05. Никаких контрольных сумм и тела сообщения не предусматривается.

**Рисунок 4. Структура пакета типа «запрос статуса» при взаимодействии через последовательный интерфейс**



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.

### 6.2.4 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.