

Dual Connector 2.0

Руководство программиста

Версия 2.0.10

История изменений документа

Версия	Дата	Перечень изменений
1.0	19.04.2019	Первоначальная версия (Версия DC Service 1.0.0.5)
1.1	18.07.2019	В пункт «2.3 Инсталляция в системе» добавлена сноска «Важно»; Добавлено сноска для поля 89 в пункт 8.2 (Версия DC Service 1.0.1.0)
1.2	22.01.2019	Изменено описание «DC Proxy» в пункт «2.1 Назначение»; Удален пункт «4.1 Интеграция через библиотеку “DC Proxy”». Изменены пункты «5.1 C++», «5.2 C#», «5.3 Java»
1.3	08.04.2020	Актуализирован пункт «3 Прямая интеграция с DC Service по HTTP»
1.4	21.09.2020	В пункте 2.2 добавлена поддержка Java8
1.5	15.02.2021	Внесены изменения в пункты: «2.2 Системные требования», «3 Прямая интеграция с DC Service по HTTP», «4.1 Основные параметры».
1.6	14.09.2021	Актуализированы пункты: «3 Прямая интеграция с DC Service по HTTP», «4.1 Основные параметры».
1.7	19.10.2022	Добавлен пункт «4.2 Параметры по TerminalID»
1.8	19.10.2023	Актуализирован пункт «3 Прямая интеграция с DC Service по HTTP», «4.1 Основные параметры», «4.2 Параметры по TerminalID» Добавлен пункт «8.1 Порядок использования API», «2.4 Создание файла параметров «Connector.xml» в директории установки»
1.9	31.10.2023	Актуализирован пункт «2.3 Инсталляция в системе», «3 Прямая интеграция с DC Service по HTTP». Добавлен пункт «4.3 Настройки параметров DC Service GUI»

Содержание

Правовая информация и сведения о поддержке продукта.....	4
1. Введение.....	5
2. Dual Connector 2.0.....	6
2.1. Назначение.....	6
2.2. Системные требования	7
2.3. Установка в системе	7
2.4. Создание файла параметров «Connector.xml» в директории установки.....	7
2.4.1. Создание и настройка файла параметров при помощи «XML Generator»	7
2.4.2. Копирование заранее созданного файла параметров во время установки.	7
2.4.3. Создание нового файла конфигурации при помощи ключей параметров во время установки.....	7
3. Прямая интеграция с DC Service по HTTP	9
4. Настройка параметров «DualConnector 2.0»	12
4.1. Основные параметры.....	12
4.2. Параметры по TerminalID.....	13
4.3. Настройки параметров DC Service GUI.....	14
5. Примеры работы с «DC Service».....	15
5.1. C++	15
5.2. C#	16
5.3. Java.....	16
6. Особенности использования DC Proxy	18
7. Взаимодействие с оператором	19
8. Приложение	24
8.1. Порядок использования API.....	24
8.1.1. Описание API	24
8.2. Свойства объекта «SAPacket».....	28

Правовая информация и сведения о поддержке продукта

Dual Connector 2.0. Версия 2.0.10 Руководство программиста: М.: ООО "Лаборатория платежных решений", 2023. — 29с.

ООО "Лаборатория платежных решений" оставляет за собой право производить незначительные изменения программного обеспечения, касающиеся функциональности и внешнего вида конфигурационных систем, без внесения изменений в настоящее Руководство без специального уведомления.

Программное обеспечение и настоящий документ не могут быть скопированы, размножены, использованы по частям для составления других текстов, переведены на другие языки, если это не оговорено в письменной форме в договоре на поставку программного обеспечения.

Программное обеспечение, описанное в настоящем Руководстве, поставляется в соответствии с договором о поставке и может использоваться или копироваться только в соответствии с условиями этого договора.

Разработчиком и правообладателем программы Dual Connector 2.0 является ООО "Лаборатория платежных решений".

Dual Connector 2.0 Версия 2.0.10 © ООО "Лаборатория платежных решений" 2023

Для зарегистрированных пользователей ПО Dual Connector 2.0 открыты линии телефонных и E-Mail-консультаций. На консультацию имеет право пользователь, который приобрел ПО Dual Connector 2.0 в компании Лаборатория платежных решений.

Линия телефонных консультаций работает с понедельника по четверг с 10.00 до 18.00, в пятницу с 10.00 до 17.00 часов по московскому времени, кроме выходных и праздничных дней.

На линиях консультаций работают квалифицированные специалисты, которые ответят на Ваш вопрос немедленно или, возможно, попросят сформулировать вопрос в письменном виде и отправить по E-Mail.

1. Введение

Данный документ предназначен для прочтения программистами, осуществляющими организацию взаимодействия между клиентским ПО (в дальнейшем Клиент) и терминалами с ПО «SmartSale».

2. Dual Connector 2.0

2.1. Назначение

«DualConnector 2.0» — сервис для интеграции кассового ПО на базе «Windows (XP/7/8/10)» 32-х и 64-х разрядных систем, реализующую интерфейс обмена с терминалом по протоколу SA. Обмен данными между кассой и «DualConnector 2.0» выполняется с помощью «HTTP-запросов».

«DualConnector 2.0» предоставляет возможность терминалу, подключенному по COM/ USB или TCP/IP, работать в режиме клиента и самостоятельно инициировать сеанс связи с коннектором. Благодаря этому при отсутствии у терминала своего канала связи с внешней сетью возможно независимое от кассового ПО взаимодействие с серверами сети «TCP/ IP» через сервис, используя коммуникации кассы.

Кассовый сервер «DualConnector 2.0» является развитием «DualConnector Windows (SmartConnector Windows)», который способен обрабатывать запросы от терминала.

Функции кассового сервиса:

- Обработка входящих запросов от кассового ПО и терминала;
- Открытие и закрытие TCP-соединений с хостами;
- Управление приоритизацией соединений терминала;
- Трансляция сообщений между TCP-соединениями и интерфейсом RS232. Трансляция сообщений между RS232 и другими программными модулями;
- Инициация служебных и тестовых операций с ККМ с помощью меню кассира.

На Рисунок 1. Организация взаимодействия с DualConnector 2.0 представлена схема взаимодействия участников процесса обмена данными.

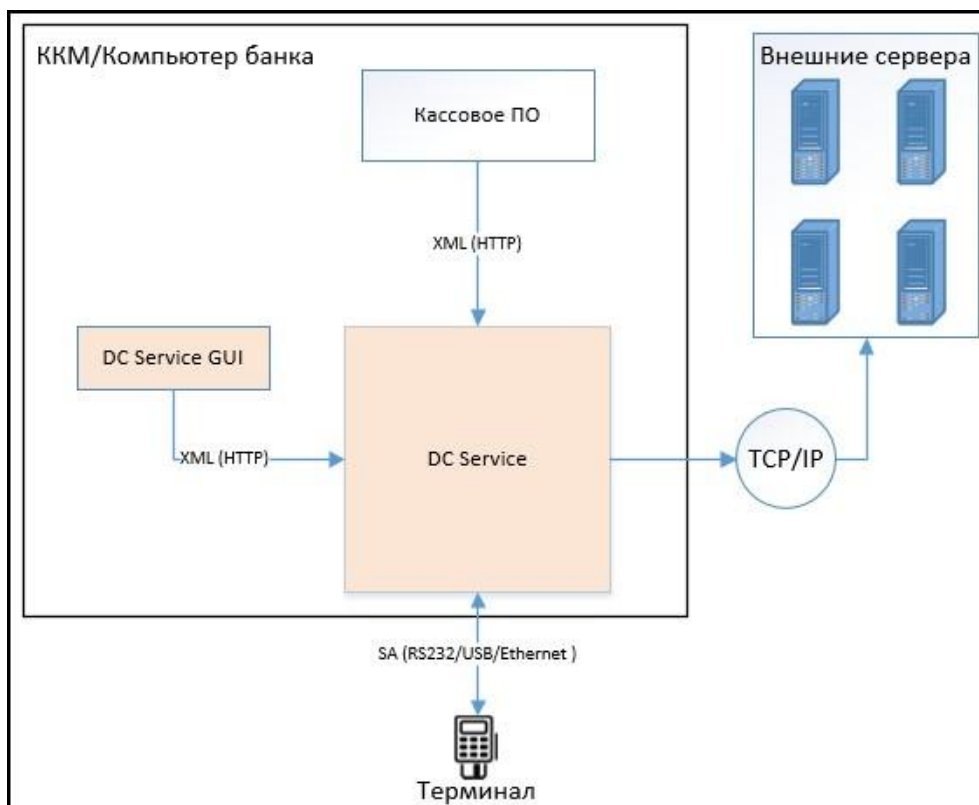


Рисунок 1. Организация взаимодействия с DualConnector 2.0

- **DC Service** — отвечает за следующую логику работы:
 - обрабатывает вызовы от терминала, т.е. непрерывно «слушает» COM-порт и обеспечивает терминал коммуникациями по его инициативе. При установке

- терминалом соединения с внешним хостом через «**DC Service**», сообщения транслируются на указанный адрес сети TCP/IP без модификации.
- управляет приоритетом соединений: в определенных случаях разрывает поток обмена терминала с хостом для доставки запроса кассы на терминал.

2.2. Системные требования

- Версия ОС: Windows XP/7/8/10;
- Установленный пакет «**.NET Framework 3.5**» и выше;
- Java7, 8, 11;
- Права доступа «**Администратор**».

2.3. Инсталляция в системе

Установка «**DualConnector 2.0**» в системе осуществляется только с помощью инсталляционного пакета. В рамках установки может происходить регистрация библиотек в среде COM, в GAC, регистрация службы «**Dual Connector Service**» и добавление необходимых переменных окружения.

В реестр Windows добавляется информация об установке «**DualConnector 2.0**».

Во время регистрации библиотек происходит «привязка» лицензии к диску директории установки. В директории создаётся файл «**reg_rpt.log**» с результатом привязки. При нормальном завершении в файле должна отобразиться строка «**License activated**».

Лицензия не влияет на работу ПО. Она необходима только для включения полного (VERBOSE) режима ведения лога в «**DC Proxy**».

Внимание!

При установке «**DualConnector 2.0**» выполняется проверка наличия установленных «**Java7**» («**Java8**») и «**.NET Framework 3.5** и выше».

2.4. Создание файла параметров «Connector.xml» в директории установки

Файла параметров «**Connector.xml**» можно создать несколькими способами.

2.4.1. Создание и настройка файла параметров при помощи «XML Generator»

Имеется возможность настроить файл конфигурации «**Connector.xml**» при помощи утилиты «**DC XML Generator**», которая входит в дистрибутив «**DualConnector 2.0**». Подробнее можно ознакомиться в руководстве пользователя «**DC XML Generator**».

Далее инсталляция происходит как в пункте 2.3.

2.4.2. Копирование заранее созданного файла параметров во время инсталляции.

Заранее создать файл конфигурации вручную или при помощи утилиты «**DC XML Generator**». Разместить файл в папке с инсталлятором ПО «**DualConnector 2.0**» («**Common Connectors Installer.exe**» или «**DualConnector 2.0 Installer.msi**»).

Параметр **COPYCONFIG** — при значении равным «1», копируется файла параметров «**Connector.xml**» из директории инсталлятора в папку, куда будет устанавливаться ПО «**DualConnector 2.0**». При любом ином значении копирование не происходит.

Запустить инсталлятор с параметром:

COPYCONFIG=1

Далее инсталляция происходит как в пункте 2.3.

2.4.3. Создание нового файла конфигурации при помощи ключей параметров во время инсталляции.

Ключи, с помощью которых можно создать файла параметров Connector.xml:

LOG_TYPE=SYSTEM — Уровень детализации логов. В пример, SYSTEM — системный

LOG_PATH=C:\temp_ — Путь к папке хранения логов.

LOG_CLEARTIME=18 — Время хранения логов в днях. Если параметр не задан, используется значение по умолчанию, 30 дней. Допустимое значение от 1 до 365.

CONFIRM_OPERATION=OFF — Настройка функционала автоматического подтверждения операции на стороне «DualConnector 2.0» или кассового ПО. Применяется только для версии «DualConnectorFull» и при включённой настройке в «UNIPOS Terminal». По умолчанию OFF.

CONTROL_SEND_DATA=OFF — Контроль отправки данных между кассой и терминалом. По умолчанию OFF.

CONTROL_SEND_DATA_TIMEOUT=34 — Время ожидания подтверждения отправки данных между кассой и POS-терминалом при использовании функционала «Контроль отправки данных между кассой и терминалом». Значение от 1 до 45 секунд. Значение по умолчанию — 5 секунд.

TRIPLEACK=ON — Настройка отправки 3 символов подтверждения (ACK) при получении пакета от терминала. По умолчанию ON.

HEX_STRING_FORMAT=ON — Выбор формата отправки ответа (response) в XML-файле. При отсутствии параметра или значения «ON», будет выполнена конвертация в шестнадцатеричный формат строки. При значении «OFF» будет проведена нормализация данных к XML-формату. Наличие не обязательно, по умолчанию выполняется конвертация данных

CONNECTION_TYPE=Ethernet — Выбирается требуемый тип подключения пин-пада к кассовому ПО.

CONNECTION_PORT=COM1 — Номер COM-порта, к которому подключен пин-пад если тип соединения COM/USB.

CONNECTION_BAUDRATE=115200 — Скорость обмена данными по COM-порту. Значение изменяется при использовании типа подключения по COM/USB. При подключении по USB необходимо оставлять настройку «по умолчанию» 115200.

IPADDRESS=127.0.0.1:27015 — IP-адрес и порт пин-пада при подключении пин-пада по Ethernet.

IPADDRESSGUI=127.0.0.1:6000 — IP-адрес сервера, на который адресуются запросы терминала.

CONNECT_TIMEOUT=78 — Прерывание установки соединения с сервером по истечению времени, в секундах. Значение по умолчанию — 30 секунд.

EXCHANGE_TIMEOUT=96 — Устанавливает максимальное время выполнения операции в секундах. Наличие необязательно, по умолчанию таймаут операции 45 секунд.

RECONNECTION_DELAY=7 — Задержка при повторном подключении/соединении к серверу после отключения в секундах. Наличие не обязательно, по умолчанию «0».

FREERESOURCE_AUTO=OFF — Настройка автоматического вызова FreeResources. Параметр не обязателен, если кассовое ПО самостоятельно вызывает данную функцию. По умолчанию ON.

Для того, чтобы создать файл конфигурации при помощи ключей параметров можно воспользоваться командной строкой или создать bat-файл со следующими командами:

```
«"Common Connectors Installer.exe" LOG_TYPE=SYSTEM LOG_PATH=C:\temp_ LOG_CLEARTIME=18  
CONFIRM_OPERATION=OFF CONTROL_SEND_DATA=OFF CONTROL_SEND_DATA_TIMEOUT=34  
TRIPLEACK=ON HEX_STRING_FORMAT=ON CONNECTION_TYPE=Ethernet CONNECTION_PORT=COM1  
CONNECTION_BAUDRATE=115200 IPADDRESS=127.0.0.1:27015 IPADDRESSGUI=127.0.0.1:6000  
CONNECT_TIMEOUT=78 EXCHANGE_TIMEOUT=96 RECONNECTION_DELAY=7 FREERESOURCE_AUTO=OFF -I  
log.txt»
```

«Common Connectors Installer.exe» (или «DualConnector 2.0 Installer.msi») — запуск инсталляционного файла.

«-I log.txt» — команда записи логов создания файла конфигурации. Файл «log.txt» будет находится в той же папке откуда производился запуск инсталляционного файла

Далее инсталляция происходит как в пункте 2.3.

3. Прямая интеграция с DC Service по HTTP

Для интеграции по HTTP необходимо сформулировать запрос методом POST.

Пример:

```
POST / HTTP/1.1
Host: 10.35.91.20:9015
User-Agent: curl/7.53.1
Accept: */*
Accept-Charset: windows-1251
Content-Type: text/xml
Content-Length: 305
```

«**HTTP-запрос**» может включать в себя некоторые поля, перечисленных в таблице «Таблица 1. Перечень свойств/ полей SAPacket».

Для корректной обработки данных необходимо указываться кодировку, в которой передаются данные в HTTP запросе. Описание кодировки указывается в заголовке «**Content-Type**», например, «**Content-Type: text/xml; charset=windows-1251**». Данные будут интерпретированы в указанной в заголовке «**charset**» кодировке. Если кодировка не указана, то данные будут интерпретированы в кодировке по умолчанию — «**windows-1251**».

При указании в заголовке Accept-Charset кодировки (например, «**windows-1251**»), будет получен файл ответа в указанной кодировке или в кодировке по умолчанию («**windows-1251**»), если заголовок не указан или имеет неподдерживаемую кодировку.

В HTTP запросе может быть использован один из двух методов «**POST**» или «**GET**», при использовании иных методов возможно появление ошибки «**3**».

Если используется метод «**GET**», то в ответе возможно получить статус работы DC с устройством. DC вернет код ответа HTTP «**200**», если в данный момент DC обменивается данными с терминалом или код ответа HTTP «**404**», если DC ожидает команды.

В ответ на полученный запрос от кассы, при ошибке, формируется и передается файл «**XML**» или «**HTML**», в зависимости от значения, указанного в «**Accept**» (в запросе).

При получении запроса от кассы, сервер анализирует HTTP заголовок «**Accept**».

Если заголовок «**Accept**» имеет значение «**text/xml**», то при возникновении ошибки, передается ответ в виде «**XML**» документа. Код ответа HTTP в этом случае «**200**».

Пример:

```
<?xml version="1.0" encoding="windows-1251" standalone="no"?>
<response>
  <errorcode>4</errorcode>
  <errordescription>REQUEST_ERROR</errordescription>
</response>
```

Если заголовок «**Accept**» имеет значение отличное от «**text/xml**», то при возникновении ошибки, передается ответ в виде «**HTML**» документа с указанием результата и описание ошибки. Код ответа HTTP в этом случае «**400**», «**404**».

Возможные значения «**errorcode**»:

- **0** — DC не смог передать ответ от терминала. Возможная причина, внутренняя ошибка DC;
- **1** — истекло время исполнения операции. Устанавливает полем «**timeout**» в запросе;
- **3** — общая ошибка, ошибка параметров DC, не поддерживаемый метод HTTP, неизвестная ошибка, ошибка во время работы DC;
- **4** — ошибка поля или ошибка запроса, как между кассой и DC, так и между DC и терминалом;

- **13** — ошибка обмена данными между DC и терминалом. Возможная причина: отсутствие устройства или ответ не по протоколу SA;
- **15** — обмен данными прерван, например, выключение DC, мониторинг подключения устройства или отменен другим обменом;
- **16** — устройство занято, например, занято другим обменом;
- **17** — сессия обмена с терминалом завершена. Возможная причина: от Кассы пришел запрос (подтверждение) с идентификатором сессии, которая уже закрыта (от терминала пришел EOT или DC отправил терминалу EOT).

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <field id="00">100</field>
  <field id="04">643</field>
  <field id="21">20150729121815</field>
  <field id="25">1</field>
  <field id="26"></field>
  <field id="27">00199991</field>
  <field id="90">0102030405060708090a0b0c0d0e0f10</field>
  <timeout>60</timeout>
  <sessionID>1934425084</sessionID>
</request>
```

Запрос может оформляться в двух вариантах: «**XML**» и «**XSD**»

Запрос оформленный в формате «**XML**» и состоит из заглавного тега XML и контейнера с перечнем полей. В случае запроса, перечень полей перечисляется в контейнере «**<request>**», в случае ответа кассе, аналогичный перечень полей перечисляется в контейнере «**<response>**». Каждое из полей описывается тэгом «**Field**», номер поля указывается атрибутом «**id**», атрибут «**hex**» указывает на формат данных HEX строки, которая требует переконвертации в бинарные.

Так же в запросе передается параметр «**sessionID**» — уникальный идентификатор для каждой сессии, который присваивается конкретной операции. Формат значения данного параметра — произвольный. DC сохраняет все идентификаторы сессий и статусы обмена данными с терминалом до перезагрузки сервиса.

Запрос оформленный в формате «**XSD**»-схемы строго описывает структуру сообщения для общения с «DC Service» (см «Пример запроса request с помощью «XSD»-схемы»)

Формат запроса («**XML**» или «**XSD**») предусматривает возможность указать адрес терминала, подключенного по «**TCP/IP**» или «**COM/USB**». Данная возможность позволит маршрутизировать запросы на тот или иной терминал, когда к кассовой сети подключено большое количество терминалов.

Для терминалов, подключенных по COM/USB:

- **ncom** — номер RS232 порта;
- **baudrate** — скорость порта (опциональный параметр).

Для терминалов, подключенных по TCP/IP:

- **ipaddr** — IP-адрес и порт подключения терминала, формат: 192.168.0.2:9000;
- **timeout** — предоставляемое время на выполнение операции в секундах. Данный параметр ограничивает время выполнению любых операций, чтобы предотвратить вхождение в бесконечный цикл в случае нештатной ситуации. Рекомендуемое значение 180 (3 минуты). Если таймаут не указан, то значение равно 0, бесконечное ожидание.

Параметр обрабатывается, только для команд от кассы.

Примечание. Параметр **timeout** — рассчитывается из принципа разумной необходимости и должен немного превосходить суммарное расчётное время на все операции с картой. Например: ввод карты клиента, ввод пинкода клиентом, обмен данными.

Пример запроса request с помощью «XML»-схемы:

```
<request>
  <field id="00">100</field>
  <field id="04">643</field>
  <field id="21">20150729121815</field>
  <field id="25">1</field>
  <field id="26"> </field>
  <field id="27">001999991</field>
  <field id="90">0102030405060708090a0b0c0d0e0f10</field>
  <ipaddr>192.168.0.2:9000</ipaddr>
  <timeout>180</timeout>
  <sessionID>1934425084</sessionID>
```

Или

```
  <ncom>9</ncom>
  <baudrate>115200</baudrate>
</request>
```

Пример запроса request с помощью «XSD»-схемы:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="field">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="id" type="xs:integer" use="required"/>
          <xs:attribute name="hex" type="xs:boolean"
use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="ipaddr" type="xs:string"/>
  <xs:element name="timeout" type="xs:integer"/>
  <xs:element name="ncom" type="xs:string"/>
  <xs:element name="baudrate" type="xs:integer"/>
  <xs:element name="request">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="field" maxOccurs="unbounded"/>
        <xs:sequence minOccurs="0">
          <xs:element ref="ipaddr"/>
        </xs:sequence>
        <xs:sequence minOccurs="0">
          <xs:element ref="ncom"/>
          <xs:element ref="baudrate"/>
        </xs:sequence>
        <xs:element ref="timeout" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

</xs:schema>

4. Настройка параметров «DualConnector 2.0»

4.1. Основные параметры

Основной файл параметров находится в директории «C:\Program Files (x86)\INPAS\DualConnector 2.0\Service\config\» (при установке по умолчанию) и называется «Connector.xml».

Содержит данные в следующей структуре xml:

```
<ROOT>
  <SERVERPORT>9015</SERVERPORT>
  <LOG_TYPE>DEBUG</TYPE>
  <LOG_PATH>/var/logs</PATH>
  <LOG_CLEARTIME>30</LOG_CLEARTIME>
  <CONFIRM_OPERATION>ON</CONFIRM_OPERATION>
  <TRIPLEACK>ON</TRIPLEACK>
  <DEVICES_TYPE>TERMINAL</DEVICES_TYPE>
  <CONNECTION_TYPE>COM</CONNECTION_TYPE>
  <CONNECTION_PORT>COM6</CONNECTION_PORT>
  <CONNECTION_BAUDRATE>115200</CONNECTION_BAUDRATE>
  <CONTROL_SEND_DATA>ON</CONTROL_SEND_DATA>
  <CONTROL_SEND_DATA_TIMEOUT>4</CONTROL_SEND_DATA_TIMEOUT>
  <IPADDRESS>10.35.1.40:1006</IPADDRESS>
  <IPADDRESSGUI>127.0.0.1:6000</IPADDRESSGUI>
  <WAITACK>6</WAITACK>
  <WAITPACKET>45</WAITPACKET>
  <CONNECT_TIMEOUT>20</CONNECT_TIMEOUT>
  <EXCHANGE_TIMEOUT>180</EXCHANGE_TIMEOUT>
  <RECONNECTION_DELAY>6</RECONNECTION_DELAY>
  <HEX_STRING_FORMAT>ON</HEX_STRING_FORMAT>
  <CONFIG_WATCHER>ON</CONFIG_WATCHER>
</ROOT>
```

1. **ROOT** — корневая область. Наличие обязательно.
2. **SERVERPORT** — порт, по которому доступны запросы к сервису. Наличие обязательно.
3. **LOG_TYPE** — тип детализации информации в файле лога. Допустимые значения в порядке увеличения выводимой информации: «OFF», «SYSTEM», «ADVANCED», «DEBUG», «VERBOSE». Наличие необязательно, по умолчанию «ADVANCED».
4. **LOG_PATH** — Путь сохранения файлов лога. Если в параметре не указан путь, куда сохранять файлы логов или вообще отсутствует данный параметр, то файлы логов сохраняются в директорию по умолчанию «/var/log/dualconnector».
5. **LOG_CLEARTIME** — Время хранения логов (в днях). Диапазон возможных значений от 1 до 365 дней. Если параметр не задан, используется значение по умолчанию 30 дней.
6. **CONFIRM_OPERATION** — Секция настройки включения автоматического подтверждения операции на стороне внешнего устройства. Наличие не обязательно. По умолчанию, выключено.
7. **TRIPLEACK** — секция настройки отправки 3 символов подтверждения (ACK) по завершении операции на терминал. Наличие не обязательно. По умолчанию выключено.
8. **DEVICES_TYPE** — Тип терминала. Допустимые значения «TERMINAL», «PINPAD». Наличие необязательно. По умолчанию «TERMINAL». Отличие типов используется для определения наличия принтера.

9. **CONNECTION_TYPE** — Тип подключения к терминалу. Допустимые значения «**COM**», «**IP**». Наличие обязательно.
10. **CONNECTION_PORT** — Номер COM-порта. Наличие обязательно при соединении по COM.
11. **CONNECTION_BAUDRATE** — Скорость обмена. Наличие необязательно. По умолчанию 115200.
12. **CONTROL_SEND_DATA** — Контроль отправки данных между кассой и терминалом. По умолчанию «**OFF**».
13. **CONTROL_SEND_DATA_TIMEOUT** — Время ожидания подтверждения отправки данных между кассой и POS-терминалом при использовании функционала «Контроль отправки данных между кассой и терминалом». Значение от 1 до 45 секунд. Значение по умолчанию — 5 секунд.
14. **IPADDRESS** — IP адрес терминала. Наличие обязательно при соединении по IP.
15. **IPADDRESSGUI** — Если касса обрабатывает команды, то указывает IP-адрес и порт кассы, если обработка команд выполняется «**DC Service GUI**», то нужно оставить значение по умолчанию.
16. **WAITACK** — Время ожидания сигнала подтверждения получения пакета в секундах. Наличие необязательно, по умолчанию 5.
17. **WAITPACKET** — Время ожидания ответного пакета в секундах (или миллисекундах при значениях выше 300). Наличие необязательно, по умолчанию 45.
18. **CONNECT_TIMEOUT** — Механизм прерывания установки соединения по истечению времени. Указывается время ожидания соединения с сервером в секундах. Значение по умолчанию — 30 секунд.
19. **EXCHANGE_TIMEOUT** — Устанавливает максимальное время выполнения операции в секундах.
20. **RECONNECTION_DELAY** — Устанавливает задержку в секундах на повторное подключение/соединение к серверу после отключения в секундах.
21. **HEX_STRING_FORMAT** — Указывает формат поле в ответе (response) в XML-файле. При необходимости переконвертирует поля в данные для передачи в XML документе и добавлен атрибут «**hex**». Если данный параметр не задан или имеет значение «**ON**», то будет проведена конвертация, при необходимости. Если данный установлен и имеет значение «**OFF**» или любое другое значение, отличное от «**ON**», то будет проведена нормализация данных к XML формату. Если данные пришли с атрибутом «**hex**», то данный будут конвертированы из HEX строки в бинарные данные вне зависимости от параметра.
22. **CONFIG_WATCHER** — Отслеживание изменений в файле настроек. Если данный параметр не задан или имеет значение «**ON**», то после изменения файла конфигурации и его прочтения, перезапускается HTTP-сервер с новыми настройками. Если параметр имеет значение «**OFF**», то изменения в файле конфигурации игнорируются.

4.2. Параметры по TerminalID

Файл параметров конкретного терминала находится в директории «**C:\Program Files (x86)\INPAS\DualConnector 2.0\Service\config\[TerminalID]**» и называется «**Connector.xml**».

Файл содержит тот же набор параметров, что и основной файл, описанный выше, но со значениями параметров под конкретный терминал (файл создается при настройке параметров терминала на вкладке «Настройки службы»). Настройки по **TerminalID** используются только для определения типа и порта соединения с терминалом.

Для отдельного **TerminalID** используются 3 основных параметра: <CONNECTION_TYPE>, <CONNECTION_PORT>, <IPADDRESS>. Все остальные параметры берутся из головного файла «**Connector.xml**».

```
<ROOT>
  <SERVERPORT>9015</SERVERPORT>
  <LOG_TYPE>ADVANCED</TYPE>
  <LOG_PATH>/var/logs</PATH>
  <LOG_CLEARTIME>30</LOG_CLEARTIME>
  <CONFIRM_OPERATION>ON</CONFIRM_OPERATION>
  <TRIPLEACK>ON</TRIPLEACK>
```

```
<DEVICES_TYPE>TERMINAL</DEVICES_TYPE>
<CONNECTION_TYPE>COM</CONNECTION_TYPE>
<CONNECTION_PORT>COM3</CONNECTION_PORT>
<CONNECTION_BAUDRATE>115200</CONNECTION_BAUDRATE>
<CONTROL_SEND_DATA>ON</CONTROL_SEND_DATA>
<CONTROL_SEND_DATA_TIMEOUT>4</CONTROL_SEND_DATA_TIMEOUT>
<IPADDRESS>10.35.1.40:1006</IPADDRESS>
<IPADDRESSGUI>127.0.0.1:6000</IPADDRESSGUI>
<WAITACK>6</WAITACK>
<WAITPACKET>45</WAITPACKET>
<CONNECT_TIMEOUT>20</CONNECT_TIMEOUT>
<EXCHANGE_TIMEOUT>180</EXCHANGE_TIMEOUT>
<RECONNECTION_DELAY>6</RECONNECTION_DELAY>
<HEX_STRING_FORMAT>ON</HEX_STRING_FORMAT>
</ROOT>
```

4.3. Настройки параметров DC Service GUI

ПО DC Service GUI.xml — модуль отображения терминальных диалоговых окон на ККИ.

В файле «DC Service GUI.xml» хранятся настройки конфигурации терминального диалогового окна, для отображения запросов терминала кассиру.

```
<root>
  <listen>127.0.0.1:6000</listen>
  <codepage>1251</codepage>
  <readtimeoutms>30</readtimeoutms>
  <closewindowkey>65</closewindowkey>
</root>
```

1. **root** — корневая область. Наличие обязательно.
2. **listen** — IP-адрес сервера, на который адресуются запросы терминала. Описание протокола смотрите в соответствующем разделе данного документа. В этом поле должен стоять такой же IP-адрес, как и в файле конфигурации «Connector.xml» в поле **IPADDRESSGUI**.
3. **codepage** — кодировка символов терминального сообщения. Числовое значение по умолчанию — 1251, что соответствует кодировке Windows-1251.
4. **readtimeoutms** — время считывания пакета запроса в миллисекундах.
5. **closewindowkey** — выбор клавиши на клавиатуре для закрытия окна терминального сообщения. В примере код «65» соответствует клавише клавиатуры «ESC».

Примечание. Код нужной клавиши клавиатуры выбирается в таблице кодов клавиш клавиатуры.

5. Примеры работы с «DC Service»

5.1. C++

```
#pragma comment(lib, "winhttp.lib")
int main()
{
    DWORD dwSize = 0;
    DWORD dwDownloaded = 0;
    LPSTR pszOutBuffer;
    BOOL bResults = FALSE;
    HINTERNET hSession = NULL, hConnect = NULL, hRequest = NULL;
    std::ifstream ifs("TextFile1.xml");
    std::string xmlString((std::istreambuf_iterator<char>(ifs)), (std::istreambuf_iterator<char>()));
    // Use WinHttpOpen to obtain a session handle.
    hSession = WinHttpOpen(L"WinHTTP Example/1.0", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BYPASS, 0);
    // Specify an HTTP server.
    if (hSession)
        hConnect = WinHttpConnect(hSession, L"127.0.0.1", 9015, 0);
    // Create an HTTP request handle.
    if (hConnect)
        hRequest = WinHttpOpenRequest(hConnect, L"POST", NULL, NULL, WINHTTP_NO_REFERER,
WINHTTP_DEFAULT_ACCEPT_TYPES, 0);
    LPCWSTR additionalHeaders = L"Content-Type: text/xml;charset=windows-1251\r\n";
    DWORD headersLength = -1;
    LPSTR data = const_cast<char*>(xmlString.c_str());
    DWORD dataLen = strlen(data);
    // Send a request.
    if (hRequest)
        bResults = WinHttpSendRequest(hRequest, additionalHeaders, headersLength, data,
dataLen, dataLen, 0);
    // End the request.
    if (bResults)
        bResults = WinHttpReceiveResponse(hRequest, NULL);
    // Keep checking for data until there is nothing left.
    if (bResults)
    {
        do
        {
            // Check for available data.
            dwSize = 0;
            if (!WinHttpQueryDataAvailable(hRequest, &dwSize))
                printf("Error %u in WinHttpQueryDataAvailable.\n", GetLastError());
            // Allocate space for the buffer.
            pszOutBuffer = new char[dwSize + 1];
            if (!pszOutBuffer)
            {
                printf("Out of memory\n");
                dwSize = 0;
            }
            else
            {
                // Read the data.
                ZeroMemory(pszOutBuffer, dwSize + 1);
                if (!WinHttpReadData(hRequest, (LPVOID)pszOutBuffer, dwSize,
&dwDownloaded))
```



```

        printf("Error %u in WinHttpReadData.\n", GetLastError());
    else
        printf("%s", pszOutBuffer);
    // Free the memory allocated to the buffer.
    delete[] pszOutBuffer;
    }
    } while (dwSize > 0);
}
// Report any errors.
if (!bResults) printf("Error %d has occurred.\n", GetLastError());
// Close any open handles.
if (hRequest) WinHttpCloseHandle(hRequest);
if (hConnect) WinHttpCloseHandle(hConnect);
if (hSession) WinHttpCloseHandle(hSession);
return 0;
}

```

5.2. C#

```

long result = -1;
string xmlString = File.ReadAllText(@"TextFile1.xml", Encoding.GetEncoding("windows-1251"));
using (WebClient client = new WebClient())
{
    string responseString = null;
    try
    {
        string serviceIpAdress = "127.0.0.1:9015";
        string head = "http://" + serviceIpAdress;
        client.Encoding = Encoding.GetEncoding("windows-1251");
        client.Headers.Add("Content-Type: text/xml;charset=" + client.Encoding.WebName);
        responseString = client.UploadString(head, xmlString);
        if (responseString != null)
        {
            result = responseString.Length;
        }
        else { result = -100; }
    }
    catch (WebException e)
    {
        Console.WriteLine(e.Message);
        result = -200;
    }
    if (result > 0)
    {
        Console.WriteLine(responseString);
    }
    else { Console.WriteLine(result); }
}

```

5.3. Java

```

try {
    final URL url = new URL("http://127.0.0.1:9015");
    final HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setDoOutput(true);
    con.setRequestMethod("POST");
    con.setRequestProperty("Content-Type", "text/xml;charset=windows-1251");
    String absolutePath = File.separator + "TextFile1.xml";
}

```



```
byte[] outputData = Files.readAllBytes(Paths.get(absolutePath));
try (DataOutputStream outputStream = new DataOutputStream(con.getOutputStream())) {
    outputStream.write(outputData);
    outputStream.flush();
    String inputData;
    try (final BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream(), Charset.forName("windows-1251")))) {
        String inputLine;
        StringBuilder content = new StringBuilder();
        while ((inputLine = in.readLine()) != null) {
            content.append(inputLine);
        }
        inputData = content.toString();
    } catch (final Exception ex) {
        ex.printStackTrace();
        inputData = "";
    }
    System.out.println(inputData);
} catch (IOException e) {
    e.printStackTrace();
}
con.disconnect();
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

6. Особенности использования DC Proxy

DC Proxy — реализует открытие API в виде вызова динамической библиотеки для сохранения интеграции с существующими стыковками с кассовым ПО. Предназначен для реализации промежуточного интерфейса между библиотекой «**DualConnector**» и сервисом «**DC Service**». Позволяет пользователям перейти на работу с «**DC Service**» не меняя свое внутреннее ПО. При этом «**DC Proxy**» не содержит логики, а является только прослойкой для преобразования запросов кассы в интерфейс «**DC Service**».

7. Взаимодействие с оператором

«DualConnector 2.0» имеет возможность транслирования запросов терминала для отображения сообщений кассиру.

Взаимодействие может быть организовано основным или альтернативным способом.

Основным является использование ПО «DC Service GUI». Для его использования необходимо убедиться в том, что модуль установлен (выбрать галочку при установке) и указать настройки соединения, которые находятся в файлах конфигурации «Connector.xml» и «DC Service GUI.xml». Подробная информация и пример файла конфигурации представлены в разделе «[Настройка параметров Dual Connector 2.0](#)». Данные в «DC Service GUI» передаются посредством стека протоколов TCP/IP в текстовом формате XML.

Альтернативный способ вывода окон необходим, когда производитель кассового ПО решает использовать свою реализацию диалоговых окон. При этом есть один способ решения данной задачи:

1. Реализовать сервер вывода окон — аналог «DC PosGUI», при этом настройки «DualConnector» остаются прежними, но установку «DC PosGUI» необходимо отменить во избежание конфликтных ситуаций.

Информационное сообщение

Предназначено для информирования кассира о событии. Ответ кассира не требуется.

Формат запроса:

```
<request>
  <type>1</type>
  <data>4^^Заголовок^Сообщение</data>
  <timeout>10</timeout>
</request>
```

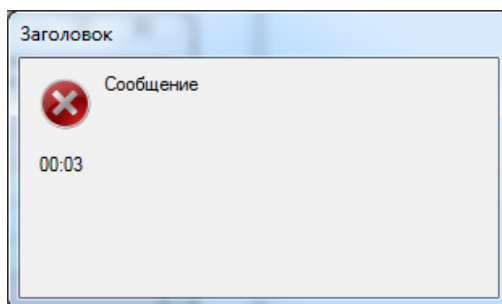
Формат ответа:

```
<response>
  <type>1</type>
  <data>0</data>
</response>
```

Где:

- type — 1 — информационное сообщение;
- data (в запросе) — данные для отображения в соответствии с форматом данных для отображения (см. ниже);
- timeout — время отображения окна в секундах;
- data (в ответе) — ответ кассира (в данном случае 0, кассир ничего не нажимал и не должен);

Пример экранной формы:



Т.к. данное сообщение не требует ответа кассира, ответ должен поступить без задержки.

Сообщение подтверждения

Предназначено для запроса у кассира определённого ответа.

Формат запроса:

```
<request>
  <type>2</type>
  <data>3^5^ Заголовок^Сообщение </data>
  <timeout>30</timeout>
</request>
```

Формат ответа:

```
<response>
  <type>2</type>
  <data>32</data>
</response>
```

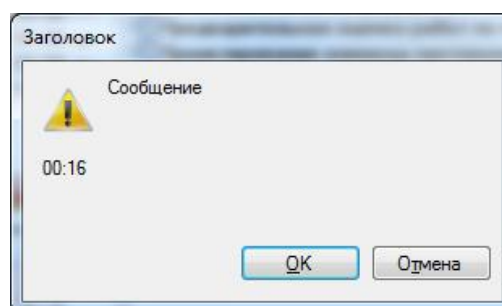
Где:

- type — 2 — сообщение подтверждения;
- data (в запросе) — данные для отображения в соответствии с форматом данных для отображения (см. ниже);
- timeout — время отображения окна в секундах;
- data (в ответе) — ответ кассира.

Возможные значения поля data в ответе кассира:

- 0 — кассир ничего не нажимал
- 1 — нажал ОК;
- 2 — нажал ответ ДА (Yes);
- 4 — нажал ответ ОТМЕНА (Cancel);
- 8 — нажал ответ НЕТ (No);
- 16 — вышло время диалога (timeout);
- 32 — кассир нажал Escape(закрыл форму без выбора варианта ответа);
- 64 — переданы ошибочные параметры, диалог не отображён.

Пример экранной формы



Сообщение выбора из списка

Предлагает кассиру выбрать вариант из списка.

Формат запроса:

```
<request>
  <type>3</type>
  <data>2^1^ Заголовок^Сообщение </data>
  <adata>RUB;USD;EUR</adata>
  <timeout>30</timeout>
</request>
```

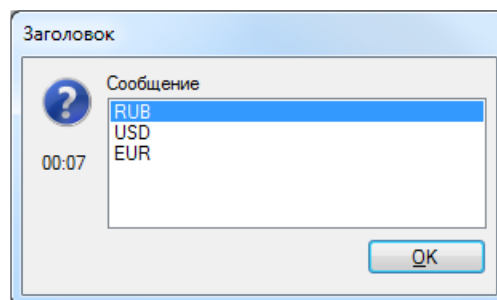
Формат ответа:

```
<response>
  <type>3</type>
  <data>1</data>
  <adata>4</adata>
</response>
```

Где:

- type — 3 — сообщение выбора;
- data (в запросе) — данные для отображения в соответствии с форматом данных для отображения (см. ниже);
- adata (в запросе) — (additional) список вариантов, разделённых символом '\n' или ';' ;
- timeout — время отображения окна в секундах;
- data (в ответе) — ответ кассира, варианты описаны ранее.
- adata (в ответе) — вариант выбора кассира в представлении N=2m.
 - где m — индекс строки, начиная с 0. Т.е. первая строка будет 1, вторая — 2, третья — 4, четвёртая — 8 и т.д.

Пример экранной формы



Сообщение ввода данных

Запрашивает у кассира символьные данные.

Формат запроса

```
<request>
  <type>4</type>
  <data>1^1^ Заголовок^Сообщение </data>
  <adata>000999</adata>
  <timeout>30</timeout>
</request>
```

Формат ответа:

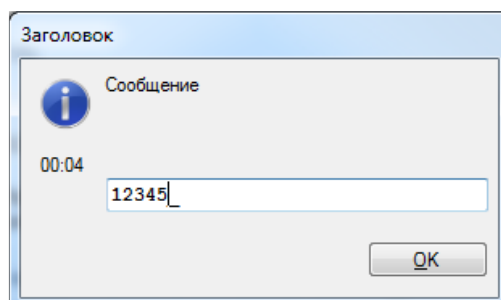
```
<response>
  <type>4</type>
  <data>1</data>
  <adata>12345</adata>
</response>
```

Где:

- type — 4 — сообщение ввода данных;
- data (в запросе) — данные для отображения в соответствии с форматом данных для отображения (см. ниже);
- adata (в запросе) — (additional) маска ввода.
- timeout — время отображения окна в секундах;
- data (в ответе) — ответ кассира, варианты описаны ранее.

- adata (в ответе) — введенные данные.

Пример экранной формы



Сообщение печати данных

Касса распечатывает данные на принтере.

Формат запроса:

```
<request>
  <type>5</type>
  <data>ДАННЫЕ ДЛЯ ПЕЧАТИ</data>
</request>
```

Формат ответа:

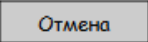

```
<response>
  <type>5</type>
  <data>0</data>
</response>
```

Где:

- type — 5 — сообщение печати данных;
- data (в запросе) — данные для печати. Строки заранее отформатированы, разделены символом '\n';
- data (в ответе) — ответ. 0 — успешная печать, 64 — произошла ошибка.

Формат данных для отображения

Элемент подполя	Описание	Атрибут	Тип поля
Уровень сообщения	<p>Определяет стиль иконки окна, выводимого на ККМ. Может принимать значения:</p> <ul style="list-style-type: none"> 1 MB_INFORMATION — информирование кассира 2 MB_ICONQUESTION — запрос кассиру, требующий ответа 3 MB_ICONEXCLAMATION, MB_ICONWARNING — сообщение об ошибке или предупреждение 4 MB_ICONSTOP критическая ошибка 	O	n1
^	Разделитель между элементами данных внутри поля.	M	
Элемент управления	<p>Определяет элементы управления (кнопки), которые должны быть отрисованы в окне, выводимом на ККМ. Поле представляет собой битовую маску:</p> <p>0x01 — Ok MB_OK</p> <p>0x02 — Yes MB_YES</p>	O	n..3

	0x04 — Cancel  MB_CANCEL		
	0x08 — No  MB_NO		
^	Разделитель между элементами данных внутри поля.	M	
Заголовок сообщения	Заголовок окна, выводимого на ККМ	O	an..40
^	Разделитель между элементами данных внутри поля.	M	
Сообщение кассиру	Строка (или набор строк, разделенных символом «\n» или «;»), содержащая текст информационного сообщения для кассира	M	an..950

M — mandatory: обязательный элемент;

O — optional: опциональный элемент, может отсутствовать.

Пример:

Все элементы присутствуют	1^1^ОПЛАТА ТОВАРА^ПОДТВЕРДИТЕ ДАННЫЕ\nНАЖМИТЕ ОК
Заголовок и элементы управления (кнопки) отсутствуют	1^^^УСТАНОВКА\nСОЕДИНЕНИЯ
Уровень сообщения отсутствует	^2^ОПЛАТА^ВВЕДИТЕ\nНОМЕР ЧЕКА

8. Приложение

8.1. Порядок использования API

Для работы с «DualConnector» рекомендуется следующая последовательность действий:

1. Создать объект «Request» — экземпляр класса «ISAPacket» — для передачи данных терминалу для проведения транзакции;
2. Создать объект «Response» — экземпляр класса «ISAPacket» — для получения результатов транзакции от терминала;
3. Создать объект «DCLink» — экземпляр класса «DualConnectorInterface» — для организации обмена данными с терминалом;
4. Проинициализировать «DCLink», используя для этого метод «InitResources». Инициализацию необходимо выполнять перед каждым вызовом метода «Exchange»;
5. При необходимости настроить параметры связи с терминалом, используя для этого метод «SetChannelTerminalParam» объекта «DCLink». Если данный метод не вызывать, настройки будут взяты из файла параметров;
6. Подготовить данные для проведения транзакции, заполнив соответствующие поля объекта «Request»;
7. Послать запрос на проведение транзакции, вызвав метод «Exchange» объекта «DCLink»;
8. Обработать результат проведения транзакции, прочитав данные из объекта «Response»;
9. Освободить ресурсы, выделенные под объекты «Request» и «Response», используя для этого метод «Release» данных объектов;
10. Освободить используемые ресурсы, используя для этого метод «FreeResources» объекта «DCLink».

Внимание!

При работе с «DualConnector» требуется учитывать, что при отправленном запросе на выполнение операции требуется ожидать его выполнения и только после этого отправлять повторный запрос на выполнение операции.

8.1.1. Описание API

Объект «DCLink»

Метод «InitResources»:

- Назначение — Инициализация ресурсов
- Формат вызова — InitResources ()
- Параметры — нет
- Возвращаемое значение — int ErrorCode

Метод «Exchange»:

- Назначение — Обмен информацией с терминалом
- Формат вызова — Exchange (Request, Response, Timeout)
- Параметры:
 - **Request** — объект с исходными данными транзакции.
 - **Response** — объект, который будет заполнен ответными данными транзакции.
 - **Timeout** — Предоставляемое время на выполнение операции. Данный таймаут защищает вызываемое приложение от «вечного» зависания в случае нештатной ситуации. Должен рассчитываться из принципа разумной необходимости и немного превосходить суммарное расчётное время на ввод карты клиента, ввод пинкода клиентом, обмен данными. Рекомендуемое значение 180 (3 минуты).
- Возвращаемое значение — int ErrorCode

Метод «FreeResources»:

- Назначение — Освобождение ресурсов
- Формат вызова — FreeResources ()
- Параметры — нет
- Возвращаемое значение — нет

Метод «SetChannelTerminalParam»:

- Назначение — динамическая установка параметров связи с терминалом (значения файла параметров игнорируются)
- Формат вызова — SetChannelTerminalParam (nCOM, BaudRate, ByteSize, Parity, StopBits, FlowCtrl)
- Параметры:
 - **nCom** — номер RS232 порта
 - **BaudRate** — скорость порта
 - **ByteSize** — размер байта (игнорируется, всегда 8)
 - **Parity** — чётность (игнорируется, всегда нет)
 - **StopBits** — количество стоп-битов
 - **FlowCtrl** — контроль передачи (Игнорируется, всегда OFF)
- Возвращаемое значение — int ErrorCode

Метод «Cancel»:

- Назначение — Прерывание выполняемой на терминале операции
- Формат вызова — Cancel ()
- Параметры — нет
- Возвращаемое значение — int ErrorCode

Метод «SetField»:

- Назначение — устанавливает для любого поля, не указанного в списке объекта «SAPacket», строковое значение, если это поле поддерживает строковый формат
- Формат вызова — bool SetField(Int32 fieldNum, string value)
- Параметры
 - **fieldNum** — номер поля
 - **value** — значение поля
- Возвращаемое значение — true

Метод «SetFieldInt»:

- Назначение — устанавливает для любого поля, не указанного в списке объекта «SAPacket», целочисленное значение, если это поле поддерживает целочисленный формат
- Формат вызова — bool SetFieldInt(Int32 fieldNum, Int32 value)
- Параметры
 - **fieldNum** — номер поля
 - **value** — значение поля
- Возвращаемое значение — true

Метод «GetField»:

- Назначение — позволяет получить строковое значение для любого поля не указанного в списке объекта «SAPacket»
- Формат вызова — string GetField(Int32 fieldNum)
- Параметры
 - **fieldNum** — номер поля
- Возвращаемое значение — string (значение поля) или null, если поле не задано

Метод «GetFieldInt»:

- Назначение — позволяет получить целочисленное значение для любого поля не указанного в списке объекта «SAPacket»
- Формат вызова — `Int32 GetFieldInt(Int32 fieldNum)`
- Параметры
 - **fieldNum** — номер поля
- Возвращаемое значение — `int` (значение поля) или 0, если поле не задано

Свойство «ErrorCode»:

- Значение — результат операции
- Возможные значения:
 - **OK = 0** — ошибок нет;
 - **TIMEOUT = 1** — истёк таймаут операции;
 - **LOG_ERROR = 2** — ошибка создания LOG файла;
 - **SYSTEM_ERROR = 3** — общая ошибка;
 - **REQUEST_ERROR = 4** — ошибка данных запроса;
 - **CONFIG_NOT_FOUND = 6** — не найден файл конфигурации;
 - **CONFIG_ERROR_FORMAT = 7** — ошибка формата файла конфигурации;
 - **CONFIG_ERROR_LOG = 8** — ошибка параметров логирования;
 - **CONFIG_ERROR_DEVICES = 9** — ошибка в параметрах терминала;
 - **CONFIG_ERROR_DUBLCOMPORTS = 10** — ошибка настройки устройства на COM порт;
 - **CONFIG_ERROR_OUTPUT = 11** — ошибка в выходных параметрах;
 - **PRINT_ERROR = 12** — ошибка при передаче образа чека;
 - **ERROR_CONNECT = 13** — ошибка установки связи с устройством;
 - **CONFIG_ERROR_GUI = 14** — ошибка в параметрах настройки интерфейса взаимодействия с пользователем;
 - **CANCEL_OPERATION = 15** — отмена операции;
 - **16** — устройство занято.

Свойство «ErrorDescription»:

- Значение — текстовое описание значения ErrorCode

Событие «OnShowWindow»

- Вызывается при необходимости отобразить терминальные диалоговые или информационные окна для пользователя. Если подписка на данное событие не осуществлена, то вывод окон производится через DC PosGUI. (см раздел *«Взаимодействие с оператором»*).

Status (статус проведения транзакции)

Описание:

Результат выполнения авторизационной транзакции должен трактоваться однозначно по одному признаку — статусу проведения транзакции. В случае ошибки в свойстве **«TextResponse»** (дополнительные данные ответа) может присутствовать в текстовом виде описание причины ошибки.

Используются следующие статусы проведения транзакций.

Значение	Описание
0	Неопределенный статус
1	Одобрено
16	Отказано
17	Выполнено в OFFLINE
34	Нет соединения
53	Операция прервана

ReceiptData (данные для печати на чеке)

Описание:

Поле является составным. Оно может содержать 1 и больше подполей, состоящих из элементов данных и имеющих следующую структуру:

Структура подполя					
1	2	3	4	5	6
Тэг	^	Имя	^	Значение	~

Описание элементов структуры подполя проведено в таблице:

№	Элемент	Описание	Обязательность
1	Тэг	Идентификатор данных	O
2	^	Разделитель между элементами данных внутри подполя	M
3	Имя	Название поля для вывода на печать	O
4	^	Разделитель между элементами данных внутри подполя	M
5	Значение	Значение поля	O
6	~	Разделитель между подполями	M

M — mandatory: обязательный элемент;

O — optional: опциональный элемент, может отсутствовать.

Примеры:

Все элементы присутствуют	0x95^TVR^0080048000~0x4F^AID^A0000000031010~
Имя для печати отсутствует	0x95^^0080048000~0x4F^^A0000000031010~
Тэг отсутствует	^TVR^0080048000~^AID^A0000000031010~

Объект «StringConverter»

Метод «Get1251Bytes»:

- Назначение — Преобразование строки
- Формат вызова — Get1251Bytes (string, [Out] byte[], Int32);
- Параметры:
 - **string** — строка, полученная из объекта SAPacket
 - **byte[]** — указатель на массив байт
 - **Int32** — размер массива
- Возвращаемое значение — Количество заполненных байт в результирующем массиве, либо -1, если произошла ошибка

8.2. Свойства объекта «SAPacket»

Таблица 1. Перечень свойств/ полей SAPacket

Свойства	Поле протокола SA/ № поля в XML	Описание	Тип значения
Amount	0	Сумма операции, выраженная в минимальных единицах валюты	String
AdditionalAmount	1	Дополнительная сумма операции, выраженная в минимальных единицах валюты	String
CurrencyCode	4	Код валюты операции	String
DateTimeHost	6	Оригинальная дата и время совершения операции YYYYMMDDHHMMSS на хосте	String
CardEntryMode	8	Способ ввода карты	Integer
PINCodingMode	9	Способ кодировки PIN-блока ¹	Integer
PAN	10	Номер карты	String
CardExpiryDate	11	Срок действия карты YYMM	String
TRACK2	12	Данные Track2	String
AuthorizationCode	13	Код авторизации	String
ReferenceNumber	14	Номер ссылки	String
ResponseCodeHost	15	Код ответа	String
PinBlock	16	Данные PIN-блока	String
PinKey	17	Рабочий ключ PIN	String
WorkKey	18	Рабочий ключ	String
TextResponse	19	Дополнительные данные ответа	String
TerminalDateTime	21	Оригинальная дата и время совершения операции YYYYMMDDHHMMSS на внешнем устройстве	String
TrxID	23	Идентификатор транзакции в коммуникационном сервере	Integer
OperationCode	25	Код операции	Integer
TerminalTrxID	26	Уникальный номер транзакции на стороне внешнего устройства	Integer
TerminalID	27	Идентификатор внешнего устройства	String
MerchantID	28	Идентификатор продавца	String
DebitAmount	29	Сумма дебетовых итогов	String
DebitCount	30	Количество дебетовых итогов	String
CreditAmount	31	Сумма кредитовых итогов	String
CreditCount	32	Количество кредитовых итогов	String
OrigOperation	34	Код оригинальной операции	Integer
MAC	36	Данные MAC	String
Status	39	Статус проведения транзакции ²	Integer
AdminTrack2	40	Track2 карты администратора	String
AdminPinBlock	41	Данные PIN-блока карты администратора	String
AdminPAN	42	Номер карты администратора	String
AdminCardExpiryDate	43	Срок действия карты администратора	String
AdminCardEntryMode	46	Способ ввода карты администратора	Integer
VoidDebitAmount	49	Сумма дебетовых отмен	String

¹ Если в ответе для **ОДОБРЕННОЙ** транзакции значение данного свойства равно 1 или 2, то это означает, что транзакция была подтверждена PIN-кодом.

² Описание свойства **Status** смотри ниже в данном пункте

Свойства	Поле протокола SA/ № поля в XML	Описание	Тип значения
VoidDebitCount	50	Статус получения Dual Connector результата операции от терминала (при обмене с кассовым ПО не используется) ³	String
VoidCreditAmount	51	Статус получения кассовым ПО результата операции от терминала	String
VoidCreditCount	52	Номер слипа ³	String
ProcessingFlag	53	Флаг обработки операции	Integer
HostTrxID	54	Идентификатор транзакции на хосте	Integer
RecipientAddress	56	Адрес получателя	Integer
CardWaitTimeout	57	Таймаут ожидания карты	Integer
DeviceSerNumber	63	Серийный номер	String
CommandMode	64	Режим выполнения команды	Integer
CommandMode2	65	Режим выполнения команды 2	Integer
CommandResult	67	Статус (результат) выполнения команды	Integer
FileData	70	Данные (файл)	String
MessageED	71	Сообщение для вывода на экран ВУ	String
CashierRequest	76	Запрос к кассиру	String
CashierResponse	77	Ответ кассира	String
AccountType	79	Тип счёта клиента	String
CommodityCode	80	Код платежа	String
PaymentDetails	81	Детали платежа	String
ProviderCode	82	Код провайдера	String
Acquirer	83	Эквайер	String
AdditionalData	86	Дополнительные данные транзакции	String
ModelNo ⁴	89	Наименование модели ВУ	String
ReceiptData	90	Данные для печати на чеке	String

³ При включенном параметре CONFIRM_OPERATION (п.3.1) поля используются для подтверждения операции на стороне DualConnector.

⁴ В данное поле может добавить информацию о своей версии касса, Dual Connector.