POS SDK

POS-Интеллект 5.4 (русский)

Обновлено 05/07/2024

Table of Contents

1 POS SDK. Введение 5
1.1 Назначение документа 5
1.2 Структура документа 5
1.3 Общие сведения о POS SDK 5
2 XML-протокол 7
2.1 Описание протокола XML 7
2.1.1 Общие сведения о протоколе XML 7
2.1.2 Контрольный пакет 8
2.1.3 Формат файла настроек xml_titles.txt 8
2.1.4 Взаимодействие с кассовым сервером 12
2.2 Настройка системы POS Интеллект при работе с XML протоколом передачи данных 13
2.3 Конфигурирование файла настроек xml_titles.txt 14
2.4 Описания пакетов для XML протокола взаимодействия с системой POS-Интеллект 19
2.4.1 Поля и их значения 19
2.4.2 Формат пакетов обмена данными с системой видеоконтроля ITV 21
3 Использование парсера с расширением .prl 29
3.1 Общие сведения о создании парсеров с расширением .prl в ПК POS-Интеллект 29
3.2 Регулярные выражения 29
3.2.1 Общие сведения о регулярных выражениях 29
3.2.2 Квантификаторы 30
3.2.3 Символьные классы 30
3.2.4 Спецсимволы 31
3.2.5 Условия выбора 32
3.3 Описание интерфейса утилиты parser_designer.exe 32
3.4 Создание парсера 34
3.5 Проверка созданного шаблона 35
3.6 Настройка парсера в ПК POS Интеллект 37
4 Библиотеки для работы с ПК POS Интеллект 41
4.1 COM. Библиотека для работы через TCP/IP соединение 41

4.2 ActiveX компонент для работы через TCP/IP соединение 42
4.3 DLL. Библиотека для работы через TCP/IP соединение 42
5 Взаимодействие с 1С 44
6 Приложение 1. Утилита tcpgen.exe для отправки тестовых данных в POS Интеллект 45
6.1 Общие сведения об утилите tcpgen.exe 45
6.2 Работа с утилитой tcpgen.exe 45

Скачать POS SDK

1 POS SDK. Введение

На странице:

- Назначение документа
- Структура документа
- Общие сведения о POS SDK

1.1 Назначение документа

Документ *Набор утилит и библиотек POS SDK* является собственностью компании ITV. Данный документ является справочно-информационным пособием и предназначен для производителей кассового ПО.

Документ предоставляет сведения о наборе утилит и библиотек POS SDK, позволяющем настроить работу *POS-Интеллект* с кассовыми терминалами различных производителей.

1.2 Структура документа

В данном документе представлены следующие материалы:

1. Описание протокола XML;
2. Сведения о работе с использованием парсеров;
3. Описание внешних библиотек для работы с ПК *POS-Интеллект*.

1.3 Общие сведения о POS SDK

Набор утилит и библиотек POS SDK можно скачать на странице.

Набор утилит в POS SDK позволяет создавать парсеры для извлечения информации из чеков, получаемых от кассовых аппаратов, для внесения ее в базу данных чеков.

В зависимости от протокола передачи чековых данных от POS-терминала на POS-сервер база данных заполняется одним из следующих способов:

1. С использованием парсера xml_titles.txt.
2. С использованием парсера с расширением .prl.

> **Примечание.**
>
> Подробная информация о настройке парсеров в ПК *POS-Интеллект* приведена в документе Программный комплекс POS-Интеллект. Руководство Администратора.

Набор библиотек в POS SDK предназначен для отправки данных в *POS-Интеллект* из сторонних программ.

> **Внимание!**

Библиотеки, входящие в состав POS SDK, приложены в качестве примеров. Компания ITV не несет ответственности за их работу.

2 XML-протокол

2.1 Описание протокола XML

2.1.1 Общие сведения о протоколе XML

Данный протокол позволяет производителям кассового ПО добавить возможность сопряжения с системой контроля кассовых операций *POS-Интеллект*, если данные о кассовых операциях отсылаются в виде XML пакетов. Данный формат позволяет передавать любое количество значимых параметров в систему и удобным образом организовать их отображение на видеоизображении и сохранение в базе транзакций.

XML пакет имеет основной тэг **<TransactionBlock>**. На каждую кассовую операцию в *Интеллект* отсылается отдельный xml пакет. Каждый такой пакет должен иметь начальный тег **<TransactionBlock>** и конечный **</TransactionBlock>**. Таким образом, каждая операция на кассовом узле имеет свое представление в виде XML пакета. Пример отосланного пакета показан на рисунке.

```
<TransactionBlock>
    <FunctionNumber>1</FunctionNumber>
    <FunctionName>Регистрация пользователя</FunctionName>
    <UserNumber>111</UserNumber>
    <UserName>Иванов</UserName>
    <TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

Все параметры представляются в виде текста. Для чисел можно использовать точку или запятую для отделения дробной части. Преобразования к нужному типу происходят автоматически.

Кассовая программа должна отсылать эти данные на любой TCP, UDP или RS232 порт (указывается в настройках). Для каждой кассы используется отдельный порт.

В случае использования TCP протокола необходимо предусмотреть автоматическое восстановление связи в случае разрыва.

Обязательные тэги xml-пакета:

1. **FunctionNumber** – номер функции.
2. **TransactionTimestamp** (условно обязательный тег) – время события, заданное в формате **yyyy-mm-ddThh:nn:ss.fff**. Указывается в часовом поясе UTC+0. Если тэг **TransactionTimestamp** отсутствует в пакете, в базу данных записывается не время события, а время прихода пакета. Использованы следующие обозначения:
   yyyy – год в виде четырехзначного числа.
   mm – месяц, в диапазоне от 01 до 12.
   dd – день месяца, в диапазоне от 01 до 31.
   hh – час в 24-часовом формате от 00 до 23.
   nn – минуты, в диапазоне от 00 до 59.
   ss – секунды, в диапазоне от 00 до 59.
   fff – тысячные доли секунды (миллисекунды) в значении даты и времени.

Элементы XML пакета необходимо обязательно отделять разделителями строк (следующие подряд символы 0xD, 0xA). Это удобнее делать на этапе отладки системы (при использовании утилиты *xml_test.exe,* см. Конфигурирование файла настроек xml_titles.txt).

Общее число параметров может быть любым. В названии параметров не должно быть пробелов. Приходящие данные отображаются на экране в виде титров и записываются в базу для последующего анализа.

Для настроек служит файл **Intellect\Modules\Pos\xml_titles.txt** (см. раздел Формат файла настроек xml_titles.txt).

Все полученные данные записываются в таблицы **POS_LOG_MASTER** и **POS_LOG_DETAIL**.

По умолчанию база данных чеков называется pos, используется СУБД MS SQL Server 2008.

2.1.2 Контрольный пакет

В ситуации, когда активности на кассе нет, но связь с *POS-Интеллект* установлена, необходимо периодически отсылать контрольный пакет, подтверждающий наличие связи по TCP/IP. Рекомендуемый интервал - 30 секунд. Контрольный пакет отсылается с параметром **FunctionNumber = 77777**.

```
<TransactionBlock>
<FunctionNumber>77777</FunctionNumber>
<DateTime>2014-04-16T17:37:18.516</DateTime>
</TransactionBlock>
```

На стороне сервера *POS* задано время в секундах, в течение которого ожидается считывание данных. Если в течение данного времени не приходит ни один XML пакет, то система считает, что связь с ПО потеряна. В таком случае соединение закрывается и устанавливается вновь.

2.1.3 Формат файла настроек xml_titles.txt

На рисунке показан примерный вид файла xml_titles.txt.

```
[2]
Новый чек  №<CheckNumber>,
Кассир : <UserName> , № <UserNumber>

[3]
        <ItemName>(<ItemNumber>)    x  <ItemQuantity>   <ItemPrice> РУБ

[9]
Печать чека №<CheckNumber> ,<TransactionDate> <TransactionTime>

[TRANSFORM]
UserName=cashier_name
UserNumber=cashier_number
ItemName=item_name
ItemQuality=item_count
ItemPrice=item_total
5:ItemPrice=item_amount
6:ItemPrice=item_price
ItemNumber=item_code
CheckNumber=check_number

[PROLOG]
2
1
[BODY]
3
8
[EPILOG]
9

[FUNCTIONNAME]
2=Открытие чека
3=Продажа товара
9=Печать чека
```

Числа в квадратных скобках представляют собой номера функций, приходящих в поле **FunctionNumber** пакета. Непосредственно под числом в скобках указывается шаблон – описание того, как указанная функция будет отображаться поверх видеоизображения. Переменные, стоящие в угловых скобках "**<>**", будут заменяться на соответствующие им значения. Если в строке находится хотя бы один такой параметр и от POS-терминала придут данные, в которых данный параметр отсутствует, то вся строка будет игнорироваться и не будет отображена на экране.

Можно указывать как одиночные функции (например **[1]** или **[2]**), так и совокупность нескольких функций, если у них одинаковый шаблон (например **[1,2,5,10]**).

Настройка секций и полей файла происходит следующим образом:

1. В секции **[PARAM_TEXT_FORMAT]** задаются параметры форматирования полей.

   ```
   [PARAM_TEXT_FORMAT]
   ItemName=25,left,upper
   ```

   Приведен пример, в котором везде, где отображается параметр **ItemName**, его максимальная длина будет 25 символов (если длина меньше, то автоматически добавятся пробелы), выравнивание будет по левому краю, все символы будут отображаться большими буквами.
   Возможные значения выравнивания**: left,right,center**.
   Возможные значения размера букв: **upper,lower**.
2. В секции **[PROLOG]** указываются номера функций, которые соответствуют началу нового чека. Когда приходит xml-пакет, функция, номер которой указан в этом поле, автоматически увеличивает (на единицу) внутренний счетчик чеков (транзакций). Параметры, соответствующие этим функциям должны являться названиями столбцов таблицы **POS_LOG_MASTER**.

a. В секции **[BODY]** указываются номера функций, которые соответствуют телу чека. Параметры, соответствующие этим функциям, должны являться названиями столбцов таблицы **POS_LOG_DETAIL**.
b. В секции **[EPILOG]** указываются номера функций, которые соответствуют концу чека. Параметры, соответствующие этим функциям должны являться названиями столбцов таблицы **POS_LOG_MASTER**.
c. В секции **[HIDE_TITLES]** указываются номера функций, которые не должны отображаться на экране в виде текста. Тем не менее, данные от этих функций сохраняются в базу POS – если они указаны в секциях **[PROLOG],[BODY],[EPILOG]** (см. **PROLOG,BODY,EPILOG**).
d. В секции **[PROLOG_EVENT_PARAM]** указываются параметры и строка для вывода. Если в пришедшем xml-пакете присутствует данный параметр, то выводится указанная строка. Данная секция применяется для систем, где нет конкретного определения начала чека.

```
[PROLOG_EVENT_PARAM]
check_number,НОВЫЙ ЧЕК!
```

3. В секции **[TRANSFORM]** указываются соответствия присылаемых параметров и названий столбцов таблиц базы данных **pos** в зависимости от номера функции. Данная секция необходима для того, чтобы не модифицировать названия столбцов базы (для совместимости).

```
[TRANSFORM]
UserName=cashier_name
UserNumber=cashier_number
ItemName=item_name
ItemQuality=item_count
ItemPrice=item_total
5:ItemPrice=item_amount
6:ItemPrice=item_price
ItemNumber=item_code
CheckNumber=check_number
```

В приведенном примере пришедшему параметру **ItemPrice** в случае, если номер функции не равен 5 или 6, будет соответствовать столбец **item_total**, в случае, если номер функции равен 5 – столбец **item_amount**, если номер функции равен 6 – столбец **item_price**.
Можно послать вместо строки вида:

```
<ItemPrice>12,34</ItemPrice>
```

строку вида:

```
<item_total>12,34</item_total>
```

В этом случае, описание в поле **[TRANSFORM]** не требуется.
4. В секции **[FUNCTIONNAME]** указываются соответствия получаемых номеров функций (**FunctionNumber**) их названиям. Используется для связывания номеров функций и их названий в отчетах, формируемых по результатам поисковых запросов оператора.
5. В секции **[IGNORE_ZERO]** указывается название параметра, при равенстве которого нулю требуется игнорировать соответствующее событие от кассы. Например, если от кассы приходит пакет, а в файле xml_titles.txt указан текст, представленный на рисунке, то блок с указанным событием не будет записан в таблицу POS_LOG_DETAIL.

```
<TransactionBlock>
        <FunctionNumber>12</FunctionNumber>
        <TerminalId>001</TerminalId>
        <TransactionDate>23.01.2008</TransactionDate>
        <TransactionTime>14:15:30.992</TransactionTime>
        <CashierID>005</CashierID>
        <CashierName>Сячина Анна Н.      </CashierName>
        <CheckNumber>0063</CheckNumber>
        <Article>   4902210219202</Article>
        <ProductName>СИГАРЕТЫ С/Ф LUCIA L</ProductName>
        <ProductQty>1</ProductQty>
        <ProductPrice>47.00</ProductPrice>
        <RowSumm>47.00</RowSumm>
        <IDCType>S</IDCType>
        <IDCCode>0</IDCCode>
        <IDCNumber>   4902210219202</IDCNumber>
        <IDCCount>1</IDCCount>
        <IDCAmount>0</IDCAmount>
</TransactionBlock>
```

Пакет с нулевым параметром IDCAmount

```
[IGNORE_ZERO]
IDCAmount

[BODY]
12
```

Фрагмент файла xml_titles.txt, указывающий на необходимость игнорирования пакета
На основе функций указанных в разделах **PROLOG,BODY,EPILOG** система определяет каким образом добавлять в базу значения параметров*. Параметры записываются в базу тогда и только тогда, когда номера функций, в которых передаются указанные параметры, добавлены в одну из трех секций (***PROLOG,BODY,EPILOG***).*

Значения в базу заносятся по следующему алгоритму:

1. При получении функции, прописанной в секции **[PROLOG]**, осуществляется добавление записи в таблицу **[POS_LOG_MASTER]** при помощи SQL операции **INSERT**. Программа автоматически увеличивает счетчик **[check_id]** и запоминает его.
2. При получении функции, прописанной в секции **[BODY]**, осуществляется добавление записи в таблицу **[POS_LOG_DETAIL]** при помощи SQL операции **INSERT**, где в поле **[id]** пишется ранее запомненное значение **[check_id]** таблицы **[POS_LOG_MASTER]**.
3. При получении функции, прописанной в секции **[EPILOG]**, осуществляется обновление записи в таблице **[POS_LOG_MASTER]** при помощи SQL операции **UPDATE,** для строки, где **[check_id]** соответствует ранее запомненному значению.
   Таким образом, поле **[check_id]** используется внутри программы, и не должно использоваться в качестве имени одного из передаваемых параметров.
   Порядок следования секций может быть произвольным. Допускается написание однострочных комментариев при помощи символов //.
   Пусть приходит пакет при добавлении товара в чек:

```
<TransactionBlock>
        <FunctionNumber>1</FunctionNumber>
<FunctionName>Добавление товара</FunctionName>
<ItemName>Сумка</ItemName>
<ItemCount>12<ItemCount>
<ItemPrice>12,34</ItemPrice>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

и этому пакету соответствует описание:

```
[1]
<FunctionName> :
Товар: <ItemName>
Кол-во: <ItemCount>
Цена: <ItemPrice>

[BODY]
1

[TRANSFORM]
ItemName=item_name
ItemCount=item_count
ItemPrice=item_price
```

В этом случае на экран будут выведены титры:

```
Добавление товара:
Товар: Сумка
Кол-во: 12
Цена: 12,34
```

В базу **pos** в таблицу **POS_LOG_DETAIL** будет добавлена запись, содержащая значения полей **item_name, item_count, item_total,** а так же значение ***FunctionNumber***.

2.1.4 Взаимодействие с кассовым сервером

Возможен случай, когда происходит интеграция с кассовым сервером, который может принимать события от всех касс (или заправочных колонок АЗС) и пересылать их в систему видеонаблюдения, используя одно логическое соединение с помощью плагина **SplitterPlugin**.

При этом к каждой кассе (заправочной колонке АЗС) должна быть привязана отдельная камера (камеры).

В этом случае в пакете обязательно должен присутствовать тэг **StationNumber** или **TerminalId**.

```
<StationNumber>ID</StationNumber>
```

```
<TerminalId>ID</TerminalId>
```

Здесь ID – уникальный идентификатор кассы (заправочной колонки).

Пример взаимодействия с кассовым сервером:

На заправке установлена одна касса, и данные в систему видеонаблюдения отправляются через нее. К кассе подключено 3 заправочных колонки, на каждую заправочную колонку смотрит отдельная камера.

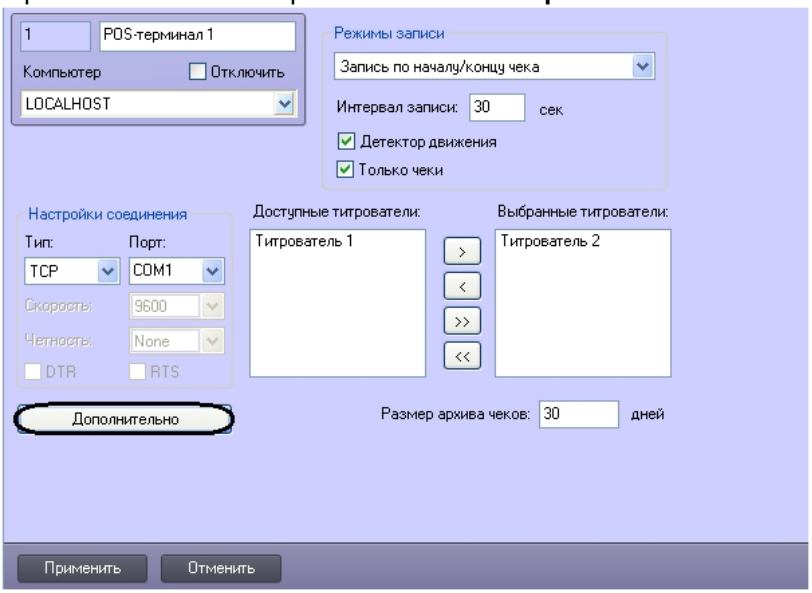
Для реализации видеомониторинга в составе системы POS-Интеллект нужно создать в дереве объектов системы 3 объекта **Pos-терминал**, к каждому из которых привязать камеру, и объект **Pos-терминал**, соответствующий кассе..

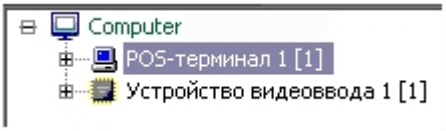
Далее касса будет отсылать данные входящей в состав системы POS-Интеллект утилите mixforward.exe , которая будет анализировать данные и перенаправлять их на созданные объекты системы.

2.2 Настройка системы POS Интеллект при работе с XML протоколом передачи данных

Настройка системы *POS-Интеллект* при работе с XML протоколом передачи данных проходит следующим образом:

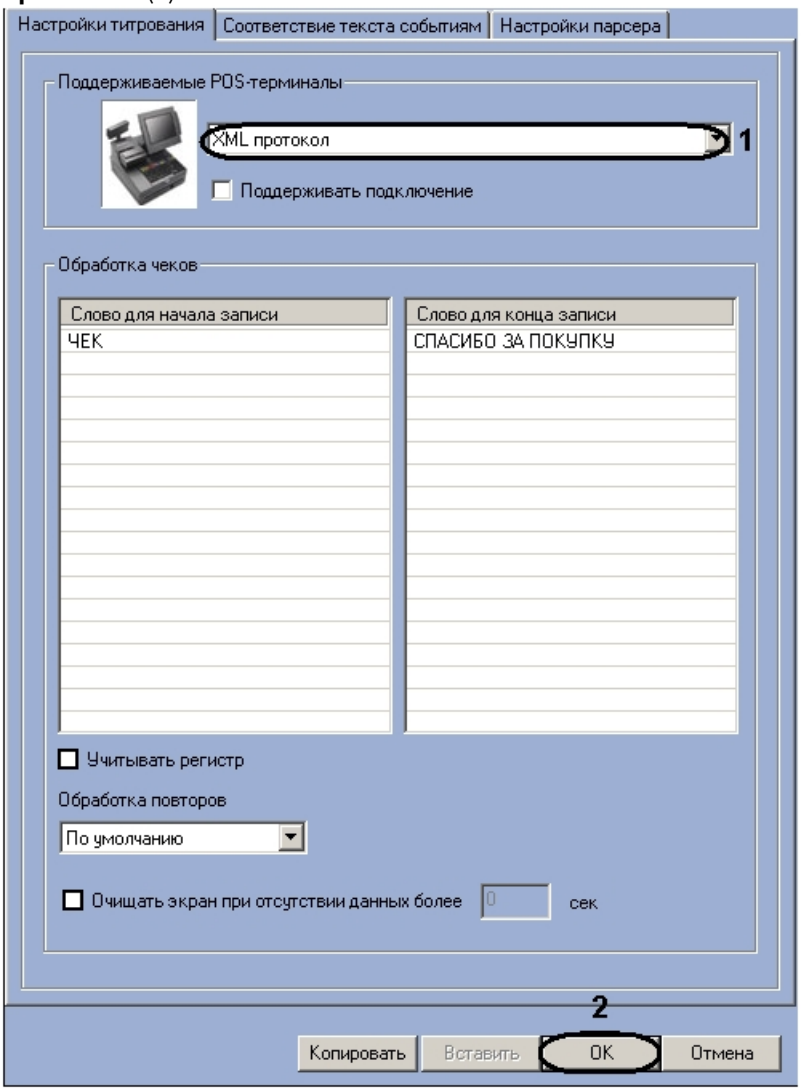
1. Перейти на панель настроек объекта **POS-терминал**.



> **Примечание.**
>
> Данный объект создается на базе объекта **Компьютер** на вкладке **Оборудование** диалогового окна **Настройка** системы.
>
> 
>

2. Нажать кнопку **Дополнительно**.

3. В открывшемся окне выбрать **XML протокол** из раскрывающего списка **Поддерживаемые POS-терминалы** (**1**).



4. Нажать кнопку **OK** (**2**).

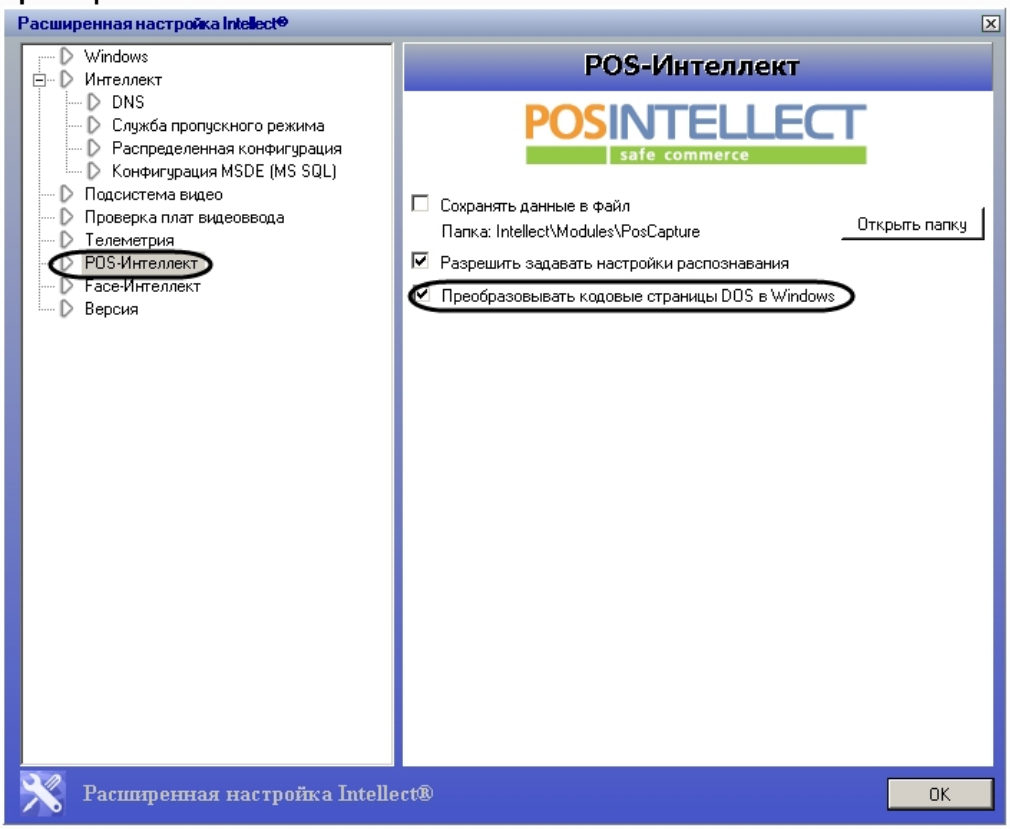
Настройка системы *POS-Интеллект* завершена.

2.3 Конфигурирование файла настроек xml_titles.txt

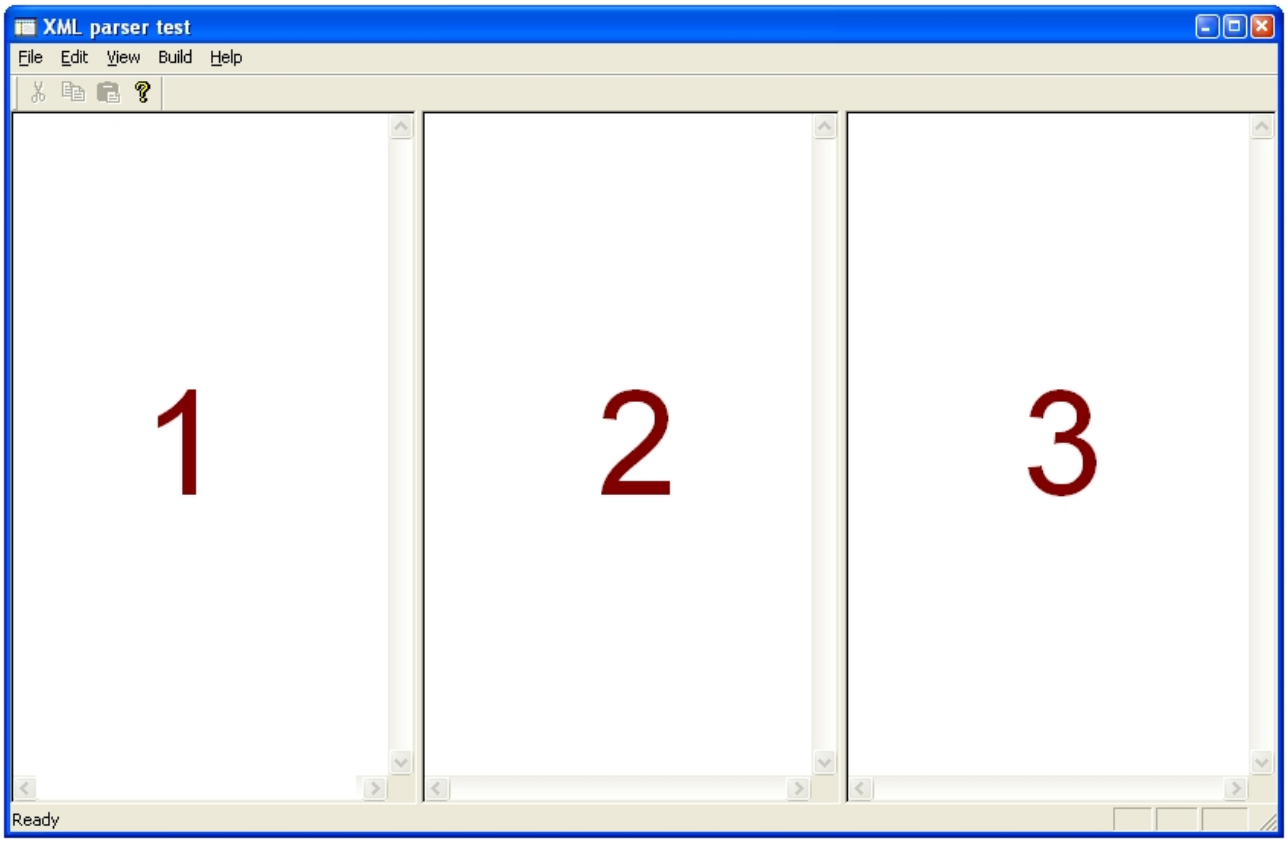
Для упрощения настройки файла *xml_titles.txt* следует использовать специализированную утилиту. Утилита *xml_test.exe* предназначена для отладки отображения титров поверх видеоизображения в случае работы с XML протоколом модуля POS-терминал. Утилита *xml_test.exe* входит в набор утилит и библиотек POS SDK. Для запуска утилиты необходимо скачать архив POS SDK, распаковать его и запустить исполняемый файл POS_SDK\XML_Protocol\Xml_test.exe.

Примечание.

Если отсылаемые данные в кодировке DOS, то необходимо запустить утилиту расширенной настройки ПК Интеллект tweaki.exe и установить флажок **Преобразовывать кодовые страницы DOS в Windows.**

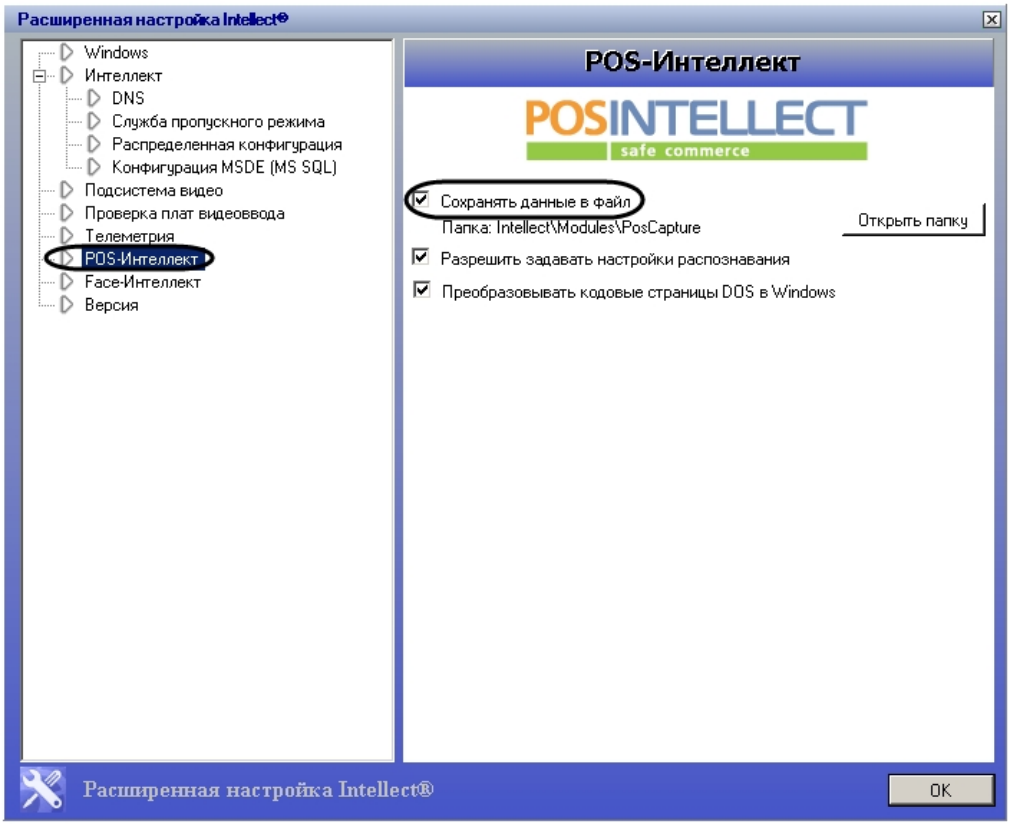


Утилита *xml_test.exe* состоит их 3 окон:



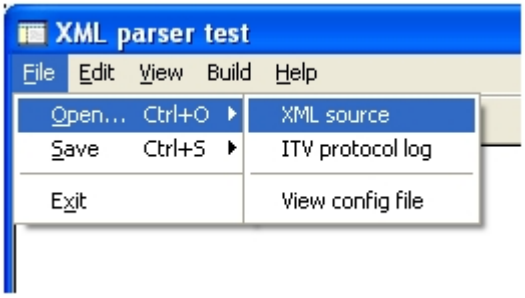
В первое окно загружается XML-лог, полученный от кассы. Логи хранятся в папке <Директория установки ПК *Интеллект*>\Modules\PosCapture. Файл лога называется pos_N.log, где N – номер POS-терминала, соответствующего кассе, с которой поступил лог.

Если в данной папке отсутствуют логи, необходимо убедиться, что в утилите tweaki.exe в разделе **POS-Интеллект** установлен флажок **Сохранять данные в файл**.

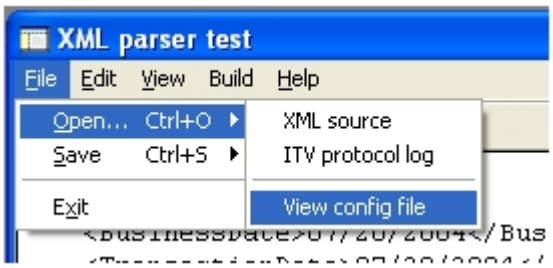


Для открытия лога необходимо перейти в меню **File**, выбрать пункт **Open** и далее выбрать пункт **XML source**).

> **Примечание.**
>
> Пункт **ITV protocol log** не используется.

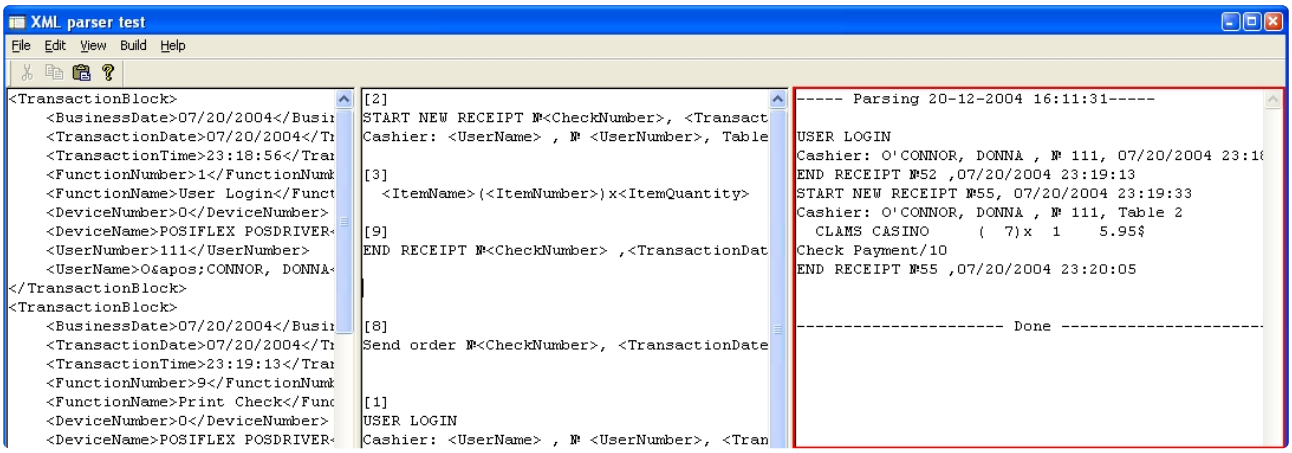


Во второе окно загружается файл парсера xml_titles.txt, который отвечает за отображение. Для открытия файла xml_titles.txt необходимо перейти в меню **File**, выбрать пункт **Open** и далее выбрать **View config file**.



Можно загрузить заранее сохраненный файл или сформировать его самостоятельно.

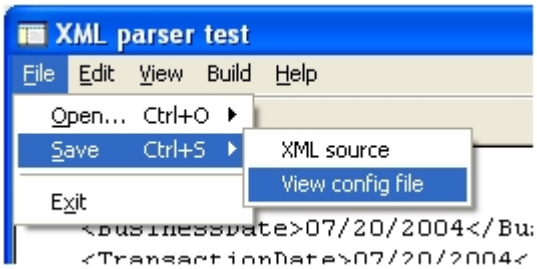
При нажатии на кнопку F7 в третьем окне отобразятся титры аналогично наложению титров поверх видеоизображения при реальной работе системы *POS-Интеллект* вместе с подключенным оборудованием.



Таким образом, основная работа сводится к редактированию содержимого второго окна.

В нём можно описывать, как должны выглядеть титры при отображении и как соотносится функция с содержанием чека.

После окончания работы, необходимо сохранить файл xml_titles.txt. Для этого необходимо перейти в меню **File**, выбрать пункт **Save** и далее **View config file**:



После это необходимо скопировать файл в **<Директория установки ПК Интеллект>\Modules\Pos\xml_titles.txt**.

2.4 Описания пакетов для XML протокола взаимодействия с системой POS-Интеллект

2.4.1 Поля и их значения

В таблице приведен список полей и их значений. Данный список является рекомендованным. Функции могут быть добавлены или удалены.

Параметры функций, их названия и количество могут быть произвольно изменены, кроме параметра **FunctionNumber** (это обязательный параметр).

Изменение функций и параметров потребует настройки файла xml_titles.txt.

Поле	**Значение**
FunctionNumber	Номер функции
FunctionName	Наименование функции
cash_number	Номер кассы
card_number	Номер карты
date	Дата формирования документа
discount_name	Название скидки
enter_type	Способ ввода номера платёжной карты: 1 – ввод с клавиатуры; 2 – через магнитный считыватель; 3 – через pin-pad; 4 – по штрих-коду через сканер
receipt_number	Номер чека
item_id	Идентификатор товара
item_name	Название товара
item_barcode	Штрихкод товара
item_price	Цена товара
item_quantity	Количество товара

item_discount	Скидка по товарной позиции
item_amount	Сумма товарной позиции в чеке
receipt_amount	Текущая сумма чека
item_pos	Номер товарной позиции
receipt_discount	Сумма скидки
receipt_discount_number	Значение скидки (%)
payment_name	Название платежного средства
payment_amount	Сумма чека
payment_amount_with_change	Сумма, принятая от покупателя
payment_change	Сумма сдачи
result	Результат завершения чека: 1– аннулированный чек; 2 – отложенный чек
result_str	Текст, поясняющий результат завершения чека
shift_number	Номер смены
TransactionTimestamp	Время события, заданное в формате **yyyy-mm-ddThh:nn:ss.fff**. Параметр необходимо указывать в часовом поясе UTC+0
receipt_type	Тип чека
restore	Тип восстановления чека
type_str	Поясняющий текст
type	Тип операции
user_name	Имя кассира

user_id	Идентификатор кассира

Примечание.

В операциях, связанных с чеками (7, 8 , 11, 19), параметр <**type>** может принимать следующие значения (если таковые возможны в реализации):
0 – чек на продажу;
1 – возврат;
2 – возврат по чеку;
5 – восстановление отложенного чека;
7 – чек на инвентаризацию;
8 – нефискальный чек;
9 – возврат по нефискальному чеку.

В операции 15 значения параметра <type>:
100 – внесение;
101 – изъятие.
Дата и время должны иметь формат: YYYY-MM-DD HH:MM:SS.MS, где YYYY – год, MM – месяц, DD – день, HH – час, MM – минуты, SS – секунды, MS – миллисекунды. Например, 2006-09-01 13:13:34.045.

2.4.2 Формат пакетов обмена данными с системой видеоконтроля ITV

Примечание

Для работы с данным форматом пакетов обмена данными можно воспользоваться следующим файлом настроек xml_titles.txt.

1. **Вход кассира**

```
<TransactionBlock>
                <FunctionNumber>1</FunctionNumber>
<FunctionName>Вход кассира</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

2. **Выход кассира**

```
<TransactionBlock>
                <FunctionNumber>2</FunctionNumber>
<FunctionName>Выход кассира</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
```

```
</TransactionBlock>
```

3. **Добавление позиции**

```
<TransactionBlock>
                <FunctionNumber>3</FunctionNumber>
<FunctionName>Добавление позиции</FunctionName>
                <receipt_number>10</receipt_number>
                <item_id>006946</item_id>
                <item_name>ДИРОЛ ДЕТСКИЙ МЭЙДЖИК ТАТУ</item_name>
                <item_barcode></item_barcode>
                <item_price>10.00</item_price>
                <item_quantity>1.000</item_quantity>
                <item_amount>10.00</item_amount>
                <receipt_amount>0.00</receipt_amount>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

4. **Смена количества**

```
<TransactionBlock>
                <FunctionNumber>4</FunctionNumber>
<FunctionName>Смена количества</FunctionName>
                <receipt_number>10</receipt_number>
                <item_id>013611</item_id>
                <item_name>ЖЕВ. ДРАЖЕ МЕНТОС ФРУКТЫ  38 ГР</item_name>
                <item_barcode></item_barcode>
                <item_price>15.10</item_price>
                <item_quantity>5.000</item_quantity>
                <item_amount>75.50</item_amount>
                <receipt_amount>29.00</receipt_amount>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

5. **Аннулирование или сторнирование позиции**

```
<TransactionBlock>
                <FunctionNumber>5</FunctionNumber>
<FunctionName>Аннулирование или сторнирование позиции</FunctionName>
                <receipt_number>10</receipt_number>
                <item_id>013611</item_id>
                <item_name>ЖЕВ. ДРАЖЕ МЕНТОС ФРУКТЫ  38 ГР</item_name>
                <item_barcode></item_barcode>
<item_price>15.10</item_price>
                <item_quantity>5.000</item_quantity>
                <item_amount>75.50</item_amount>
                <receipt_amount>29.00</receipt_amount>
                <date>27.09.2006 15:42:19</date>
```

```
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

6. **Изменение цены текущей позиции**

```
<TransactionBlock>
                <FunctionNumber>6</FunctionNumber>
<FunctionName>Изменение цены текущей позиции</FunctionName>
                <receipt_number>10</receipt_number>
                <item_id>005393</item_id>
                <item_name>ГАЗ. НАПИТОК БАРН С КОФЕИНОМ 0,25 Л</item_name>
                <item_barcode></item_barcode>
                <item_price>29.00</item_price>
                <item_quantity>1.000</item_quantity>
                <item_amount>29.00</item_amount>
                <receipt_amount>0.00</receipt_amount>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

7. **Отложенный или аннулированный чек**

```
<TransactionBlock>
                <FunctionNumber>7</FunctionNumber>
<FunctionName>Отложенный или аннулированный чек</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
                <receipt_number>283</receipt_number>
                < receipt_type>0</ receipt_type>
                <result>1</result>
                <result_str>АННУЛИРОВАНИЕ ЧЕКА</result_str>
                <receipt_amount>240.00</receipt_amount>
                <receipt_discount>0.00</receipt_discount>
<receipt_discount_number>0</receipt_discount_number>
                <cash_number>9999999</cash_number>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

8. **Завершение чека**

```
<TransactionBlock>
                <FunctionNumber>8</FunctionNumber>
<FunctionName>Завершение чека</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
                <receipt_number>280</receipt_number>
                <receipt_type>0</receipt_type>
                <result>0</result>
```

```
                <receipt_discount>0.00</receipt_discount>
<receipt_discount_number>0</receipt_discount_number>
                <receipt_amount>721.90</receipt_amount>
                <cash_number>9999999</cash_number>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

9. **Подытог**

```
<TransactionBlock>
                <FunctionNumber>9</FunctionNumber>
<FunctionName>Подытог</FunctionName>
                <receipt_number>10</receipt_number>
                <receipt_amount>29.00</receipt_amount>
                <receipt_discount>0.00</receipt_discount>
<receipt_discount_number>0</receipt_discount_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

10. **Платёж по чеку**

```
<TransactionBlock>
                <FunctionNumber>10</FunctionNumber>
<FunctionName>Платёж по чеку</FunctionName>
                <receipt_number>10</receipt_number>
                <payment_name>Наличные</payment_name>
                <payment_amount>721.90</payment_amount>
                <payment_amount_with_change>800.00</payment_amount_with_change>
                <payment_change>78.10</payment_change>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

11. **Открытие чека**

```
<TransactionBlock>
                <FunctionNumber>11</FunctionNumber>
<FunctionName>Открытие чека</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
                <receipt_number>285</receipt_number>
                <receipt_type>5</receipt_type>
                <type_str>ВОССТАНОВЛЕНИЕ</type_str>
<restore>ВОССТАНОВЛЕНИЕ ПОСЛЕ СБОЯ</restore>
                <cash_number>9999999</cash_number>
                <shift_number>5</shift_number>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

> **Примечание.**
>
> В этом примере показано открытие чека после сбоя, о чем свидетельствует значение параметра receipt_type. Подробнее о возможных значениях этого параметра см. Поля и их значения.

12. **Закрытие смены**

```
<TransactionBlock>
                <FunctionNumber>12</FunctionNumber>
<FunctionName>Закрытие смены</FunctionName>
                <receipt_number>286</receipt_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

13. **Скидка**

```
<TransactionBlock>
                <FunctionNumber>13</FunctionNumber>
<FunctionName>Скидка</FunctionName>
                <receipt_number>10</receipt_number>
                <discount_name>Hand_disk</discount_name>
                <receipt_amount>0.00</receipt_amount>
                <receipt_discount>0.00</receipt_discount>
<receipt_discount_number>0</receipt_discount_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

14. **Открытие денежного ящика**

```
<TransactionBlock>
                <FunctionNumber>14</FunctionNumber>
<FunctionName>Открытие денежного ящика</FunctionName>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

15. **Денежная операция**

```
<TransactionBlock>
                <FunctionNumber>15</FunctionNumber>
<FunctionName>Денежная операция</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
                <receipt_number>288</receipt_number>
                <type>101</type>
```

```
                <type_str>ИЗЪЯТИЕ</type_str>
                <payment_name>Наличные</payment_name>
                <payment_amount>16050.00</payment_amount>
                <shift_number>6</shift_number>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

> **Примечание.**
>
> В данном примере показана функция для изъятия денег, о чем свидетельствует значение параметра type. Возможно также **внесение** денег в кассу, см. п. Поля и их значения.

16. **Платежная карта**

```
<TransactionBlock>
                <FunctionNumber>16</FunctionNumber>
<FunctionName>Платежная карта</FunctionName>
                <receipt_number>10</receipt_number>
                <enter_type>1</enter_type>
                <card_number>4895</card_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

17. **Дисконтная карта**

```
<TransactionBlock>
                <FunctionNumber>17</FunctionNumber>
<FunctionName>Дисконтная карта</FunctionName>
                <receipt_number>10</receipt_number>
                <enter_type>1</enter_type>
                <card_number>2367</card_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

18. **Ручная скидка**

```
<TransactionBlock>
                <FunctionNumber>18</FunctionNumber>
<FunctionName>Ручная скидка</FunctionName>
                <receipt_number>10</receipt_number>
                <type_str>СКИДКА НА ТОВАРНУЮ ПОЗИЦИЮ</type_str>
                <item_pos>1</item_pos>
                <item_discount>-15</item_discount>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

19. **Печать копии чека**

```
<TransactionBlock>
                <FunctionNumber>19</FunctionNumber>
<FunctionName>Печать копии чека</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
                <receipt_number>291</receipt_number>
                <receipt_type>0</receipt_type>
                <type_str>ПРОДАЖА</type_str>
                <cash_number>9999999</cash_number>
                <shift_number>6</shift_number>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

20. **Открытие денежного ящика без транзакции**

```
<TransactionBlock>
                <FunctionNumber>20</FunctionNumber>
<FunctionName>Открытие денежного ящика без транзакции</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

21. **Z - отчёт**

```
<TransactionBlock>
                <FunctionNumber>21</FunctionNumber>
<FunctionName>Z - отчёт</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <shift_number>5</shift_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

22. **X - отчёт**

```
<TransactionBlock>
                <FunctionNumber>22</FunctionNumber>
<FunctionName>X - отчёт</FunctionName>
                <user_name>Петров</user_name>
                <user_id>19</user_id>
                <shift_number>6</shift_number>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

23. **Контрольный пакет**

```
<TransactionBlock>
<FunctionNumber>77777</FunctionNumber>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

24. **Ручной ввод**

```
<TransactionBlock>
<FunctionNumber>23</FunctionNumber>
<FunctionName>Ручной ввод</FunctionName>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

25. **Попытка запрещенной операции**

```
<TransactionBlock>
<FunctionNumber>25</FunctionNumber>
<FunctionName>Попытка запрещенной операции</FunctionName>
<type_str>Название операции</type_str>
                <date>2006-09-07 15:20:45.051</date>
<TransactionTimestamp>2009-01-01T04:04:33.004</TransactionTimestamp>
</TransactionBlock>
```

3 Использование парсера с расширением .prl

3.1 Общие сведения о создании парсеров с расширением .prl в ПК POS-Интеллект

Парсер с расширением .prl определяет правила заполнения базы данных чеков в случае, если данные от POS-терминала передаются на POS-сервер по протоколу, отличному от XML. При создании парсера используются регулярные выражения.

Создание парсеров осуществляется в следующем порядке:

1. Создать шаблоны с использованием утилит parser_designer.exe и ParserTest.exe.
2. Добавить созданные шаблоны в парсер средствами *POS-Интеллект*.

> **Примечание.**
>
> Способов записи шаблонов на один и тот же фрагмент текста может быть множество. Поэтому приведенные в данном разделе примеры не обязательно являются единственно верными.

3.2 Регулярные выражения

> **Примечание.**
>
> За основу нижеприведенного описания регулярных выражений взят текст с сайта http://phpclub.ru , адрес статьи http://phpclub.ru/detail/article/regexp_1

3.2.1 Общие сведения о регулярных выражениях

Для анализа кассовых операций требуется разобрать поток данных, приходящий от кассового терминала, и выбрать нужные сведения. Для этого используются регулярные выражения.

Регулярные выражения - язык шаблонов.

Обработка шаблона происходит посимвольно. Например, чтобы найти букву d в слове stadium, нужно перебрать все буквы слова stadium и сравнить их с буквой, которую требуется найти, что представляет собой простой перебор.

Шаблон - это своеобразный указатель, что искать в строке. Искать можно цифры, буквы, невидимые символы (пробел, табуляция и т.д.).

Чтобы в слове stadium найти букву d либо букву m, пока не произойдет первое совпадение одного из символов, необходимо также воспользоваться перебором: брать каждую букву слова и сравнивать с тем, что нужно найти, т.е. поочередно с d и с m. Но каждый раз теперь придется сравнивать букву слова (строки) с двумя буквами условия поиска. Такой набор условий поиска называется символьным классом и записывается на языке регулярных выражений следующим образом: [dm] - это означает, что ищется либо d либо m

Чтобы указать, что ищется любая буква алфавита, нужно либо перечислить их все [abc....xyz], либо просто указать интервал [a-z].

Такое возможно с русскими буквами [а-я], а так же с цифрами [0-9], но есть еще заглавные [A-Z], т.е. чтобы получить символьный класс со всеми буквами латинского алфавита требуется поставить в шаблоне [a-zA-Z]

Такой символьный класс описывает только один символ, а в строке их множество, это решается при помощи квантификаторов.

3.2.2 Квантификаторы

Допустим, имеются две строки:

abcd12345efg
fghi56789qwe

Условие: Найти в строках участки, которые состоят из четырех любых букв латинского алфавита, после которых следуют пять любых цифр.

Выше рассказывалось, как описать символ, который будет совпадать с любой латинской буквой (для этого используется символьный класс: [a-z] (пока не обращаем внимания на то, что есть еще заглавные буквы)), а также символ условия, который совпадет с любой цифрой: [0-9]

Требуется найти строки с четырьмя буквами вначале, сразу после которых идут пять цифр.

Можно написать: [a-z][a-z][a-z][a-z][0-9][0-9][0-9][0-9][0-9]

Такая форма записи условия поиска будет работать. Так как мы описали каждым символьным классом один символ в условии поиска. Конструкция кажется слишком громоздкой. В данной ситуации следует использовать квантификаторы. Квантификатор выражает количество символов в условии поиска. Условие поиска упрощается при помощи квантификаторов: [a-z]{4}[0-9]{5}

В фигурных скобках написано, сколько символов, описанных в символьном классе, может идти подряд в строке, в которой ищется совпадение.

Таким образом, после четырех любых букв латинского алфавита идет пять любых цифр. Каждый символьный класс описывает только один символ, количество схожих символов идущих подряд описывается квантификаторами.

Подобное задание количества символов в условии поиска не является единственным. Квантификаторы бывают разные: [a-z]{1,3} означает, что подряд может идти от одного до трех букв латинского алфавита. [a-z]{2,} означает, что может идти минимум 2 буквы латинского алфавита подряд.

Но квантификатор в фигурных скобках не является единственным способом задать количество символов идущих подряд, которые описаны символьным классом. [a-z]* означает, что подряд может идти сколь угодно букв латинского алфавита, а может быть, что и ни одной, идентично [a-z]{0,}. [a-z]+ означает, что обязательно подряд должна идти минимум одна буква латинского алфавита, но максимальное количество не указано, идентично [a-z]{1,} [a-z]? означает, что количество латинских букв не должно превышать 1, буква также может вообще отсутствовать, идентично [a-z]{0,1}.

Условие поиска совпадения, которое описывается одним неспециальным символом, называется литералом.

К литералам можно применять любой из вышеприведенных квантификаторов.

Ясно, что если применить условие поиска abcd{1,4}efg для строки abcefg, совпадение найдено не будет, так как квантификатор {1,4} подразумевает, что после abc перед efg идет минимум одна, максимум четыре буквы d.

3.2.3 Символьные классы

В символьный класс может входить любой литерал, а так же интервалы литералов. Для описания интервалов литералов используется символ '-', который ставится между первым символом интервала и последним. Примеры задания различных интервалов в одном символьном классе: [1-5] - числа в

диапазоне от 1 до 5, [a-f] - буквы латинского алфавита от a до f, [a-fq-x] - буквы латинского алфавита от a до f и от q до x, в последнем символьном классе используются два диапазона.

Если в определенном месте строки могут стоять символы: либо a, либо g, либо 7, либо 4, то символьный класс будет иметь вид: [ag47]

В символьном классе можно перечислять допустимые в условии поиска - литералы. Перечисление литералов можно совмещать с указанием интервалов: [14a-kz] - это означает, что символ в строке может совпадать с 1, 4, буквами латинского алфавита с a по k, а так же с буквой z. Естественно литералами могут быть не только буквы и цифры, а так же знаки препинания, математические знаки, например ',' (запятая), '!' (восклицательный знак), '+' (плюс). Можно использовать и '-' (минус), даже если он же используется и для описания интервалов. Если поставить минус между a и z, то это будет интервал, но если - сразу же после открытой квадратной скобки, то это будет минус. Пример: [-,a-z] - означает, что в символьный класс входят минус, запятая, а так же буквы латинского алфавита от a до z.

Для того, чтобы написать символьный класс, в который входят все символы кроме заданных, например, все кроме a, b, c, существует специальный спецсимвол отрицания: ^ (крышка). Нужно написать: [^abc] - все символы (не буквы, а именно символы) кроме букв латинского алфавита a, b, c.

Потренироваться в написании шаблонов можно здесь : http://www.pcre.ru/eval/.

3.2.4 Спецсимволы

Для того, чтобы указать, например, что в условии поиска присутствует пробел, необходимо сделать его видимым, т.е. ввести какой-то символ, набор символов, которые будут интерпретироваться, как невидимые.

\s - если в условии поиска поставить друг за другом символ обратного слеша, а после него сразу букву s, то таким образом будет описан либо пробел, либо символ табуляции. Конечно, в условии поиска можно поставить пробел так, как он обычно ставится на письме, но запись [a-z\s] будет намного понятнее и читабельней чем [a-z]. Внимательно используйте этот спецсимвол, так в дополнении к тому, что он совпадает с пробелом и табулятором, он совпадет также с символом новой строки.

\S - это видимые символы, т.е. все, что не совпадает с \s

\w - спецсимвол, который призван заменить целый символьный класс, в него входят все символы, которые могут входить в слово, обычно это [a-zA-Z_], хотя много может зависеть от установленной локали, поддержки юникода и т.д.

\W - все что не входит в определение \w. либо [^a-zA-Z_]

\d - все цифры, т.е. символьный класс [0-9]

\D - все, что не является цифрой

Часто спецсимволы описывают часто используемые множества символов, но эти множества имеют границы, т.е. либо цифры, либо буквы и подчеркивание, либо невидимые символы. Для описания всех символов следует использовать точку. Пример:

.{5} – означает 5 любых символов.

Если надо описать именно точку, а не все символы, следует поставить перед точкой обратный слеш: \.

Если вы ищите в тексте обратный слеш, после которого идет точка, чтобы поставить обратный слеш в виде литерала в условии поиска, надо его удвоить: \\

Аналогично, чтобы поставить два обратных слеша их надо тоже удвоить вот так: \\\\

3.2.5 Условия выбора

Имеется возможность описать условие выбора при помощи регулярных выражений. В качестве примера возьмем две группы литералов, одна группа состоит из двух литералов идущих друг за другом b и e, вторая группа состоит из девяти литералов идущих друг за другом: n, o, t, \s, t, o, \s, b, e

Понятно, что \s представляет собой один пробел. Группа литералов - последовательность символов, которые описаны либо символьными классами, либо собственно литералами. Группу литералов описывают в круглых скобках. Они же сохраняют совпавшую группу литералов в специальных переменных. Вот примеры групп литералов:

(be)
(not\sto\sbe)

Выбор из двух групп литералов: (be)|(not\sto\sbe), символ ‘|’ между группами литералов и есть условие выбора, читается как 'или'. Регулярное выражение проверки:

(be)|(not\sto\sbe)

Регулярное выражение совпадет в случае

- если строка равна "be"
- если или строка равна "not to be"

Выбирать можно между литералами и между группами литералов. Группы литералов объединяются круглыми скобками, если надо выбрать между одиночными литералами, то два литерала, между которым стоит вертикальная черта, нужно сгруппировать скобками.

Пример выбора из двух литералов: s(o|u)n совпадет как с son, так и с sun.

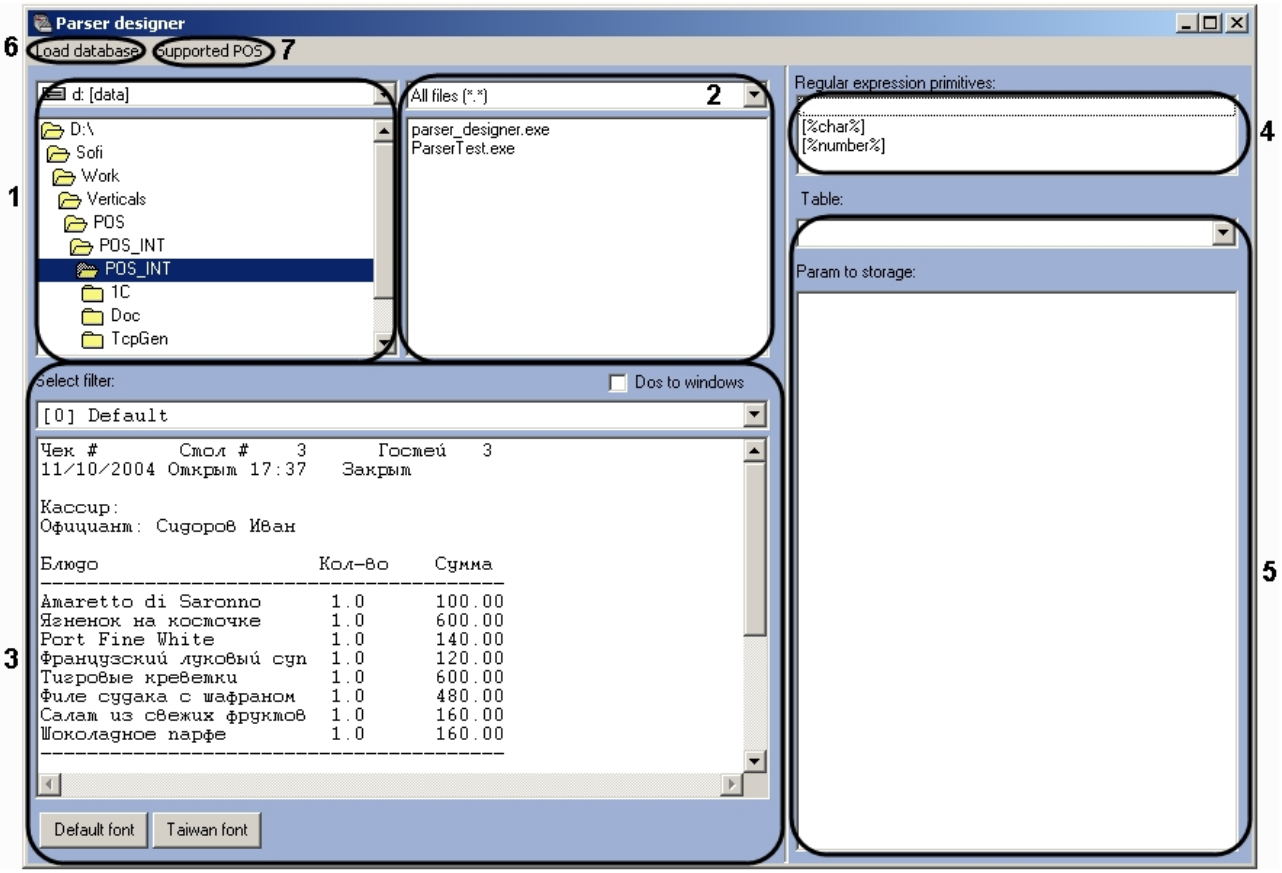
Пример выбора из двух групп литералов: (son)|(sun) аналогично совпадет с son и с sun

В случае с выбором между группами литералов, либо между одиночными литералами, литералы объединяются при помощи круглых скобок.

Если надо поставить литерал "|", то надо перед ним поставить спецсимвол, который будет обозначать, что в данном случае вертикальная черта является литералом. Этим спецсимволом служит обратный слеш: \|

3.3 Описание интерфейса утилиты parser_designer.exe

Внешний вид окна утилиты parser_designer.exe представлен на рисунке.



В рабочей области окна утилиты расположены области, предназначенные для выполнения следующих функций:

1. Выбор папки с логом от POS – терминала (1).
2. Выбор лога (2).
3. Отображение текста лога (3).
    a. Раскрывающийся список **Select filter** предоставляет возможность выбора протокола.
    b. Флажок **Dos to windows** предназначен для перекодировки символов, если текст лога представлен в кодировке DOS.
    c. Кнопки **Default font** и **Taiwan font** предназначены для выбора используемых шрифтов.
4. Шаблоны регулярных выражений (4).
5. Список полей таблицы, выбранной в поле **Table,** из подключенной БД (5). Для работы используются таблицы POS_LOG_MASTER и POS_LOG_DETAIL.
    a. Таблица POS_LOG_MASTER хранит в себе данные о заголовке и итоговой части чека: номер чека, имя и код кассира, сумма чека, сумма сдачи и т.д.
    b. POS_LOG_DETAIL хранит данные «тела» чека, т.е. информацию о товаре пробитом в этом чеке.

Главное меню утилиты предоставляет доступ к следующим функциям:

1. **Load database** – настройка подключения к базе данных POS (6).

> **Примечание.**
>
> На момент написания данного документа для корректного подключения к базе данных POS требовалось использовать учетные сведения Windows, т.к. в утилите была

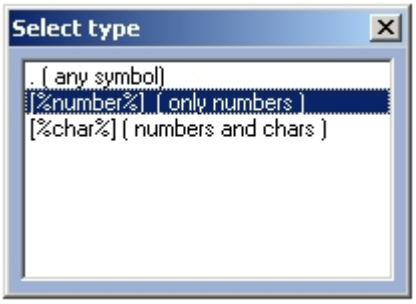
отключена функциональная возможность сохранения пароля при подключении к базе данных с использованием имени пользователя и пароля.

2. **Supported POS** – отображение списка поддерживаемых POS-терминалов в данной версии утилиты (7).

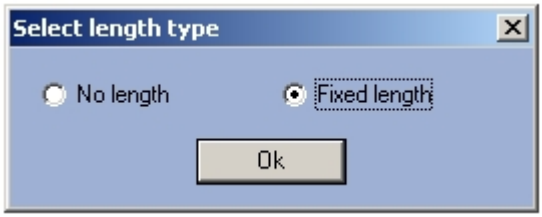
3.4 Создание парсера

Создание парсера осуществляется следующим образом:

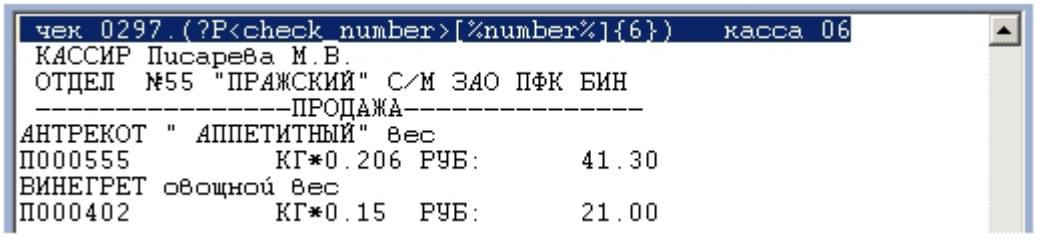
1. Выделить в окне с текстом лога нужный фрагмент текста. В окне 5 выделить поле в таблице, в которое нужно записать фрагмент текста. Нажать левую клавишу мыши и перетащить поле в окно с тестом лога.
2. В появившемся окне выбрать формат шаблона. Доступны следующие форматы:
   a. . - любой символ.
   b. [%number%] - только цифры и десятичная точка.
   c. [%char%] - цифры и символы.



3. Выбрать тип длины. Доступны следующие типы длины:
4. No length – неограниченная длина.
5. Fixed length – фиксированная длина, считается автоматически по выделенному фрагменту.

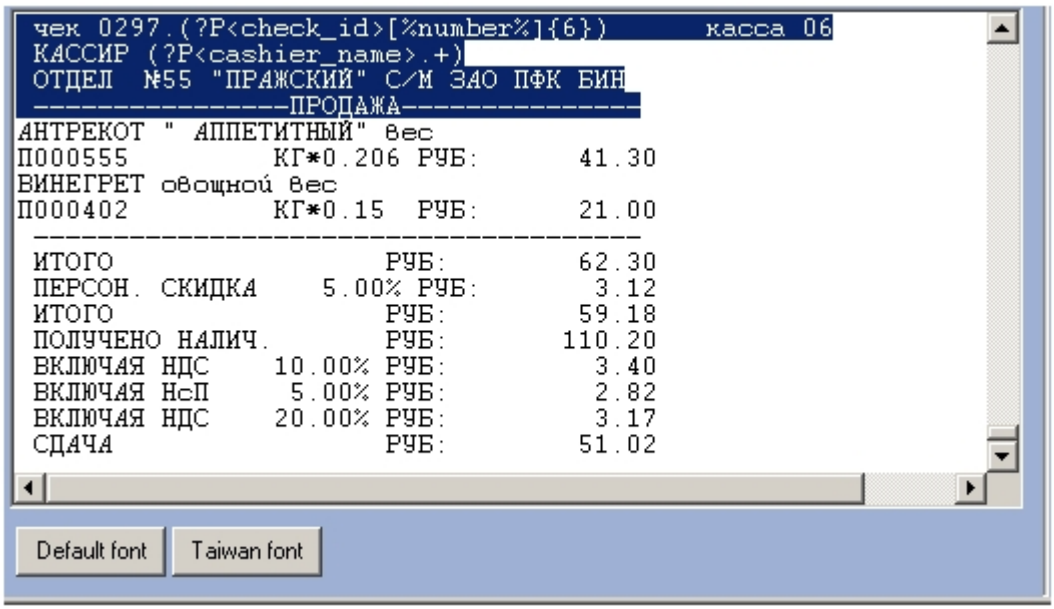


6. Строка в поле лога принимает вид шаблона с указанием имени поля. Конструкция вида ?P<item_name> указывает поле таблицы, в которое будет записано распознанное парсером значение – например, это поле check_number. При необходимости можно изменить следующее за данным текстом регулярное выражение.



7. Таким же образом создать шаблоны для всех требуемых полей базы данных.

Например, таким образом будет выглядеть шаблон для имени кассира (здесь длина не ограничена).

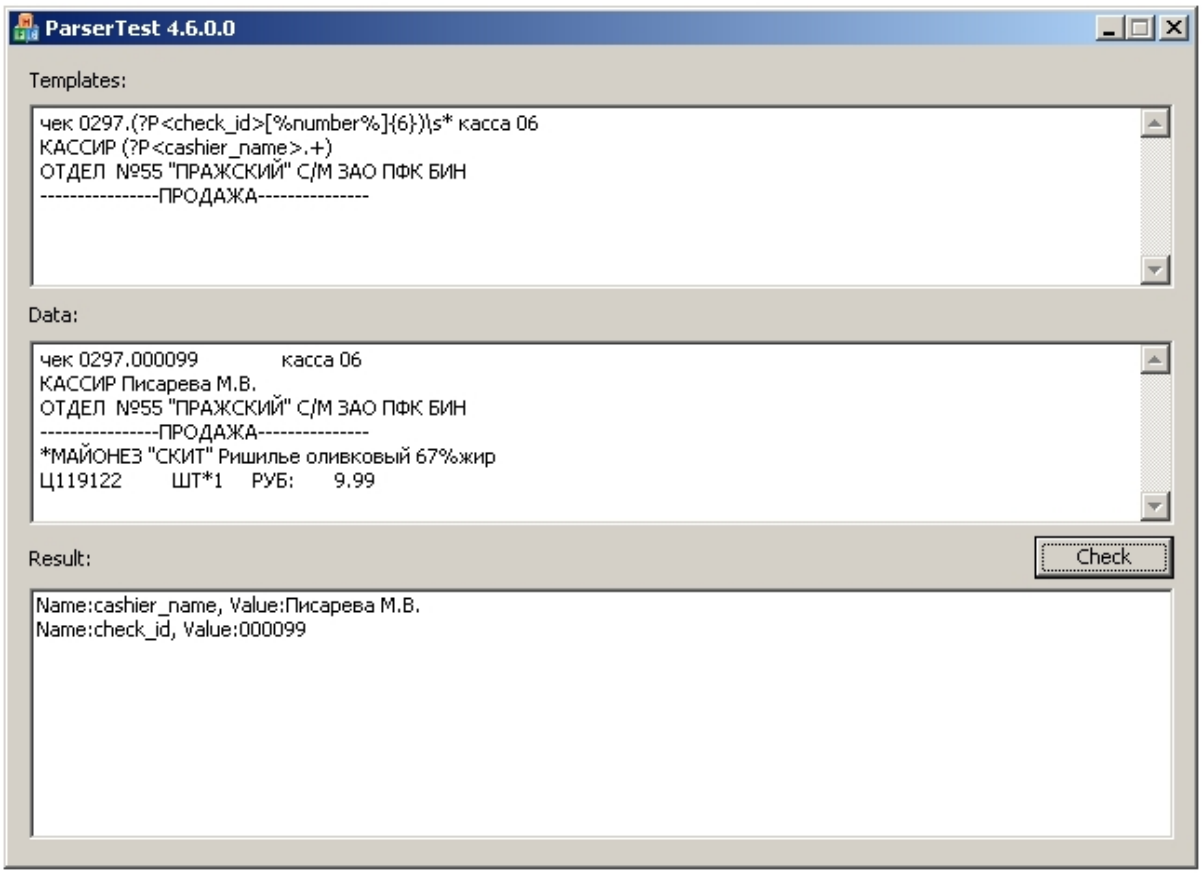


Выделенный на рисунке фрагмент есть шаблон заголовочной части чека.

> **Примечание.**
>
> Следует учитывать, что данные из тела чека хранятся в таблице, отличной от той, в которой хранятся данные из заголовка и итоговой части чека, поэтому при переходе к созданию парсера для тела чека следует выбрать соответствующую таблицу из раскрывающегося списка **Table**.

3.5 Проверка созданного шаблона

Необходимо проверить, как отрабатывается созданный шаблон. Для проверки используется утилита ParserTest.exe



Проверка производится следующим образом:

1. Фрагмент парсера необходимо скопировать в поле **Templtes**.
2. В поле **Data** поместить фрагмент текста чека.
3. Нажать кнопку **Check.**
4. Если шаблон составлен правильно, то в поле **Result** отобразятся наименования полей и их значения, которые будут записаны в базу. Если ничего не появилось, значит в шаблоне присутствует ошибка.

   > **Примечание.**
   >
   > Роль могут играть лишние пробелы в начале строк, знаки конца строки и т.д. Как видно в примере на рисунке, в окончательном варианте шаблон:
   > **«(?P<check_id>[%number%]{6}) касса 06 »**
   > с рисунка в разделе Создание парсера видоизменен следующим образом:
   > **«(?P<check_id>[%number%]{6})\s*касса 06».**
   > Пробелы заменены на «\s*», чтобы избежать разночтения в количестве пробелов.

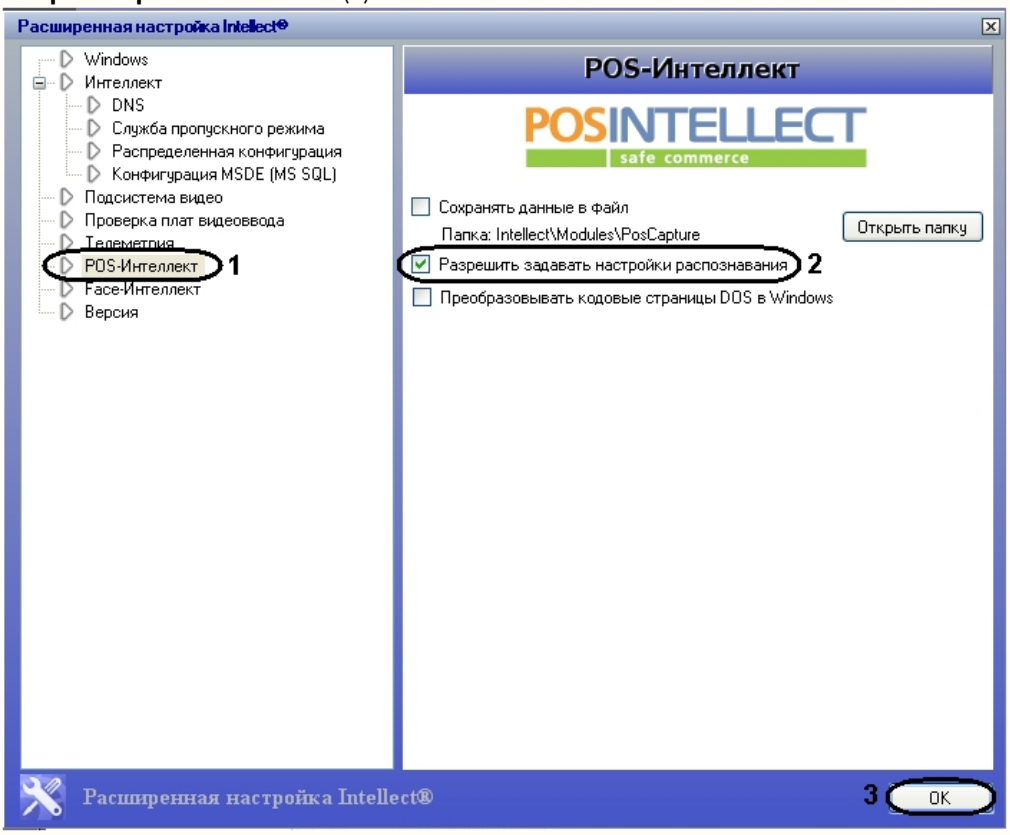
5. Если данные в поле **Result** отобразились верно, то необходимо перенести шаблон в настройки парсера объекта **POS-терминал** при помощи функций копирования и вставки текста (см. раздел Настройка парсера в ПК POS-Интеллект).

> **Примечание.**
>
> В случае примеров, рассматриваемых на рисунках, шаблон необходимо поместить в раздел **Пролог чека**. Аналогичным образом добавляются остальные разделы чека.

3.6 Настройка парсера в ПК POS Интеллект

Внимание!

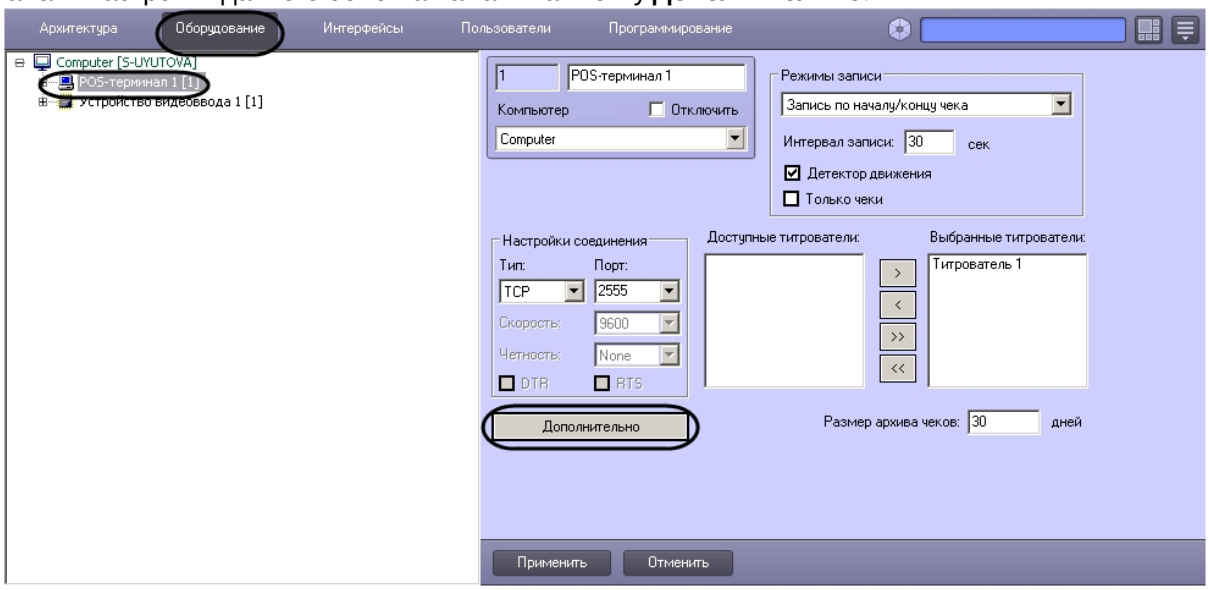
Чтобы получить возможность редактировать шаблоны, необходимо запустить программу tweaki.exe, перейти в раздел **POS-Интеллект** (1) и установить флажок **Разрешить задавать настройки распознавания** (2).



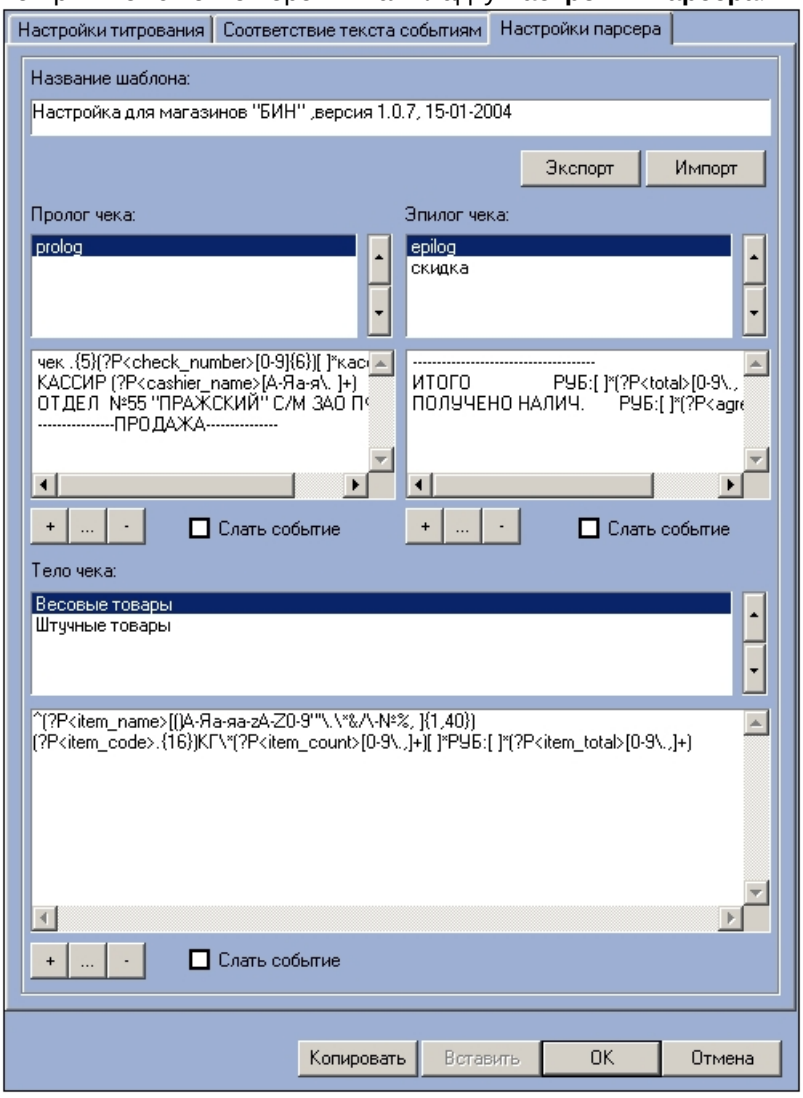
После того, как созданы и проверены шаблоны для всех требуемых полей, необходимо сформировать файл парсера. Для этого следует выполнить следующие действия:

1. Запустить ПК *POS-Интеллект*, Перейти на вкладку **Оборудование** диалогового окна **Настройка системы** и выбрать требуемый объект **POS-терминал**, для которого настраивается парсер. На

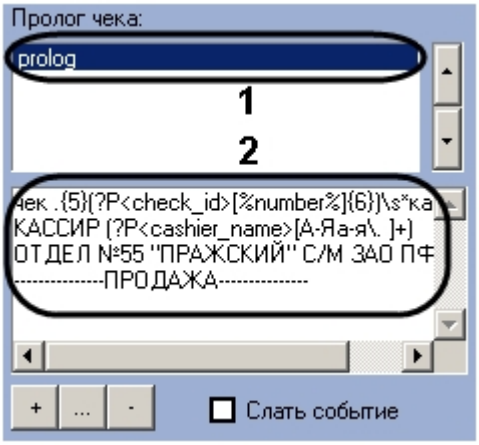
панели настройки данного объекта нажать на кнопку **Дополнительно**.



2. В открывшемся окне перейти на вкладку **Настройки парсера**.



3. Настроить пролог/эпилог/тело чека:
   a. При помощи кнопки + добавить правило структурирования (1).



   b. Скопировать текст правила структурирования, сформированный при помощи утилиты parser_designer.exe и проверенный при помощи утилиты ParserTest.exe (2).

   c. Повторять шаги 3.a-3.b для всех требуемых правил структурирования пролога/эпилога/тела чека.
4. Указать название парсера в поле **Название шаблона**.
5. Нажать на кнопку **Экспорт** для сохранения парсера в виде файла с расширением .prl.

> **Примечание.**
>
> Более подробно процесс создания парсера описан в документе Программный комплекс POS-Интеллект. Руководство Администратора.

4 Библиотеки для работы с ПК POS Интеллект

> **Внимание!**
>
> Библиотеки, входящие в состав POS SDK, приложены в качестве примеров. Компания ITV не несет ответственности за их работу.

4.1 COM. Библиотека для работы через TCP/IP соединение

Библиотека в основном предназначена для интеграции кассового ПО (работающего под Windows) с системой **POS-Интеллект**. Она скрывает от программиста код обеспечения связи через TCP/IP. Например, имея код отправки данных на чековый принтер, можно путем незначительной модификации кода программы переправлять данные в сеть, используя протокол TCP/IP. Библиотека предоставляем COM объект, который можно использовать из любой среды, поддерживающей COM технологию (например, Visual Basic, Delphi, 1C).

> **Внимание!**
>
> Предварительно необходимо зарегистрировать библиотеку:
> regsvr32.exe poslib.dll

Пример использования (для VB.NET) :

Создание объекта:

```
Dim pos As Object
pos = CreateObject("Poslib.Net")
```

Начать процесс установки соединения:

```
Dim port As System.UInt32
port = Convert.ToUInt32(“5000”)
pos.Open(ip, port)
```

Используемый метод:

HRESULT Open(BSTR ip_address,DWORD port) - вызывается в начале работы с библиотекой, инициирует установку связи

- ip_address – адрес сервера
- port – порт для связи

Послать текст:

```
pos.Send(“Test!” & vbNewLine)
```

Используемый метод:

HRESULT Send(BSTR str);

- **-** str – сообщение, посылаемое системе

Закрыть соединение:

```
pos.Close()
```

Используемый метод:

• HRESULT Close() – вызывается в конце работы с библиотекой

При вызове метода **Open** библиотека сама осуществляет подсоединение через TCP/IP и восстановление соединения в случае потери связи. Все вызовы происходят асинхронно и не влияют на основной поток приложения, вызывающего их. Отправка данных также происходит из отдельного потока. Все используемые методы потокобезопасны.

При вызове метода **Send** в случае наличия связи данные будут отправлены, в случае ее отсутствия отправки не произойдет. Никакого подтверждения удачи/неудачи отправки данных не происходит.

В настройках объекта **POS -терминал** системы **POS-Интеллект** необходимо указать тип соединения TCP и ввести порт, заданный в качестве метода функции **Open**.

4.2 ActiveX компонент для работы через TCP/IP соединение

Библиотека в основном предназначена для интеграции кассового ПО (работающего под Windows) с системой **POS-Интеллект**. Она скрывает от программиста код обеспечения связи через TCP/IP. Например, имея код отправки данных на чековый принтер, можно путем незначительной модификации кода программы переправлять данные в сеть, используя протокол TCP/IP. Библиотека предоставляем ActiveX компонент, который можно использовать из любой среды, поддерживающую ActiveX технологию (например, Visual Basic, Delphi, 1C).

> ⚠ **Внимание!**
>
> Предварительно необходимо зарегистрировать библиотеку:
> regsvr32.exe posx.ocx

Используемые методы:

1. Open(BSTR ip_address,DWORD port) - вызывается в начале работы с библиотекой, инициирует установку связи
    a. ip_address – адрес сервера
    b. port – порт для связи
2. Send(BSTR str);
    a. str – сообщение, посылаемое системе
3. Close() - вызывается в конце работы с библиотекой

При вызове метода **Open** библиотека сама осуществляет подсоединение через TCP/IP и восстановление соединения в случае потери связи. Все вызовы происходят асинхронно и не влияют на основной поток приложения, вызывающего их. Отправка данных также происходит из отдельного потока. Все используемые методы потокобезопасны.

При вызове метода **Send** в случае наличия связи данные будут отправлены, в случае ее отсутствия отправки не произойдет. Никакого подтверждения удачи/неудачи отправки данных не происходит.

В настройках объекта **POS -терминал** системы **POS-Интеллект** необходимо указать тип соединения TCP и ввести порт, заданный в качестве метода функции **Open**.

4.3 DLL. Библиотека для работы через TCP/IP соединение

Библиотека в основном предназначена для интеграции кассового ПО (работающего под Windows) с системой **POS-Интеллект**. Она скрывает от программиста код обеспечения связи через TCP/IP. Например, имея код отправки данных на чековый принтер, можно путем незначительной модификации кода программы переправлять данные в сеть, используя протокол TCP/IP.

Функции библиотеки:

1. **void __stdcall Open (LPCTSTR id,LPCTSTR ip_address,DWORD port)** – вызывается в начале работы с библиотекой, инициирует установку связи

- ip_address – адрес сервера
- port – порт для связи

1. **void __stdcall Close (LPCTSTR id)** – вызывается в конце работы с библиотекой
2. **void __stdcall Send(LPCTSTR id,LPCTSTR str)**

- str – сообщение посылаемое системе.

Общий для всех функций параметр **LPCTSTR id** является идентификатором соединения. Допускается иметь параллельно несколько соединений с разными серверами, различая их по этому параметру.

При вызове функции **Open** библиотека сама осуществляет подсоединение через TCP/IP и восстановление соединения в случае потери связи. Тип вызова функций **__stdcall**. Все вызовы происходят асинхронно и не влияют на основной поток приложения, вызывающего их. Отправка данных также происходит из отдельного потока. Все используемые функции потокобезопасны.

При вызове функции **Send** в случае наличия связи данные будут отправлены, в случае ее отсутствия отправки не произойдет. Никакого подтверждения удачи/неудачи отправки данных не происходит.

В настройках объекта **POS -терминал** системы **POS-Интеллект** необходимо указать тип соединения TCP и ввести порт, заданный в качестве параметра функции **Open**.

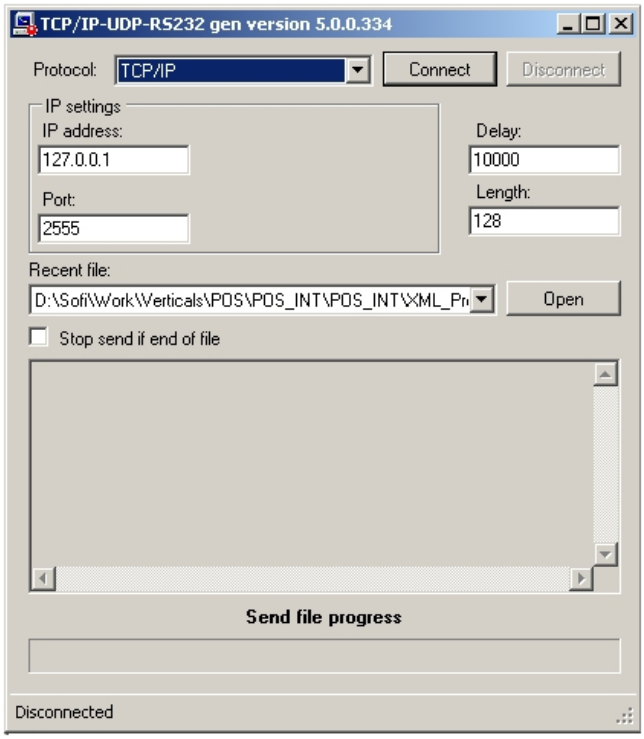
5 Взаимодействие с 1С

В комплект POS SDK входит пример программы 1С, позволяющей посылать сообщения в *POS-Интеллект*.

6 Приложение 1. Утилита tcpgen.exe для отправки тестовых данных в POS Интеллект

6.1 Общие сведения об утилите tcpgen.exe

Утилита tcpgen.exe позволяет отправлять в *POS-Интеллект* лог с кассового терминала с целью проверки работы настроенного объекта **POS-терминал**. Интерфейс утилиты представлен на рисунке.



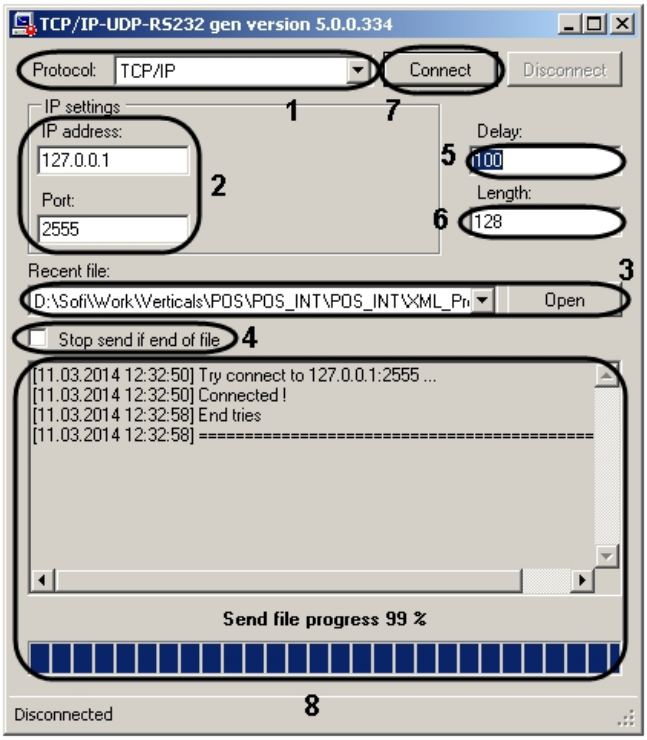
Посылаемые данные будут обрабатываться *POS-Интеллект* в рабочем режиме: будет производиться наложение титров на видеоизображение, запись данных в базу данных и пр.

> **Примечание.**
>
> Загрузить данную утилиту можно в составе POS SDK на данной странице.

6.2 Работа с утилитой tcpgen.exe

Работа с утилитой tcpgen.exe осуществляется в следующем порядке:

1. Запустить утилиту tcpgen.exe.



2. Из раскрывающегося списка **Protocol** выбрать транспортный протокол, по которому следует отправлять данные в *POS-Интеллект* (1). Доступны для выбора протоколы TCP/IP, UDP и RS232.
3. Если параметры подключения по умолчанию не подходят, настроить параметры подключения (2). При использовании TCP/IP и UDP задаются IP-адрес и порт подключения к *POS-Интеллект*, при использовании протокола RS232 задаются номер и скорость порта.
4. Нажать на кнопку **Open** и с помощью стандартного диалога открытия файлов Windows выбрать текстовый файл, содержащий лог от кассового терминала (3).
5. В случае, если следует прекратить пересылку данных в *POS-Интеллект* при достижении конца файла лога, установить флажок **Stop send if end of file** (4). В случае, если отправку данных необходимо продолжать циклически, оставить данный флажок снятым.
6. В поле **Delay** указать требуемую задержку между пакетами в миллисекундах (5).
7. В поле **Length** указать длину пакета в байтах (6).
8. Нажать на кнопку **Connect** (7).
9. Процесс пересылки будет отображаться в поле статуса (8). В случае, если по каким-то причинам пересылка данных невозможна, в данном поле будет приведено описание проблемы.

> **Примечание.**
>
> Для остановки пересылки данных нажать на кнопку **Disconnect**.

Работа с утилитой tcpgen.exe завершена.