

Dual Connector 2.0

Руководство программиста

Версия 2.0.8

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 Ошибка! Источник ссылки не найден. (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 Реализация интеграции с использованием DualConnector 2.0 (DC Service). 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 Реализация интеграции с использованием DualConnector 2.0 (DC Service). Прямая интеграция с DC Service по HTTP», «4.1 Основные параметры».
1.6	14.09.2021	Актуализированы пункты: «3 Реализация интеграции с использованием DualConnector 2.0 (DC Service). Прямая интеграция с DC Service по HTTP», «4.1 Основные параметры».
1.7	19.10.2022	Добавлен пункт «4.2 Настройка работы с несколькими терминалами по TerminalID»

Содержание

Правовая информация и сведения о поддержке продукта	4
1. Введение	5
2. Общие сведения о Dual Connector 2.0	6
2.1. Назначение	6
2.2. Системные требования	7
2.3. Установка в системе	7
3. Реализация интеграции с использованием DualConnector 2.0 (DC Service).	8
3.1 Прямая интеграция с DC Service по HTTP.....	8
3.2 Реализация работы с DC Service через компоненты DC Console Service	11
4. Настройка параметров «DualConnector 2.0» (DC Service)	13
4.1. Основные параметры	13
4.2. Настройка работы с несколькими терминалами по TerminalID	14
5. Примеры работы с «DualConnector 2.0» («DC Service»)	16
5.1. C++	16
5.2. C#	17
5.3. Java	17
6. Особенности использования модуля DC Proxy	19
7. Взаимодействие с оператором	19
8. Приложение	20
Альтернативный способ реализации вывода терминальных окон.	20

Правовая информация и сведения о поддержке продукта

Dual Connector 2.0. Версия 2.0.8 Руководство программиста: М.: ООО "Лаборатория платежных решений", 2023. — 25с.

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 2.0.8 © 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 программистов, реализующим взаимодействия между клиентским кассовым ПО и терминальным ПО «UNIPOS Terminal (с функционалом Smart Sale)» или «Unipos Droid (в режиме Smart Sale)» с использованием вспомогательного ПО «**DualConnector 2.0**» (**DC Service**)».

2. Общие сведения о Dual Connector 2.0

2.1. Назначение

«DualConnector 2.0» (DC Service) - сервис для интеграции кассового ПО на базе «Windows (XP/7/8/10/11)» 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, TCP/IP, Bluetooth 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 вспомогательного ПО «DualConnector 1.x».

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 (USB). Трансляция сообщений между RS232 (USB) 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.

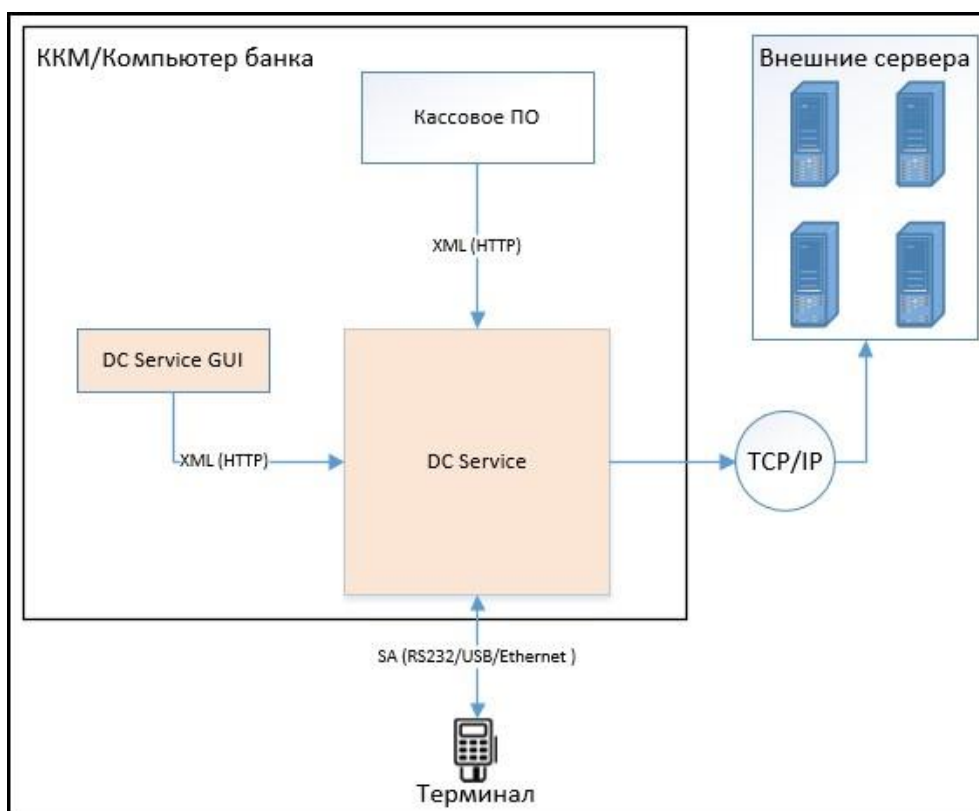


Рисунок 1. Организация взаимодействия с DualConnector 2.0

- **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/7/8/10/11;
- 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.

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 при использовании модуля «**DC Proxy**» (см. п.п. 3.2).

Внимание!

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

3. Реализация интеграции с использованием DualConnector 2.0 (DC Service).

3.1 Прямая интеграция с DC Service по HTTP

Обмен между кассовым ПО и DC Service осуществляется по HTTP протоколу.

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

Пример HTTP заголовка с использование метода POST:

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

Пример HTTP заголовка с использование метода GET:

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

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

Для корректной обработки данных в http-заголовке 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 кодировке. Если кодировка не указана, то данные будут интерпретированы в кодировке по умолчанию – «**ASCII**».

В заголовке «**Accept**» регулируется ответ на полученный запрос от кассы. При ошибке, в ответе формируется и передается файл «**XML**» или «**HTML**».

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 ответа при возникновении ошибки при значении параметра Accept: text/xml:

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

При этом, возможные значения «**errorcode**»:

- **0** – DC не смог передать ответ от терминала. Возможная причина, внутренняя ошибка DC;
- **1** – истекло время исполнения операции. Устанавливает поле «**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 в отсутствии 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.

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

Пример ответа при возникновении ошибки при значении параметра Accept: text/html:

```
HTTP/1.1 404 Not Found
Content-Type: text/html; charset=UTF-8
Date: Fri, 14 Apr 2023 13:33:47 GMT
connection: close
Content-Length: 79
```

<h1>DEVICE_ERROR</h1><h>Can't open device COM2. Serial port COM2\$ not found</h>

При этом, возможные значения **«H1»**:

- OK - Нет ошибок;
- OK_PARTIAL - Нет ошибок, часть пакета;
- IO_PENDING - Недостаточно данных;
- ERROR - Общая ошибка;
- FIELD_ERROR - Неверное значение поля;
- DATA_ERROR - Ошибка данных;
- DATA_LEN_ERROR - Ошибка длины данных;
- ACK_ERROR - Ошибка отправки/получения ACK;
- CRC_ERROR - Ошибка контрольной суммы;
- NAK_ERROR - Ошибка. превышено количество негативных подтверждений;
- PACKET_ERROR - Ошибка. разбора пакета;
- ERR_EOT - Получен сигнал завершения соединения;
- DEVICE_ERROR - Ошибка устройства;
- TIMEOUT - истекло время ожидания;
- CANCEL – Отменено;
- DEVICE_BUSY - Устройство занято;
- RESPONSE_ERROR - Устройство занято;
- EXCHANGE_TIMEOUT - истекло время обмена данными;
- EXCHANGE_END_WITH_TERMINAL - сессия обмена с терминалом завершена.

Для получения ответа по результату операции в определенной кодировке необходимо использовать заголовок Accept-Charset. В заголовке Accept-Charset необходимо указывать тип кодировки, в которой необходимо получить ответ о результат операции.

Для получения ответа по результату операции в кодировке utf-8 в заголовке должен быть использован заголовок Accept-Charset со значением utf-8.

```
POST / HTTP/1.1
Host: 10.35.91.20:9015
User-Agent: curl/7.53.1
Accept: */*
Content-Type: text/html
Accept-Charset: utf-8
```

Content-Length: 305

Для получения ответа по результату операции в кодировке windows-1251 в заголовке должен быть использован заголовок Accept-Charset со значением windows-1251.

POST / HTTP/1.1

Host: 10.35.91.20:9015

User-Agent: curl/7.53.1

Accept: */*

Content-Type: text/html

Accept-Charset: windows-1251

Content-Length: 305

В случае отсутствия заголовка Accept-Charset в запросе, ответ с результатом операции будет передан в кодировке windows-1251.

Непосредственно запрос оформляется 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 (в примере ниже выделено красным шрифтом).

Например.

```
<?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>
```

```
  <field hex="true"
```

```
    id="86">EE38D10430303530D228393534373841343742414442384443333337313132333830343333341  
    364436384438424145424534D306343038313131</field>
```

```
</request>
```

Где:

№ поля в XML/ Поле протокола SA	Описание
0	Сумма операции, выраженная в минимальных единицах валюты
4	Код валюты операции
21	Оригинальная дата и время операции YYYYMMDDHHMMSS на внешнем устройстве
25	Код операции
26	Уникальный номер транзакции на стороне внешнего устройства
27	Идентификатор внешнего устройства (Terminal ID)
86	Дополнительные данные
90	Данные для печати на чеке

Примечание. Перечень полей, которые необходимо передавать для конкретного типа операции, указан в отдельном документе «Кассовые решение на базе SmartSale.»

Формат запроса предусматривает возможность указать напрямую адрес терминала, подключенного по «TCP/IP» или «COM/USB» и максимальное время выполнения операции. (При этом данные для соединения, указанные в файле настройки «Connector.xml» (см. п.п. 4.) будут

игнорироваться). Данная возможность позволит маршрутизировать запросы на тот или иной терминал, когда к кассовой сети подключено большое количество терминалов, и регулировать максимальное время выполнения операции.

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;

Максимальное время операции (для всех типов коммуникаций):

- **timeout** – предоставляемое время на выполнение операции в секундах. 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). Если таймаут не указан, то значение равно 0, бесконечное ожидание. Параметр обрабатывается, только для команд от кассы.

Пример запроса с указанием ip-адреса терминала:

```
<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>
  <ipaddr>192.168.0.2:9000</ipaddr>
  <timeout>180</timeout>
</request>
```

Пример запроса с указанием COM-порта, к которому подключен терминал:

```
<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>
  <ipaddr>192.168.0.2:9000</ipaddr>
  <timeout>180</timeout>
  <ncom>9</ncom>
  <baudrate>115200</baudrate>
</request>
```

В случае необходимости прерывания операции с терминалом до получения результата выполнения операции на стороне терминального ПО, со стороны кассового ПО необходимо отправлять пакет RST.

3.2 Реализация работы с DC Service через компоненты DC Console Service

Для реализации работы с DC Service можно использовать компоненту «DC Console Service». Компонента «DC Console Service» предназначена для организации взаимодействия кассового ПО и ПО терминала с использованием параметров командной строки. Подробнее работа с компонентой DC Console описана в документе «DCConsole Service.pdf».

4. Настройка параметров «DualConnector 2.0» (DC Service)

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».

Примечание. Для настройки файла конфигурации «DualConnector 2.0» можно воспользоваться утилитой «DC Control», которая входит в дистрибутив «DualConnector 2.0». Подробное описание приведено в документе «DC Control Service.pdf».

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>
  <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>
```

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. **IPADDRESS** – IP адрес терминала. Наличие обязательно при соединении по IP.
13. **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.
14. **WAITACK** – время ожидания сигнала подтверждения получения пакета в секундах. Наличие необязательно, по умолчанию 5.
15. **WAITPACKET** – время ожидания ответного пакета в секундах (или миллисекундах при значениях выше 300). Наличие необязательно, по умолчанию 45.
16. **CONNECT_TIMEOUT** – механизм прерывания установки соединения по истечению времени. Указывается время ожидания соединения с сервером в секундах. Значение по умолчанию – 30 секунд.
17. **EXCHANGE_TIMEOUT** – устанавливает максимальное время выполнения операции в секундах.
18. **RECONNECTION_DELAY** – устанавливает задержку в секундах на повторное подключение/соединение к серверу после отключения в секундах.
19. **HEX_STRING_FORMAT** – указывает формат поле в ответе (response) в XML-файле. При необходимости перекодирует 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.

4.2. Настройка работы с несколькими терминалами по TerminalID

DualConnector 2.x поддерживает работу с несколькими терминалами, используя значение параметра «TerminalID». (Значение «TerminalID» настраивается сотрудниками банков).

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>
  <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>
```

5. Примеры работы с «DualConnector 2.0» («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 WinHttpRequestData.\n", GetLastError());
    else
        printf("%s", pszOutBuffer);
    // Free the memory allocated to the buffer.
    delete[] pszOutBuffer;
    }
    } while (dwSize > 0);
}
// Report any errors.
if (lResults) 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 serviceIpAddress = "127.0.0.1:9015";
        string head = "http://" + serviceIpAddress;
        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 модуля DC Proxy

Модуль **DC Proxy** – реализует открытие API в виде вызова динамической библиотеки для сохранения интеграций, выполненных с использованием «**DUALConnector 1.x**». Он позволяет пользователям перейти на работу с «**DC Service**» не меняя своё кассовое ПО, работающее с «**DUALConnector 1.x**». 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 способом реализации является использование модуля «**DC Service GUI**» (входит в состав «**DualConnector 2.0**»). Для использования «**DC Service GUI**» необходимо 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 подразумевает реализацию в кассовом ПО обработку поступающих команд на вывод окон и описан в п.п 8 данного документа.

8. Приложение

Альтернативный способ реализации вывода терминальных окон.

Альтернативный способ вывода окон необходим, когда производитель кассового ПО решает использовать свою реализацию диалоговых окон. В данном случае на стороне кассового ПО необходимо 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>
```

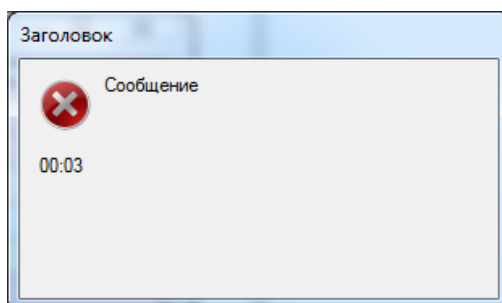
Формат ответа:

```
<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>
```

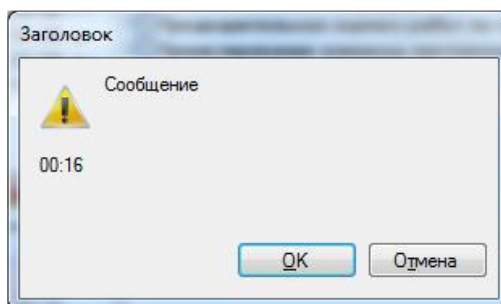
Где:

- 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>
```

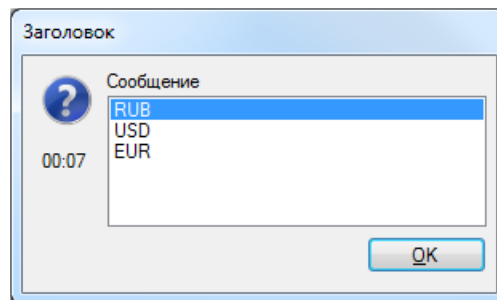
Формат ответа:

```
<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>
```

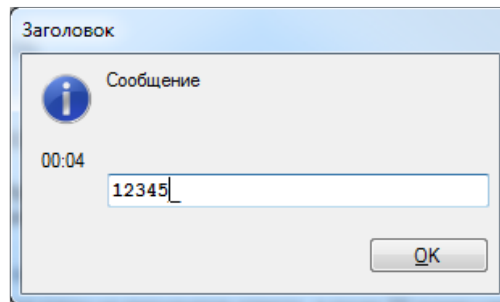
Формат ответа:

```
<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.

Информационное сообщение с уникальным идентификатором

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

Формат запроса:

```
<request>
  <type>6</type>
  <data>6^^Заголовок^Уникальный идентификатор^Сообщение</data>
  <timeout>10</timeout>
</request>
```

Формат ответа:

```
<response>
  <type>6</type>
  <data>0</data>
</response>
```

Где:

- type - 6 – информационное сообщение с уникальным идентификатором;
- 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).

Заккрытие окна

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

Формат запроса:

```
<request>
  <type>7</type>
  <timeout>0</timeout>
</request>
```

Формат ответа:

```
<response>
  <type>7</type>
  <data>0</data>
</response>
```





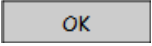
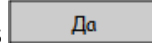
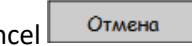

Где:

- type - 7 – закрытие окна;
- timeout - время отображения окна в секундах, должно быть значение 0;
- data (в ответе) - ответ кассира (в данном случае 0, кассир ничего не нажимал и не должен).

Возможные значения поля data в ответе кассира:

- 0 – окно закрыто;
- 1 – произошла ошибка.

Формат данных для отображения

Элемент подполя	Описание	Атрибут	Тип поля
Уровень сообщения	Определяет стиль иконки окна, выводимого на ККМ. Может принимать значения: <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	Определяет элементы управления (кнопки), которые должны быть отрисованы в окне, выводимом на ККМ. Поле представляет собой битовую маску: 0x01 – Ok  MB_OK 0x02 – Yes  MB_YES 0x04 – Cancel  MB_CANCEL 0x08 – No  MB_NO	O	n..3
^	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	М	ап..950
-------------------	---	---	---------

М – mandatory: обязательный элемент;

О – 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