Руководство программиста

Описание протоколов управления

программного комплекса QKkmServer:
программного обеспечения для управления фискальными регистраторами АТОЛ и Штрих-М.

19.10.17

Ростов-на-Дону

Содержание

Краткое описание и предназначение QKkmServer

Кассовый сервер QKkmServer предназначен для решения широкого спектра задач автоматизации, в которых необходимо обеспечить работу с фискальными регистраторами.

QKkmServer предлагает мощные интеграционные возможности для быстрого внедрения печати чеков в самые разные проекты.

Поддерживается широкий спектр моделей фискальных регистраторов и ОнЛайн касс производства Штрих-М, АТОЛ и RRElectro(подразделение Штрих-М).

Кассовый сервер имеет модульную структуру:

1. Для работы с оборудованием предназначен модуль QKkmServer, который должен располагаться на том ПК, к которому физически подключен кассовый аппарат. Выполнен в виде службы/демона. Принимает входящие подключения по TCP на порт 20000 (настраивается). Формат команд — XmlAPI.
2. SuperVisor (СуперВизор, СВ) — графический клиент для QKkmServer. Производит:
    - постоянный мониторинг кассы и кассового ядра
    - отслеживает накопления денежных сумм в регистрах ККМ
    - ведёт журнал операций в виде базы данных
    - предоставляет функции «Тест драйвера» - можно послать любую команду кассовому серверу
    - функция «Произвольный чек»
    - поддержка печати чеков с использованием файлового протокола управления FileAPI
    - поддержка получения заданий на печать путем опроса внешнего сервера по WebAPI

Описание каждой из указанных функций будет предоставлено в соответствующем разделе данного руководства.

Установка

Версии

В настоящее время основными поддерживаемыми системами являются:

- MS Windows
- Ubuntu 14.04 i386 (x86)
- Ubuntu 14.04 i686 (x64)
- CentOS7 x64
- Raspbian, x86

  Получить последние версии установочных дистрибутивов можно на странице загрузки.

Проект разбит на 2 части: кассовое ядро и графический клиент, т. к. архитектурно и логически они могут располагаться на разных ПК.
Установка производится стандартным для выбранной ОС методом.

Настройка QKkmServer

QKkmServer является консольным приложением с режимом запуска «Служба» / «Демон» / «Сервис».

В ОС Windows файл настроек qkkmserver.cfg.xml находится в папке, куда была произведена установка ПО.

В ОС Linux файл настроек находится в /etc/AzsKit2

Настройка АТОЛ

```
<QKkmServer   tcpPort="TCP_ПОРТ">
  <connection
     protocol="ВЕРСИЯ_ПРОТОКОЛА"
     pluginLibrary="libplugin_atol54_qkkmserver.so"
         model="ИД_МОДЕЛИ_ККМ"
         port="НОМЕР_COM_ПОРТА"
  />
</QKkmServer>
```

TCP_ПОРТ — порт сервера, на который будет открыт для входящих подключений. По-умолчанию 20000.
ВЕРСИЯ_ПРОТОКОЛА — может принимать одно из 2-х значений: 2 или 3. Зависит от аппаратных настроек ККМ. Как правило после февраля 2017 года ККМ АТОЛ идут настроенные на 3-ю версию протокола. Значение по-умолчанию: 3.

ИД_МОДЕЛИ_ККМ — числовой код модели. Требуется обязательно указать, т. к. на основании этой информации по-разному обрабатываются некоторые команды
Значение по-умолчанию: 47
Указывать ОБЯЗАТЕЛЬНО!

30 < FPrint-02K / ЕНВД
31 < FPrint-03K / ЕНВД
32 < FPrint-88K / ЕНВД
35 < FPrint-5200K / ЕНВД
47 < FPrint-55 ПТК / K / ЕНВД
52 < FPrint-22 ПТК / K / ЕНВД
51 < FPrint-11 ПТК / ЕНВД
53 < FPrint-77 ПТК / ЕНВД
54 < FPrintPay-01ПТК
57 < АТОЛ 25Ф
61 < АТОЛ 30Ф
62 < АТОЛ 55Ф
63 < АТОЛ 22Ф (АТОЛ FPrint-22ПТК)
64 < АТОЛ 52Ф
65 < АТОЛ 03Ф
67 < АТОЛ 11Ф
68 < АТОЛ 02Ф
69 < АТОЛ 77Ф
71 < АТОЛ СМ-02 ПТК / ЕНВД
72 < АТОЛ 90Ф
73 < Принтер документов FPrint-30 для ЕНВД
74 < АТОЛ СТ2Ф
75 < АТОЛ 60Ф
77 < АТОЛ 42ФС
78 < АТОЛ 15Ф

port – *необязательный атрибут*, т. к. производится опрос всех портов. Иногда может понадобиться, тогда в ОС Windows он задает имя COM-порта (COM1, COM2...) , а в ОС Linux – имя устройства из /dev – ttyACM1, например.

boudrate –

```
boudrate
```

Настройка ШТРИХ-М

```
<QKkmServer   tcpPort="TCP_ПОРТ">
  <connection
     pluginLibrary="libplugin_atol54_qkkmserver.so"
```

```
  />
</QKkmServer>
```

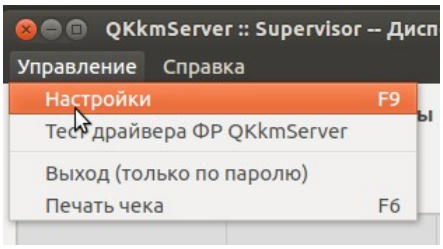
TCP_ПОРТ — порт сервера, на который будет открыт для входящих подключений. По-умолчанию 20000.

Необязательные параметры
portName — имя или путь к порту устройства.
boudrate – скорость обмена данными с портом

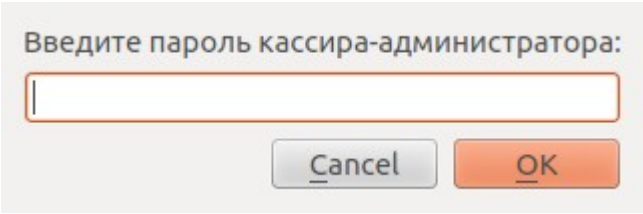
По-умолчанию фискальный регистратор ищется автоматически путем перебора всех доступных портов и их скоростей обмена из числа допустимых.

Настройка SuperVisor

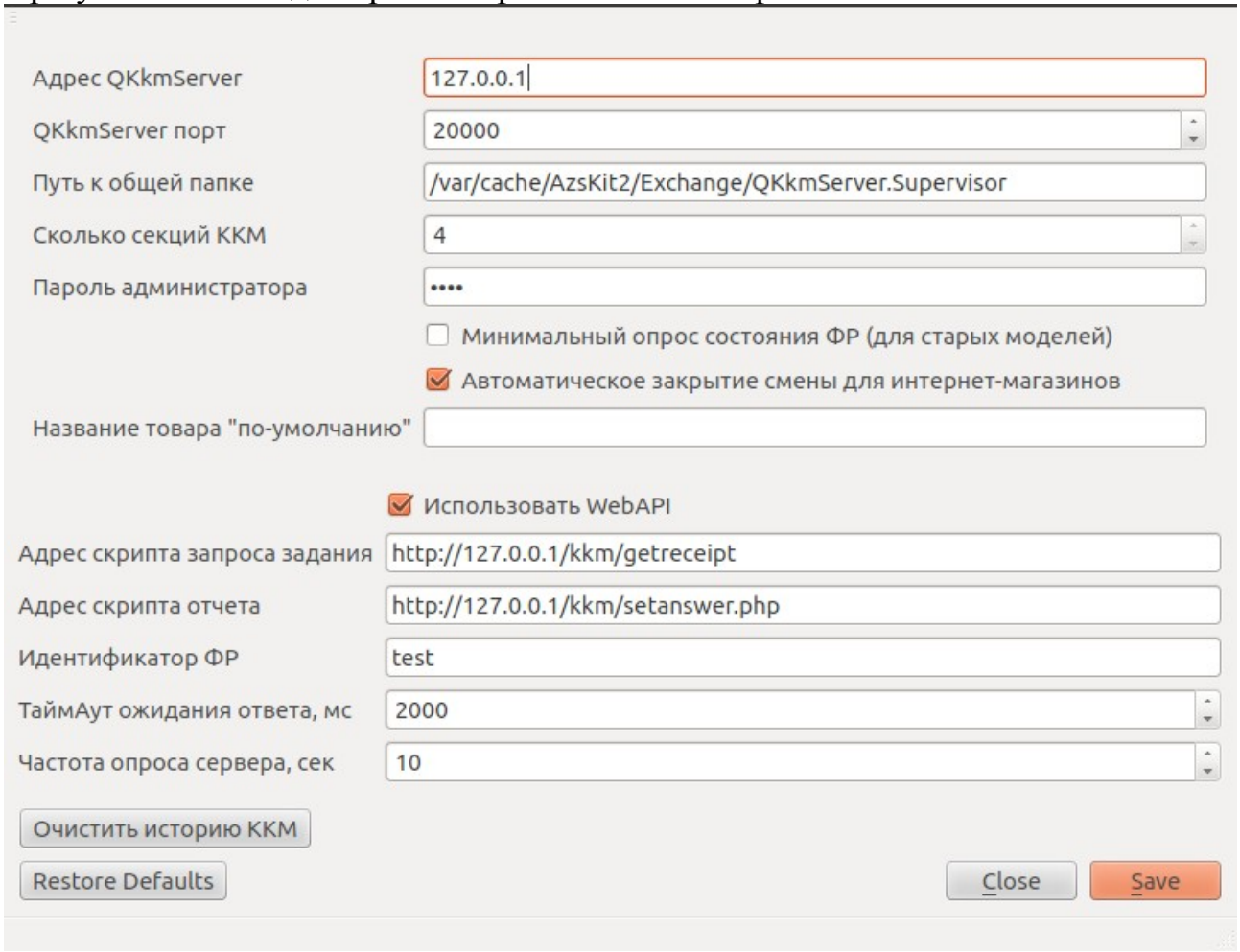
Для доступа к настройкам необходимо в меню выбрать пункт Управление → Настройки (или нажать F9).



Доступ к настройкам ограничен паролем. Значение пароля «по-умолчанию» «0000» (четыре нуля).



При успешном вводе пароля откроется окно настроек



Разберем подробно каждую настройку.

АДРЕС QKkmServer – сетевой адрес или имя компьютера, на котором

установлен кассовый сервер QKkmServer. Так же указывается QKkmServer порт.
ПУТЬ К ОБЩЕЙ ПАПКЕ – путь к директории, содержимое которой будет отслеживаться на предмет обнаружения файлов-заданий в формате FileAPI. Внешняя управляющая система, соответственно, должна размещать файл-задание в этой же директории.

СКОЛЬКО СЕКЦИЙ ККМ – программа Супервизор в фоновом режиме отслеживает состояние кассы. Так же отслеживаются накопления по денежным регистрам. Если в учете используется только одна секция, то имеет смысл ограничить опрос именно этой, первой, секцией, т. к. в остальных секция по причине отсутствия деятельности, накоплений не будет. Программа Супервизор в свою очередь, будет совершать меньшее число запросов к кассе.

ПАРОЛЬ АДМИНИСТРАТОРА – пароль на доступ к некоторым функциям программы: пробитие произвольного чека, настройки.

МИНИМАЛЬНЫЙ ОПРОС СОСТОЯНИЯ ФР – отслеживается только минимальный набор параметров состояния кассы. Опрос денежных регистров не производится.

АВТОМАТИЧЕСКОЕ ЗАКРЫТИЕ СМЕНЫ ДЛЯ ИНТЕРНЕТ-МАГАЗИНОВ – как правило интернет-магазины (ИМ) ведут торговлю используя электронную доставку чеков. Соответственно за кассой никто не следит и она должна работать в автоматическом режиме. С введением 54-ФЗ возник вопрос: а кто будет следить за закрытием и открытием смены (обязательные операции!). Данная опция позволяет автоматически закрывать смену как только касса переходит в состояние «Смена открыта. 24 часа кончились». Так же смена автоматически открывается в том случае, если касса находится в состоянии «Закрытая смена».

НАЗВАНИЕ ТОВАРА «ПО-УМОЛЧАНИЮ» – в программе присутствует возможность печати произвольного чека. Часто данной возможностью пользуются интернет-магазины у которых основная часть названия товара - «Реализация по накладной №....». Именно эту неизменяемую часть наименования можно задать в данной настройке и упростить работу кассира.

ИСПОЛЬЗОВАТЬ WebAPI – задействует механизм работы по WebAPI. Будет производиться регулярный опрос сервера на предмет получения заданий для печати чеков.

ИДЕНТИФИКАТОР ФР – служит для уникальной идентификации рабочего места кассира на центральном сервере при использовании WebAPI. Данный идентификатор передается на сервер как GET-параметр kkm_id.

ТАЙМАУТ ОЖИДАНИЯ ОТВЕТА, МС – время после установления

соединения, в течении которого сервер должен передать данные о задании. Если канал связи очень плохой, то значение можно увеличить до 5000мс (5 секунд). Для среднестатистических каналов связи хватает 2000мс для получения информации в объеме 1-2 КБайта (с учетом всех заголовков HTTP).
Если ответ от сервера в указанное время получен не будет, то Супервизор закроет соединение в аварийном режиме.

ЧАСТОТА ОПРОСА СЕРВЕРА, СЕК – интервал времени между завершением одного опроса сервера и началом другого.

> **ВНИМАНИЕ!**
>
> Большая часть настроек начинает действовать только при перезапуске программы.
>
> Для применения настроек необходимо нажать на кнопку «Save» и перезапустить программу.

С версии 5.17.09.26 введена опция «Усиленное подтверждение получения ответа сервером».
Суть заключается в том, что не всегда сервер получив статус исполнения задания успешно помечает это у себя в диспетчере чеков. Возникают ситуации, когда данные приняты, но скрипт по какой-то причине отрабатывает на сервере не корректно. В итоге СВ постоянно опрашивает сервер на предмет новых заданий, а сервер возвращает задание с номером, который уже исполнен последним и на который СВ считает что ответ уже отправил.
Для выхода из данной ситуации от сервера в качестве подтверждения успешности и корректности обработки данных от СуперВизора требуется квитанция в формате

успех
<package id="ИД_ПАКЕТА" result="ok"></package>

внутренняя ошибка центрального сервера
<package id="ИД_ПАКЕТА" result="error"
error="ОПИСАНИЕ_ОПИБКИ_опционально"></package>

Таким образом, СуперВизор гарантированно будет знать о том, что центральный сервер получил и корректно обработал ответ о WebAPI задании на печать чека. Только в этом случае СВ прекратит попытки отправить ответ на центральный сервер.

ЗАПРЕТ ЗАПУСКА ВТОРОЙ КОПИИ ПРОГРАММЫ
Часто имели случаи некорректной работы ПО ввиду того, что пользователи запускали две и более копии программы СуперВизор от одного и того же

пользователя. Это вело к нарушению совместного доступа к ряду системных ресурсов и вело к нарушению работы программы.
Данная опция ограничивает запуск второй копии ПО.
Значение по-умолчанию: включена.

Основные форматы данных

Описание формата данных XmlAPI

XmlAPI используется для управления кассовым сервером QKkmServer.

Кассовый сервер ждет подключения на TCP-порт с передачей ему XML-сообщения.

Полученное XML-сообщение проходит проверку на соответствие формальным требованиям протокола и в случае успеха, преобразуется в команду управления соответствующего кассового оборудования (АТОЛ или Штрих-М).

Результат исполнения команды на ККМ так же оформляется в виде XML-сообщения и возвращается клиенту.

На этом сеанс связи прекращается по инициативе сервера, т. к. за одно подключение сервер обслуживает только одну команду.

СТРУКТУРА КОМАНДЫ

```
<ControlProtocol messageType="request">
КОМАНДА[+параметры]
</ControlProtocol>
```

СТРУКТУРА ОТВЕТА

```
<ControlProtocol messageType="answer">
<error id="КОД_СТАТУСА" text="ОПИСАНИЕ_СТАТУСА"/>
<НАЗВАНИЕ_КОМАНДЫ [ПАРАМЕТРЫ] />
</ControlProtocol>
```

Секция «error» является обязательной.
КОД_СТАТУСА - код ответа сервера. 0 — успешное выполнение команды. 255 – нет связи с ККМ. Остальные коды аппаратозависимы.

ОПИСАНИЕ_СТАТУСА - текстовое описание поля КОД_СТАТУСА. В случае успешного исполнения команды ОПИСАНИЕ_СТАТУСА=Ошибок нет.

НАЗВАНИЕ_КОМАНДЫ - название команды которая исполнялась сервером.
ПАРАМЕТРЫ - ответы кассы, если данное предусмотрено.

Пример команды XmlAPI «Отрезка чека»

```
<ControlProtocol messageType="request">
<CutCheck />
</ControlProtocol>

<ControlProtocol messageType="answer">
 <error text="Нет связи с устройством" id="255"/>
 <CutCheck  isFullCut="1"/>
</ControlProtocol>
```

Описание формата данных FileAPI

FileAPI используется для управления СуперВизором. В основе лежит принцип — одна строка в файле=одной команде на ККМ.

Структурно FileAPI представляет набор строк, где параметры разделяются символом «;» (точка с запятой).

В одном файле-задании может находиться неограниченное количество команд, однако рекомендуем придерживаться правила: «Один файл = один чек».

Каждая команда преобразуется в соответствующую команду в формате XmlAPI и отправляется на кассовый сервер QKkmServer.

Выполнение команд из файла-задания производится последовательно.

Если в результате исполнения очередной команды произошел сбой (код ответа отличен от 0), то дальнейшее выполнение команд прерывается, так как не гарантируется корректность работы и логичность последующих команд.

СуперВизор производит периодическое сканирование указанного в настройках каталога и ищет в нем файлы-задания по расширению *.txt или *.check.

При наличии в каталоге нескольких файлов, в очередь они заносятся в соответствии с сортировкой по имени файла. Поэтому, если критична очередность исполнения файлов-заданий, то следует именовать файлы примерно следующим образом:

```
task-YYYYMMDDhhmmsszzz.txt
```

Где YYYYMMDDhhmsszzz — текущая метка даты-времени.

Пример файла-задания FileAPI

```
B;0;Иванов Иван Иванович;
P;Просто строка текста;
SMDE;Товар 1;50000;1000;1;0;0;
SMDE;Товар 2;10000;2000;1;0;0;
TMDE;70000;1;
```

В результате будет распечатан чек с двумя товарами 500 руб x 1шт + 100руб x 2шт. Оплачен безналом (форма оплаты №1) 700 руб.

```
X;
```

Будет распечатан сменный отчет без гашения (X-отчет).

Формат управления WebAPI

В случае, когда имеется распределенная сеть кассовых узлов с централизованным управлением печатью чеков (например, интернет-магазины), целесообразно использовать возможности WebAPI.

Установленный у клиента экземпляр программы СуперВизор периодически (настраивается, обычно каждые 10 секунд) производит GET запрос к внешнему скрипту (адрес настраивается) на предмет наличия задания. При этом в качестве параметра kkm_id передается установленный в настройках идентификатор кассы. По этому признаку центральный сервер должен различать рабочие места кассиров.

В случае наличия задания, СуперВизор исполняет его, результат исполнения заворачивается в конверт-ответа и отправляется в POST-запросе. При этом СуперВизор будет повторять попытки отправки ответа до тех пор, пока не удостоверится (по статусу HTTP) что ответ сервером получен.

Пока у СуперВизора имеется недоставленный результат исполнения команды по WebAPI, внешние опросы сервера на наличие новых команд производиться не будут!

Структура команды

Сервер должен передать по запросу XML-ответ. Этот ответ именуется пакетом (package).

В случае отсутствия задания для исполнения, внешний сервер должен вернуть пустой пакет, с идентификатором=0

```
<package id='0'>
</package>
```

Получение данной XML-структуры говорит СВ о том, что связь с управляющим сервером есть, просто заданий для исполнения нет.

В случае, если у сервера имеются задания, то обязательно устанавливается целочисленный идентификатор пакета отличный от нуля и в поле data передаётся структура команд, соответствующая FileAPI, закодированная в BASE64.

```
<package id='ID'>
    <data>BASE64(FileAPI)</data>
</package>
```

т. е. управляющий сервер подготавливает текстовый файл-задание в соответствии с FileAPI, применяет к нему алгоритм кодирования BASE64 и помещает в XML-структуру.

Пример запроса

```
<package id="15">
<data>Y3V0X2NoZWNrOw==</data>
</package>"
```

В результате будет исполнена команда отрезки чека cut_check;

Обратите внимание, что программа СуперВизор отслеживает идентификаторы пакетов! Если с сервера будут приходить подряд несколько пакетов с одинаковым id, то исполнен будет только первый из них.

Настройки программы производятся в окне «Настройки». Доступ ограничен паролем (по-умолчанию «0000»).

Структура ответа СуперВизора

После исполнения команды (успешной или не успешной), СуперВизор пробует передать результат на сервер.

Вызывается POST скрипт Ответа. В составе URL передается идентификатор ККМ kkm_id и номер пакета package_id на который производится передача ответа.

В теле POST запроса передается переменная data, которая содержит BASE64URL-кодированный ответ FileAPI.

Общий формат строк:

```
КОД_КОМАНДЫ;КО_ОШИБКИ;ОПИСАНИЕ_ОШИБКИ;Base64(Ответ
сервера в XmlAPI);
```

Обратите внимание, что в случае возникновения ошибки, СВ останавливает работу на сбойной команде.

Усиленное подтверждение ответа от сервера

Настоятельно рекомендуется использовать опцию "Усиленное подтверждение ответа от сервера".

Что это такое? В некоторых случаях возникала ситуация, когда СуперВизор работая в режиме WebAPI чек успешно печатал, ответ на центральный сервер передавал, а вот этот самый центральный сервер ответ по какой-то причине не принимал/не обрабатывал/не использовал в работе.

В итоге СВ запрашивая новое задание от центрального сервера получает задание с номером пакета, которое уже исполнил и игнорирует его (защита от повторной печати чека).

Для противодействия это ситуации введена опция "Усиленное подтверждение ответа", которая заставляет СВ поверить в то, что центральный сервер действительно получил и корректно обработал ответ только в том случае, если получит в ответе квитанцию в формате:
успех

```
<package id="ИД_ПАКЕТА" result="ok"></package>
```

внутренняя ошибка центрального сервера

```
<package id="ИД_ПАКЕТА" result="error"
error="ОПИСАНИЕ_ОПИБКИ_опционально"></package>
```

Таким образом, СуперВизор гарантированно будет знать о том, что центральный сервер получил и корректно обработал ответ о WebAPI задании на печать чека. Только в этом случае СВ прекратит попытки отправить ответ на центральный сервер.

С версии 5.17.10.20 в СуперВизор введена в окне «Настройки» кнопка «Сброс серевого обмена», которая в случае подвисания обмена с Web-сервером очищает очередь обмена (номер задания и результат исполнения последнего задания).

Так же очередь автоматически очищается, если сервер 15 раз не принял ответ.

Алгоритм работы с очередью чеков

При построении системы очереди чеков нужно понимать, что ККМ — устройство крайне не надежное, в любой момент времени может произойти сбой: кончится чековая лента, пропадет питание, откажет связь с ККМ, произойдут некоторые внутрение процессы в ККМ, в результате чеко ККМ будет в состоянии, когда печать чеков невозможна.

Мы предлагаем придерживаться следующего алгоритма, контролирующего очередь чеков с контролем состояния ККМ.

1. Посылается пакет команд на исполнение
2. Сервер получает от СВ ответ.
3. Если присутствует ошибка (код ответа отличен от нуля), то следует запросить СТАТУС ККМ.

   Анализируем статус ККМ:

   - код=255 – отсутствие связи с ККМ. Можно предпринять N-повторов, после чего пометить чек как нераспечатанный.
   - проблема с закрытой сменой — открыть смену
   - смена 24 часа кончилась – закрыть и открыть смену
   - ККМ в состоянии «открытый документ» – аннулировать чек
   - ККМ в состоянии «после обрыва бумаги» – продалжить печать + анулировать чек

4. Перейти к п.1

Использование Токена сессии

В качестве средства авторизации добавлена поддержка работы токена.

Принцип работы:

1. Делается запрос по адресу ПОЛУЧИТЬ ЗАДАНИЕ или ПЕРЕДАТЬ ОТВЕТ. Если статус исполнения = 301, значит серверу для работы необходимо передать токен.
2. Производится обращение к скрипту ПОЛУЧИТЬ_ТОКЕН. Устанавливаются заголовки

   Content-Type = application/json-patch+json

   accept = text/plain

   Содержимое POST:

   {"userName": "ИМЯ_ПОЛЬЗОВАТЕЛЯ", "password": "ПАРОЛЬ"}
3. В ответ получаю значение accessToken
4. Полученное значение accessToken далее используется в работе всех скриптов:

   Authorization = Bearer ТОКЕН

   Content-Type = application/json
5.

Команды протоколов

Команда ГУДОК

XmlAPI

```
<ControlProtocol messageType="request">
<Beep />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<Beep />
</ControlProtocol>
```

FileAPI

```
отсутствует
```

В результате исполнения команды касса издает гудок / трель / пиликанье. Служит для привлечения внимания к кассе.

Команда ЗАПРОС ДЕНЕЖНОГО РЕГИСТРА

XmlAPI

```
<ControlProtocol messageType="request">
<getMoneyReg idReg="id"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<getMoneyReg idReg="id" Value="Значение"/>
</ControlProtocol>
```

FileAPI

```
m;id;
```

Получить накопления по указанному (id) номеру денежного регистра.

В ответе сервера накопления по регистру возвращаются в атрибуте Value.

Команда ЗАПРОС ОПЕРАЦИОННОГО РЕГИСТРА

XmlAPI

```
<ControlProtocol messageType="request">
<getOperReg idReg="id"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<getOperReg idReg="id" Value="Значение"/>
</ControlProtocol>
```

FileAPI

```
отсутствует
```

Получить накопления по указанному (id) номеру операционного регистра.

В ответе сервера накопления по регистру возвращаются в атрибуте Value.

Команда ОБРЕЗАТЬ ЧЕК

XmlAPI

```
<ControlProtocol messageType="request">
<CutCheck />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<CutCheck />
</ControlProtocol>
```

FileAPI

```
cut_check;
```

Обрезка чека.

Команда ПРОТЯЖКА ЛЕНТЫ

XmlAPI

```
<ControlProtocol messageType="request">
<FeedDocument Positions="КОЛ_ВО__СТРОК"
DocumentType="docCheck|docControl|docList"  />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<FeedDocument />
</ControlProtocol>
```

FileAPI

```
feed;КОЛ_ВО__СТРОК;
```

Протяжка ленты

docCheck – чековая лента

docControl – контрольная лента

docList – подкладной документ

Команда ОТЧЕТ БЕЗ ГАШЕНИЯ (X-отчет)

XmlAPI

```
<ControlProtocol messageType="request">
<XReport />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<XReport />
</ControlProtocol>
```

FileAPI

```
x;
```

В результате исполнения команды кассовый аппарат автоматически формирует отчет о работе за смену и выводит его на печать.

Команда ОТЧЕТ С ГАШЕНИЕМ (Z-отчет)

XmlAPI

```
<ControlProtocol messageType="request">
<ZReport />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<ZReport />
</ControlProtocol>
```

FileAPI

```
z;
```

В результате исполнения команды кассовый аппарат автоматически формирует отчет о работе за смену и выводит его на печать.

В отличии от команды ОТЧЕТ БЕЗ ГАШЕНИЯ, происходит закрытие кассовой смены.

Обязательно использовать команду необходимо так же в том случае, если фискальный регистратор перешел в режим «Смена открыта. 24 часа кончились».

Обращаем так же внимание на то, что формат печатаемого документа, не зависит от настроек программы QKkmServer. Формат документа жестко зафиксирован в настройках самого фискального регистратора.

Команда ОТЧЕТ ПО СЕКЦИЯМ (ОТДЕЛАМ)

XmlAPI

```
<ControlProtocol messageType="request">
<SectionsReport />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<SectionsReport />
</ControlProtocol>
```

FileAPI

```
rep_sections;
```

В результате исполнения команды кассовый аппарат автоматически формирует отчет о движении денежных средств с разделением на секции (отделы).

Команда ВНЕСЕНИЕ СРЕДСТВ

XmlAPI

```
<ControlProtocol messageType="request">
<CashIn Summa="СУММА_В_КОПЕЙКАХ" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<CashIn />
</ControlProtocol>
```

FileAPI

```
imde;СУММА_В_КОПЕЙКАХ;
```

Команда предназначена для фиксации внесения в денежный ящик средств, полученных не от реализации товаров/услуг.

Команда ИЗЪЯТИЕ СРЕДСТВ (Инкассация)

XmlAPI

```
<ControlProtocol messageType="request">
<CashOut Summa="СУММА_В_КОПЕЙКАХ" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<CashOut />
</ControlProtocol>
```

FileAPI

```
imde;-СУММА_В_КОПЕЙКАХ;
```

Команда предназначена для фиксации убытия денежных средств из кассового ящика. Обычно это связано с проведением инкассации.

Обращаем внимание, что в формате FileAPI вносится ОТРИЦАТЕЛЬНАЯ сумма.

Команда НЕОБНУЛЯЕМАЯ СУММА

XmlAPI

```
<ControlProtocol messageType="request">
<GetNotClearedSumma />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<GetNotClearedSumma Value="СУММА_В_КОПЕЙКАХ" />
</ControlProtocol>
```

FileAPI

```
a;
```

Опрашивается регистр ФР отвечающий за текущие накопления. Счетчик ведётся небнуляемым накоплением.

Для Штрих-М возвращает корректный результат только в режиме «Закрытая смена». В других режимах корректный результат не гарантируется.

Команда ОТКРЫТЬ ЧЕК

XmlAPI

```
<ControlProtocol messageType="request">
<OpenCheck
 type="ТИП_ЧЕКА"  operator="ИМЯ_КАССИРА"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<OpenCheck />
</ControlProtocol>
```

FileAPI

```
b;ТИП_ЧЕКА;ИМЯ_КАССИРА;
```

Открывает кассовый чек.

Тип чека определяется значением ТИП_ЧЕКА:

Sale – чек продажи. FileApi=0

ReturnSale – чек возврата продажи; FileAPI=1.

Buy – чек покупки. FileApi=2

ReturnBuy – чек возврата покупки; FileAPI=3.

ИМЯ_КАССИРА – позволяет задать имя кассира, которое будет передаваться в ОФД. Драйвер использует в работе кассира с номером 30 (Сист. администратор). Если ИМЯ_КАССИРА не ПУСТО, то происходит перезапись в настройках ККМ имени кассира, т. е. строка СИС.АДМИН. меняется на ИМЯ_КАССИРА.

Для ККМ Штрих-М команда не является обязательной, однако настоятельно рекомендуется её использовать в случае работы по FileAPI, т. к. на основании данной команды ПО СуперВизор определяет логику работы с чеком при его закрытии.

Для ККМ АТОЛ команда обязательна, так как переводит кассовый аппарат в соответствующий режим работы (формирование фискальной операции).

Команда ПРОДАЖА

XmlAPI

```
<ControlProtocol messageType="request">
<Sale Text="НАЗВАНИЕ_ТОВАРА" Amount="КОЛИЧЕСТВО_В_МДЕ"
Price="ЦЕНА_В_МДЕ" Group="ОТДЕЛ(СЕКЦИЯ)" Tax1="0..3"
Tax2="0..3" Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<Sale />
</ControlProtocol>
```

FileAPI

```
smde;НАЗВАНИЕ_ТОВАРА;ЦЕНА_В_МДЕ;КОЛИЧЕСТВО_В_МДЕ;НОМЕР
_НАЛОГА1;ОТДЕЛ(СЕКЦИЯ);
```

Производится добавление товара в чек.

ЦЕНА_В_МДЕ – цена товара в копейках. Целое число. 1234=12 руб 34 коп.

КОЛИЧЕСТВО_В_МДЕ – количество товара в тысячных долях (граммы, милилитры и т. п.). 1шт = 1000. 1кг=1000. 12345=12кг 345 гр. 96=0.096шт.

На чек может начисляться несколько налогов. Номер налога указывается в параметре Tax{1..4}.

При работе по FileAPI указывается только один номер налога (обычно этого достаточно).

ОТДЕЛ – отдел магазина, секция.

ВНИМАНИЕ! Тип операции (ПРОДАЖА или ВОЗВРАТ ПРОДАЖИ) определяется исходя из параметров команды ОТКРЫТЬ ЧЕК. По-умолчанию открывается чек продажи (в случае отсутствия команды ОТКРЫТЬ ЧЕК), однако, исходя из требований 54-ФЗ настоятельно рекомендуется подавать в шаблоне чека команду ОТКРЫТЬ ЧЕК

Команда ВОЗВРАТ ПРОДАЖИ

XmlAPI

```
<ControlProtocol messageType="request">
<ReturnSale Text="НАЗВАНИЕ_ТОВАРА"
Amount="КОЛИЧЕСТВО_В_МДЕ" Price="ЦЕНА_В_МДЕ"
Group="ОТДЕЛ(СЕКЦИЯ)" Tax1="0..3" Tax2="0..3"
Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<ReturnSale />
</ControlProtocol>
```

FileAPI

```
smde;НАЗВАНИЕ_ТОВАРА;ЦЕНА_В_МДЕ;КОЛИЧЕСТВО_В_МДЕ;НОМЕР
_НАЛОГА1;ОТДЕЛ(СЕКЦИЯ);
```

Производится добавление товара в чек.

ЦЕНА_В_МДЕ – цена товара в копейках. Целое число. 1234=12 руб 34 коп.

КОЛИЧЕСТВО_В_МДЕ – количество товара в тысячных долях (граммы, милилитры и т. п.). 1шт = 1000. 1кг=1000. 12345=12кг 345 гр. 96=0.096шт.

На чек может начисляться несколько налогов. Номер налога указывается в параметре Tax{1..4}.

При работе по FileAPI указывается только один номер налога (обычно этого достаточно).

ОТДЕЛ – отдел магазина, секция.

ВНИМАНИЕ! Тип операции (ПРОДАЖА или ВОЗВРАТ ПРОДАЖИ) определяется исходя из параметров команды ОТКРЫТЬ ЧЕК. По-умолчанию открывается чек продажи (в случае отсутствия команды ОТКРЫТЬ ЧЕК), однако, исходя из требований 54-ФЗ настоятельно рекомендуется подавать в шаблоне чека команду ОТКРЫТЬ ЧЕК

Команда ПОКУПКА

XmlAPI

```
<ControlProtocol messageType="request">
<Buy Text="НАЗВАНИЕ_ТОВАРА" Amount="КОЛИЧЕСТВО_В_МДЕ"
Price="ЦЕНА_В_МДЕ" Group="ОТДЕЛ(СЕКЦИЯ)" Tax1="0..3"
Tax2="0..3" Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<Buy />
</ControlProtocol>
```

FileAPI

```
smde;НАЗВАНИЕ_ТОВАРА;ЦЕНА_В_МДЕ;КОЛИЧЕСТВО_В_МДЕ;НОМЕР
_НАЛОГА1;ОТДЕЛ(СЕКЦИЯ);
```

Производится добавление товара в чек.

ЦЕНА_В_МДЕ – цена товара в копейках. Целое число. 1234=12 руб 34 коп.

КОЛИЧЕСТВО_В_МДЕ – количество товара в тысячных долях (граммы, милилитры и т. п.). 1шт = 1000. 1кг=1000. 12345=12кг 345 гр. 96=0.096шт.

На чек может начисляться несколько налогов. Номер налога указывается в параметре Tax{1..4}.

При работе по FileAPI указывается только один номер налога (обычно этого достаточно).

ОТДЕЛ – отдел магазина, секция.

ВНИМАНИЕ! Тип операции (ПРОДАЖА или ВОЗВРАТ ПРОДАЖИ) определяется исходя из параметров команды ОТКРЫТЬ ЧЕК. По-умолчанию открывается чек продажи (в случае отсутствия команды ОТКРЫТЬ ЧЕК), однако, исходя из требований 54-ФЗ настоятельно рекомендуется подавать в шаблоне чека команду ОТКРЫТЬ ЧЕК

Команда ВОЗВРАТ ПОКУПКИ

XmlAPI

```
<ControlProtocol messageType="request">
<ReturnBuy Text="НАЗВАНИЕ_ТОВАРА"
Amount="КОЛИЧЕСТВО_В_МДЕ" Price="ЦЕНА_В_МДЕ"
Group="ОТДЕЛ(СЕКЦИЯ)" Tax1="0..3" Tax2="0..3"
Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<ReturnBuy />
</ControlProtocol>
```

FileAPI

```
smde;НАЗВАНИЕ_ТОВАРА;ЦЕНА_В_МДЕ;КОЛИЧЕСТВО_В_МДЕ;НОМЕР
_НАЛОГА1;ОТДЕЛ(СЕКЦИЯ);
```

Производится добавление товара в чек.

ЦЕНА_В_МДЕ – цена товара в копейках. Целое число. 1234=12 руб 34 коп.

КОЛИЧЕСТВО_В_МДЕ – количество товара в тысячных долях (граммы, милилитры и т. п.). 1шт = 1000. 1кг=1000. 12345=12кг 345 гр. 96=0.096шт.

На чек может начисляться несколько налогов. Номер налога указывается в параметре Tax{1..4}.

При работе по FileAPI указывается только один номер налога (обычно этого достаточно).

ОТДЕЛ – отдел магазина, секция.

ВНИМАНИЕ! Тип операции (ПРОДАЖА или ВОЗВРАТ ПРОДАЖИ) определяется исходя из параметров команды ОТКРЫТЬ ЧЕК. По-умолчанию открывается чек продажи (в случае отсутствия команды ОТКРЫТЬ ЧЕК), однако, исходя из требований 54-ФЗ настоятельно рекомендуется подавать в шаблоне чека команду ОТКРЫТЬ ЧЕК

Команда СКИДКА НА ЧЕК (устарело в связи с ФЗ-54)

XmlAPI

```
<ControlProtocol messageType="request">
<DiscountCheck Text="НАЗВАНИЕ_ТОВАРА"
Summa="СУММА_В_МДЕ" Tax1="0..3" Tax2="0..3"
Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<DiscountCheck />
</ControlProtocol>
```

FileAPI

```
отсутствует
```

Скидка на чек в денежном эквиваленте.

ВНИМАНИЕ! Команда считается устаревшей, т. к. в соответствии с рекомендациями к 54-ФЗ все суммы в чеке должны указываться уже со всеми скидками / наценками.

Команда АННУЛИРОВАТЬ ЧЕК

XmlAPI

```
<ControlProtocol messageType="request">
<CancelCheck />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<DiscountCheck />
</ControlProtocol>
```

FileAPI

```
g;
```

До тех пор, пока чек не закрыт командой ЗАКРЫТЬ ЧЕК, его можно отменить. Т.е. добавленные позиции текущего чека отменяются. ККМ «забывает» про этот недооформленный чек.

Команда ПОДЫТОГ ЧЕКА

XmlAPI

```
<ControlProtocol messageType="request">
<SubTotal />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<SubTotal Money="СУММА_В_КОПЕЙКАХ" />
</ControlProtocol>
```

FileAPI

```
отсутствует
```

Получение из кассы суммы продаж в рамках текущего чека.

В FileAPI как правило не используется, т. к. управление ведётся из внешней товароучетной системы и сумма чека заранее известна.

Команда ЗАКРЫТЬ ЧЕК

XmlAPI

```
<ControlProtocol messageType="request">
<CloseCheck Text="СОПРОВОДИТЕЛЬНЫЙ_ТЕКСТ"
SummaCash="НАЛИЧНЫЕ_В_КОПЕЙКАХ"
Summa2="КАРТЫ_В_КОПЕЙКАХ" Summa3="3-
я_ФОРМА_ОПЛАТЫ_В_КОПЕЙКАХ" Summa4="4-
я_ФОРМА_ОПЛАТЫ_В_КОПЕЙКАХ"
Discount="ВЫРАВНИВАНИЕ_В_КОПЕЙКАХ" Tax1="0..3"
Tax2="0..3" Tax3="0..3" Tax4="0..3"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<CloseCheck Money="СДАЧА_В_КОПЕЙКАХ" />
</ControlProtocol>
```

FileAPI

Упрощенный вариант, одна форма оплаты

```
tmde;НОМЕР_ФОРМЫ_ОПЛАТЫ;СУММА_В_КОПЕЙКАХ;
```

Полная версия, 4 формы оплаты

```
tmde;СУММА_1;СУММА_2;СУММА_3;СУММА_4;НАЛОГ_1;НАЛОГ_2;Н
АЛОГ_3;НАЛОГ_4;
```

Закрытие чека указанными формами оплаты. Одновременно может использоваться до 4 форм оплаты за чек.

Если используется одна из безналичных форм оплаты, то сумма должна в точности совпадать с суммой чека, т. к. безналичная форма оплаты не подразумевает какой-либо сдачи.

Если данное условие не выполняется, то ККМ вернет ошибку о недостаточности средств для оплаты чека и чек закрыт не будет.

Закрытие (расчет) по чеку.

Чек может закрываться несколькими формами оплаты. SummaCash — для наличных. Все остальные — для карт, кредита, тарой и прочих форм расчета, наименования которых указываются в настройках ФР.

На чек может начисляться несколько налогов. Номер налога указывается в параметре Tax{1..4}.

ВЫРАВНИВАНИЕ_В_КОПЕЙКАХ – до ОнЛайн касс в этом параметре

находилось денежное значение скидки на чек. В новых ОнЛайн кассах этот параметр может принимать значение от 0 до 99 и служит для выравнивания суммы чека до целого числа рублей.

ВНИМАНИЕ! Тип операции (ПРОДАЖА или ВОЗВРАТ ПРОДАЖИ) определяется исходя из параметров команды ОТКРЫТЬ ЧЕК. По-умолчанию открывается чек продажи (в случае отсутствия команды ОТКРЫТЬ ЧЕК), однако, исходя из требований 54-ФЗ настоятельно рекомендуется подавать в шаблоне чека команду ОТКРЫТЬ ЧЕК

В версии 5.17.10.20 исправлена ошибка обработки налога в драйвере АТОЛ. Налог передается в первом параметре tax1. Остальные не обрабатываются, т. к. ККМ Атол могут применять только один налог на чек, в отличие от Штрих-М, где можно применить до 4-х разных налогов на чек.

Команда ПРОДОЛЖИТЬ ПЕЧАТЬ

XmlAPI

```
<ControlProtocol messageType="request">
<ContinuePrint />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<ContinuePrint />
</ControlProtocol>
```

FileAPI

```
continue_print;
```

Продолжить печать после обрыва ленты или отключения питания.

В кассовый драйвер встроена логика, которая отслеживает состояние ФР «после аварии» и автоматически шлет команду «ПРОДОЛЖИТЬ ПЕЧАТЬ». Как правило использовать данную команду не требуется, только в аварийной ситуации.

Команда ПЕЧАТЬ СТРОКИ ЧЕКА

XmlAPI

```
<ControlProtocol messageType="request">
<PrintTextLine Text="СТРОКА_ДЛЯ_ПЕЧАТИ"
DocumentType="docCheck|docControl|docList" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<PrintTextLine />
</ControlProtocol>
```

FileAPI

```
p;СТРОКА_ДЛЯ_ПЕЧАТИ;
```

Продолжить печать после обрыва ленты или отключения питания.

DocumentType определяет, на чем следует печатать

docCheck – чековая лента

docControl – контрольная лента

docList – подкладной документ

Команда ПЕЧАТЬ МНОЖЕСТВЕННЫХ СТРОК (СЛИП-чек)

XmlAPI

```
<ControlProtocol messageType="request">
<PrintTextLineLong Text="СТРОКА_ДЛЯ_ПЕЧАТИ"
DocumentType="docCheck|docControl|docList" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<PrintTextLineLong />
</ControlProtocol>
```

FileAPI

```
pm;СТРОКА_ДЛЯ_ПЕЧАТИ;
```

Печать строки текста шрифтом размера по-умолчанию.

DocumentType определяет, на чем следует печатать

docCheck – чековая лента

docControl – контрольная лента

docList – подкладной документ

В отличии от команды ПЕЧАТЬ СТРОКИ, эта команда предназначена для печати сразу нескольких строк, например, слип-чека или какого-то билета.

Разбиение длинной строки производится маркером #kkm_br#.

```
Пример.

pm;Будет#kkm_br#напечатано#kkm_br#четыре#kkm_br#строки
.;

Будет
напечатано
четыре
строки
```

Команда ПЕЧАТЬ КОПИИ ПОСЛЕДНЕГО ЧЕКА

XmlAPI

```
<ControlProtocol messageType="request">
<RepeatDocument />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<RepeatDocument />
</ControlProtocol>
```

FileAPI

```
c;
```

Происходит печать копии последнего документа из памяти кассы. На чеке ставится отметка «Повтор документа».

На большинстве моделей кассовых аппаратов печать чека при закрытой смене невозможна.

Если в открытой смене нет фискальных документов, то и повтора документа не будет.

Если в процессе печати использовались команды печати произвольного текста, то этой информации в дубликате чека не будет. Печатается только фискальная информация (как в кабинете ОФД).

Команда ОТКРЫТЬ ДЕНЕЖНЫЙ ЯЩИК

XmlAPI

```
<ControlProtocol messageType="request">
<OpenCashBox BoxId="НОМЕР_ЯЩИКА"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<OpenCashBox />
</ControlProtocol>
```

FileAPI

```
open_cash_box;НОМЕР_ЯЩИКА;
```

Если к кассе подключен денежный ящик, то на соответствующий разъем подается импульс и открывается замок ящика.

Команда ПЕЧАТЬ СТРОКИ ЗАДАННЫМ ШРИФТОМ

XmlAPI

```
<ControlProtocol messageType="request">
<PrintTextLineFont Text="СТРОКА_ДЛЯ_ПЕЧАТИ"
DocumentType="docCheck|docControl|docList"
FontId="НОМЕР_ШРИФТА_ККМ"/>
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<PrintTextLineFont />
</ControlProtocol>
```

FileAPI

```
print_font;НОМЕР_ШРИФТА_ККМ;СТРОКА_ДЛЯ_ПЕЧАТИ;
```

Печать строки текста указанным номером шрифта. Номер обычно от 0 до 15.

DocumentType определяет, на чем следует печатать

docCheck – чековая лента

docControl – контрольная лента

docList – подкладной документ

Команда ПЕЧАТЬ СТРОКИ ЧЕКА ЖИРНЫМ ШРИФТОМ

XmlAPI

```
<ControlProtocol messageType="request">
<PrintTextLineBold Text="СТРОКА_ДЛЯ_ПЕЧАТИ"
DocumentType="docCheck|docControl|docList" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<PrintTextLineBold />
</ControlProtocol>
```

FileAPI

```
print_bold;СТРОКА_ДЛЯ_ПЕЧАТИ;
```

Печать строки текста жирным / увеличенным шрифтом.

DocumentType определяет, на чем следует печатать

docCheck – чековая лента

docControl – контрольная лента

docList – подкладной документ

Команда ОТКРЫТЬ СМЕНУ

XmlAPI

```
<ControlProtocol messageType="request">
<OpenSession />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<OpenSession />
</ControlProtocol>
```

FileAPI

```
open_session;
```

Открывает новую смену. Печатается чек открытия смены.

Команда является ОБЯЗАТЕЛЬНОЙ для всех ОнЛайн касс, т. к. на основании этой команды в ОФД передаются данные о начале смены.

Смена в ККМ может быть открыта только один раз. Если смена уже открыта, а будет подана команда открытия смены, то будет возвращена ошибка с сообщением о том, что ККМ находится в неподходящем состоянии.

Команда ОТКРЫТЬ СМЕНУ В ФИСКАЛЬНОМ НАКОПИТЕЛЕ

XmlAPI

```
<ControlProtocol messageType="request">
<OpenSessionFN />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<OpenSessionFN />
</ControlProtocol>
```

FileAPI

```
open_session_fn;
```

Открывает новую смену в фискальном накопителе.

СЕРВИСНАЯ ФУНКЦИЯ! В обычной практике использовать не требуется.

Команда ПЕРЕДАТЬ TLV-структуру

XmlAPI

```
<ControlProtocol messageType="request">
<SetTLV type="ТИП_ПАРАМЕТРА" len="ДЛИНА_В_БАЙТАХ"
data="ДАННЫЕ" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<SetTLV />
</ControlProtocol>
```

FileAPI

```
set_tlv;ТИП_ПАРАМЕТРА;ДЛИНА_В_БАЙТАХ;ДАННЫЕ;
```

Передача дополнительных атрибутов чека в виде TLV-структуры. Все данные TLV кассовый аппарат передает в неизменном виде оператору фискальных данных (ОФД).

Полное описание TLV формата данных приведено в документе «Описание протокола уровня представления данных. Форматы фискальных документов.» ФНС РФ (версия 1.0 документа доступна по ссылке http://forum.rnditsoft.ru/download/file.php?id=11).

ВНИМАНИЕ! Все команды-инструкции TLV относятся к какому-то чеку. Вне чека они не имеют смысловой нагрузки. Наиболее корректно вызывать команду ПЕРЕДАТЬ TLV непосредственно перед командой ЗАКРЫТЬ ЧЕК.

Наиболее часто используется для указания ОФД type=1008 — передать в ОФД номер телефона или электронной почты. В этом случае команда будет иметь примерно следующий вид:

```
set_tlv;1008;22;test-mail@abcmail.test;
```

Команда СФОРМИРОВАТЬ ЧЕК КОРРЕКЦИИ (Верс. 1)

XmlAPI

```
<ControlProtocol messageType="request">
<CorrectionCheck
 type="ТИП_ПАРАМЕТРА" summa="СУММА_В_КОПЕЙКАХ"  />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<CorrectionCheck />
</ControlProtocol>
```

FileAPI

```
correction;ТИП_ПАРАМЕТРА;СУММА_В_КОПЕЙКАХ;
```

Сформировать чек коррекции. В ОФД передается скорректированная сумма движения денежных средств.

ТИП_ПАРАМЕТРА может принимать одно из значений

docIn – чек продажи

docRetIn – чек возврата продажи

docOut — чек покупки

docRetOut – чек возврата покупки

Команда ПОЛУЧИТЬ НОМЕР ПОСЛЕДНЕГО ЧЕКА В ФП

XmlAPI

```
<ControlProtocol messageType="request">
<GetLastFdId />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<GetLastFdId id="НОМЕР_ЧЕКА" />
</ControlProtocol>
```

FileAPI

```
отдельная реализация отсутствует. Смотрите ПОЛУЧИТЬ
ФИСКАЛЬНЫЙ ПРИЗНАК
```

Получить из фискальной памяти (ФП) номер последнего проведенного документа.

При использовании FileAPI команда объединена с командой ПОЛУЧИТЬ ФИСКАЛЬНЫЙ ПРИЗНАК при НОМЕР_ЧЕКА_ИЗ_ФП=0.

Команда ПОЛУЧИТЬ ФИСКАЛЬНЫЙ ПРИЗНАК

XmlAPI

```
<ControlProtocol messageType="request">
<GetFiskalMarkById  id="НОМЕР_ЧЕКА_ИЗ_ФП" />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
< GetFiskalMarkById
    fiskalDocId="НОМЕР_ЧЕКА_ИЗ_ФП"
    fiskalMark="ФИСКАЛЬНЫЙ ПРИЗНАК"
    date_time="дата-время печати чека"/>
</ControlProtocol>
```

FileAPI

```
get_fiscal_mark;НОМЕР_ЧЕКА_ИЗ_ФП;
```

Получить из фискальной памяти (ФП) номер последнего проведенного документа.

Команда имеет слегка разное поведение для ШТРИХ-М и АТОЛ

ШТРИХ-М — позволяет получить из памяти ККМ фискальный признак любого документа по номеру, указанному во входном параметре id. Единственное условие — чек должен присутствовать в памяти ККМ.

Если НОМЕР_ЧЕКА_ИЗ_ФП=0, то будет произведен поиск номера последнего документа и уже для него определен фискальный признак.

Будет возвращен номер чека и фискальный признак. Дату печати чека ККМ не возвращает.

АТОЛ — можно получить фискальный признак и номер только для последнего документа. Поэтому указывать НОМЕР_ЧЕКА_ИЗ_ФП лучше всегда равный нулю.

Будет возвращен номер чека и фискальный признак и дата печати чека последнего чека. Формат даты-времени ГГГГ-ММ-ДД ЧЧ:ММ:СС.

Команда ПОЛУЧИТЬ НОМЕР ЧЕКА В СМЕНЕ

с версии 5.17.10.20

XmlAPI

```
<ControlProtocol messageType="request">
<GetNumSaleCheck />
</ControlProtocol>

<ControlProtocol messageType="answer">
<error text="Ошибок нет" id="0"/>
<GetNumSaleCheck session="0" value="" />
</ControlProtocol>
```

FileAPI

```
get_num_sale_check;
get_num_returnsale_check;
get_num_buy_check;
get_num_returnbuy_check;
```

Команда позволяет получить номер фискального документа value в рамках смены session.

Если в смене ещё не было чеков, то вернётся 0.

Команда поддерживает 4 типа чеков

Чек продажи — xml-название тега GetNumSaleCheck fileAPI название команды get_num_sale_check

Чек возврата продажи — xml-название тега GetNumReturnSaleCheck fileAPI название команды get_num_returnsale_check

Чек покупки — xml-название тега GetNumBuyCheck fileAPI название команды get_num_buy_check

Чек возврата покупки — xml-название тега GetNumReturnBuyCheck fileAPI название команды get_num_returnbuy_check

В ПО СуперВизор добавлено отображение номера чека и смены

13:31:57 19.10.2017	cmd.txt	**GetNumReturnBuyCheck**	**(0) -- Ошибок нет** Чек возврата покупки: смена: 805 чек:0
13:31:57 19.10.2017	cmd.txt	**GetNumBuyCheck**	**(0) -- Ошибок нет** Чек покупки: смена: 805 чек:0
13:31:57 19.10.2017	cmd.txt	**GetNumReturnSaleCheck**	**(0) -- Ошибок нет** Чек возврата продажи: смена: 805 чек:0
13:31:57 19.10.2017	cmd.txt	**GetNumSaleCheck**	**(0) -- Ошибок нет** Чек продажи: смена: 805 чек: 0

Команда идентично работает на ККМ АТОЛ и Штрих-М.

Общие комментарии

Проект «Кассовый сервер QKkmServer» находится в постоянном развитии. Особенно в последнее время с введением 54-ФЗ об ОнЛайн кассах.

Команда проекта будет признательна, если Вы будете сообщать об ошибках и неточностях в работе ПО в техподдержку.

Так же принимаются пожелания по внедрению нового функционала.

Контактные данные:

Сайт www.rnditsoft.ru

Почта rndit@mail.ru

skype: knyazev_em

telegram: +79282701319

tel: +79282701319

Руководитель проекта: Князев Евгений Михайлович