



ул. Октябрьская, д. 72.
Москва, 127521, Россия
Тел./факс: +7 495 721 36 21

Dual Connector

Версия 1.2

Руководство программиста

Содержание

1 Правовая информация и сведения о поддержке продукта	4
2 Введение	6
2.1 Назначение	6
2.2 Системные требования	7
2.3 Порядок использования API	7
2.3.1 Описание API	7
3 Настройка параметров Dual Connector	13
3.1 Основные параметры	13
3.2 Дополнительные параметры	15
3.3 XmlGenerator	15
3.4 Инсталляция в системе	15
4 Рекомендации по использованию библиотеки DualConnector	17
4.1 C++	17
4.2 C#	18
4.3 Взаимодействие с оператором	18

Раздел



1 Правовая информация и сведения о поддержке продукта

Dual Connector. Версия 1.2. Руководство программиста: М.: ООО “ИНПАС КОМПАНИ”, 2017. — 23 с.

ООО “ИНПАС СОФТ” 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. Особенности конкретной версии программного обеспечения приведены в файле New.pdf.

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 является ООО “ИНПАС СОФТ”.

Dual Connector Версия 1.2 © ООО “ИНПАС СОФТ” 2017

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

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.

Линия E-mail - консультаций:

support@inpas.ru

Адрес сайта технической поддержки:

<http://support.inpas.ru/> (www.inpas.ru)

Адрес: 127521, Москва, Октябрьская ул., 72

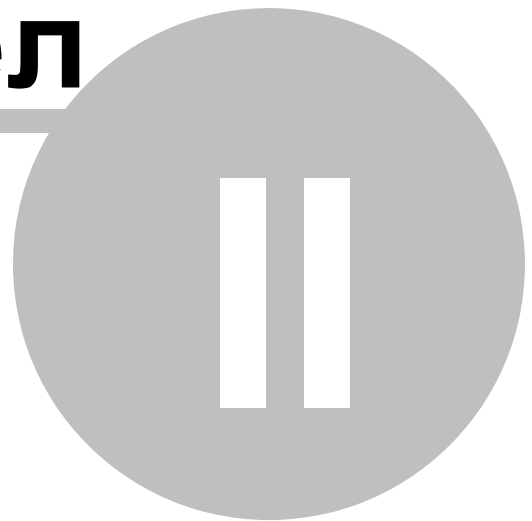
Телефоны для справок: (495) 721-36-21

Факс: (495) 721-36-21

E-Mail: support@inpas.ru

Web: <http://www.inpas.ru/>

Раздел



2 Введение

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

2.1 Назначение

DualConnector представляет из себя .Net библиотеку, реализующую интерфейс обмена с терминалом по протоколу SA. Для интеграции с кассовым ПО DualConnector представлен в GAC и зарегистрирован как COM-объект. В процессе проведения транзакции на терминале DC может предоставить терминалу возможность обмена с хостом банка средствами ОС с или без использования SSL. Т.о. отпадает необходимость иметь на терминале коммуникационные модули.

Позволяет устанавливать SSL соединение, используя предоставляемые терминалом сертификаты.

На *рисунке 1* представлена схема взаимодействия участников процесса обмена данными.

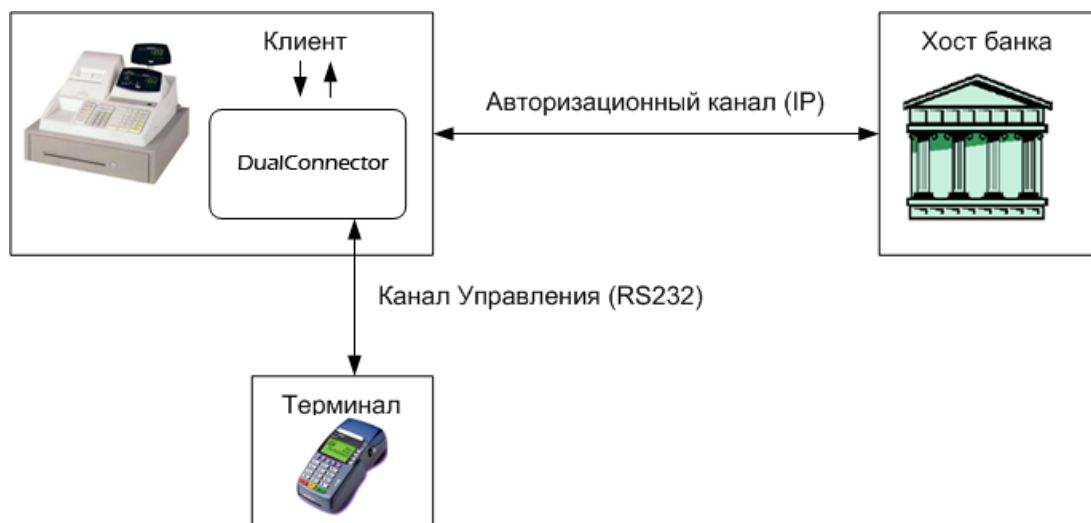


Рис. 1. Организация взаимодействия с DualConnector

DualConnector предоставляет клиентскому ПО следующие интерфейсы:

- **ISAPacket** – для создания экземпляров объектов, содержащих информацию, необходимую для проведения транзакций;
- **DualConnectorInterface** – для создания экземпляров объектов, непосредственно организующих взаимодействие;

2.2 Системные требования

- Версия ОС: Windows XP/7/8/10 (разрядность x32);
- Разрядность x32 или x64 (в режиме эмуляции x32);
- Установленный пакет .NET Framework 3.0 и выше;
- Права доступа «Администратор».

2.3 Порядок использования API

Для работы с Dual Connector 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 выполнение и только после этого отправлять повторный запрос на выполнение операции.

2.3.1 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 пина клиентом, обмен данными.
 Рекомендуемое значение 180000 (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

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.

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”).

Объект SAPacket**Свойства объекта**

Свойство	Поле протокола SA	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

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

Свойство	Поле протокола SA	Описание	Тип значения
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
VoidDebitCount	50	Количество дебетовых отмен	String
VoidCreditAmount	51	Сумма кредитовых отмен	String
VoidCreditCount	52	Количество кредитовых отмен	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	89	Наименование модели ВУ	String
ReceiptData	90	Данные для печати на чеке	String
TermResponseCode	106	Код подтверждения получения ответа от терминала	Integer
SlipNumber	108	Номер слипа	String

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

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

описание причины ошибки.

Используются следующие статусы проведения транзакций:

Значение	Описание
0	Неопределённый 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 приведено в таблице:

	Элемент	Описание	Обязательность
1	Тэг	Идентификатор данных	О
2	^	Разделитель между элементами данных внутри подполя	М
3	Имя	Название поля для вывода на печать	О
4	^	Разделитель между элементами данных внутри подполя	М
5	Значение	Значение поля	О
6	~	Разделитель между подполями	М

М – mandatory: обязательный элемент;

О – 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

Раздел



3 Настройка параметров Dual Connector

3.1 Основные параметры

Основной файл параметров должен находиться в директории с DualConnector.dll и называться DualConnector.xml.

Содержит данные в следующей структуре xml:

```
<ROOT>
  <LOG>
    <TYPE>DEBUG</TYPE>
    <PATH>Z:/LOG</PATH>
    <CLEARTIME>30</ CLEARTIME >
  </LOG>
  <FREERESOURCE_AUTO>OFF</FREERESOURCE_AUTO>
  <CONFIRM_OPERATION>OFF</ CONFIRM_OPERATION >
  <TRIPLEACK >OFF</ TRIPLEACK>
  <DEVICES>
    <DEVICE>
      <TYPE>TERMINAL</TYPE>
      <CONNECTION>
        <TYPE>COM</TYPE>
        <PORT>COM10</PORT>
        <BAUDRATE>115200</BAUDRATE>
        <IPADDR>192.168.0.2:7777</IPADDR>
      </CONNECTION>
      <SA>
        <WAITACK>6</WAITACK>
        <WAITPACKET>45</WAITPACKET>
      </SA>
      <GUI>
        <IPADDR>127.0.0.1:6060</IPADDR>
      </GUI>
    </DEVICE>
  </DEVICES>
  <OUTPUT>
    <SSL>ON</SSL>
    <SSLTYPE>TLS</SSLTYPE>
    <CHECKNAME>OFF</CHECKNAME>
    <CERTNAME></CERTNAME>
  </OUTPUT>
</ROOT>
```

1. **ROOT** – корневая область. Наличие обязательно.
2. **LOG** – секция настройки ведения лог файла. Наличие обязательно.
 - 2.1. **TYPE** – тип детализации информации. Допустимые значения в порядке увеличения выводимой информации: NONE, SYSTEM, ADVANCED. Наличие необязательно, по умолчанию NONE.
 - 2.2. **PATH** – путь к папке ведения логов. При отсутствии лог ведётся в директории программы. Может содержать переменные окружения, например, %USERPROFILE%
 - 2.3. **CLEARTIME** – время хранения логов (в днях). Диапазон возможных значений от 1 до 365 дней. Если параметр не задан, используется значение по умолчанию, 30 дней.
3. **FREERESOURCE_AUTO** – секция настройки автоматического вызова FreeResources. Наличие не обязательно. После окончания транзакции будет автоматически вызываться метод

FreeResources при значении опции «ON». Остальные значения или отсутствие секции означает, что метод вызываться не будет.

4. **CONFIRM_OPERATION** – секция настройки включения автоматического подтверждения операции на стороне внешнего устройства. Наличие не обязательно. По умолчанию, выключено.
5. **TRIPLEACK** – секция настройки отправки 3 символов подтверждения (ACK) по завершении операции на терминал. Наличие не обязательно. По умолчанию, выключено.
6. **DEVICES** – описание подключённых терминалов. Наличие обязательно.
 - 6.1. **DEVICE** – описание подключенного терминала или пинпада
 - 6.1.1. **TYPE** – тип терминала. Допустимые значения **TERMINAL**, **PINPAD**. Наличие необязательно. По умолчанию **TERMINAL**. Отличие типов используется для определения наличия принтера. Если в ответе получен образ чека от **PINPAD**, то он будет перенаправлен на первое в списке устройство с типом **TERMINAL**.
 - 6.1.2. **CONNECTION** – описание подключения к терминалу. Наличие обязательно
 - 6.1.2.1. **TYPE** – тип подключения. Допустимые значения **COM**, **IP**.
 - 6.1.2.2. **PORT** – номер **COM** порта. Наличие обязательно при соединении по **COM**
 - 6.1.2.3. **BAUDERATE** – скорость обмена. Наличие необязательно. По умолчанию 115200.
 - 6.1.2.4. **IPADDR** – IP адрес терминала. Наличие обязательно при соединении по **IP**
 - 6.1.3. **SA** – Описание параметров протокола. Наличие необязательно
 - 6.1.3.1. **WAITACK** – время ожидания сигнала подтверждения получения пакета в секундах (или миллисекундах при значениях выше 300). Наличие необязательно, по умолчанию 5.
 - 6.1.3.2. **WAITPACKET** – время ожидания ответного пакета в секундах (или миллисекундах при значениях выше 300). Наличие необязательно, по умолчанию 45.
 - 4.1.4. **GUI** – сервер отображения запросов терминала. Наличие необязательно
 - 4.1.4.1. **IPADDR** – IP адрес сервера, на который адресуются запросы терминала. Описание протокола смотрите в соответствующем разделе данного документа.
7. **OUTPUT** – секция выходных параметров. Наличие необязательно.
 - 7.1. **SSL** – Признак обработки данной секции параметров

SSLTYPE – Тип протокола соединения SSL. Варианты работы DC в зависимости от файла конфигурации.

		SSL TYPE		
		Отсутствует	TLS	SSL3
SSL	Отсутствует	FAIL	FAIL	FAIL
	ON	TLS	TLS	SSL3
	OFF	TLS	TLS	SSL3

- 7.2. **CHECKNAME** – Только при наличии параметра и значения **OFF** не осуществляется проверка имени серверного сертификата.
- 7.3. **CERTNAME** – Имя клиентского сертификата (подставляется вместо имени из самого сертификата) при хранении сертификатов на кассе, или имя серверного сертификата, при получении сертификатов с терминала.

3.2 Дополнительные параметры

Для различных схем использования может потребоваться хранить основные параметры не в директории программы.

Тогда в директорию помещается файл со следующими параметрами:

```
<ROOT>  
    <PATH>d:\params.xml </PATH>  
</ROOT>
```

PATH – путь к файлу с основными параметрами. Может содержать переменные окружения, например, %USERPROFILE%

3.3 XmlGenerator

Для настройки файла конфигурации DualConnector можно воспользоваться утилитой DC XML Generator, которая входит в дистрибутив DualConnector. Подробнее можно ознакомиться в руководстве пользователя на DC XML Generator.

3.4 Инсталляция в системе

Установка DualConnector в системе осуществляется только с помощью инсталляционного пакета. В рамках установки происходит регистрация библиотек в среде COM, в GAC 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 строка «License activated».

Лицензия не влияет на работу ПО. Она необходима только для включения отладочного (DEBUG) режима ведения лога.

Для использования DualConnector без регистрации в GAC требуется выполнить следующие пункты:

- Выполнить копирование всех файлов с расширением .ddl из директории установки DualConnector в директорию кассового ПО, при этом удаление файлов из установленной по умолчанию директории DualConnector запрещено;
- Выполнить команды, под учетной записью пользователя имеющей права администратора в операционной системе:
 - Regasm.exe DualConnector.dll /codebase;
 - Regasm.exe Managedopenssl.dll /codebase.

Раздел



IV

4 Рекомендации по использованию библиотеки DualConnector

Внимание! Модули Dual Connector (DC) размещаются в Global Assembly Cache (GAC) операционной системы. При интеграции кассового ПО с DC не допускается:

- Поиск библиотек в коде по абсолютному или относительному пути;
- “Привязка” кода к определенной версии DC.

4.1 C++

```
CoInitialize(NULL);
DualConnector::ISAPacketPtr request;
DualConnector::ISAPacketPtr response;
DualConnector::DualConnectorInterfacePtr dc;

dc.CreateInstance(__uuidof(DualConnector::DCLink ));
request.CreateInstance(__uuidof(DualConnector::SAPacket));
response.CreateInstance(__uuidof(DualConnector::SAPacket));
if ( dc == NULL || request == NULL || response == NULL )
{
    MessageBox( _T("COM init error!"), _T("Error!"), MB_OK | MB_ICONERROR );
    return;
}

DualConnector::ISAPacket *pRequest = request.Detach();
DualConnector::ISAPacket *pResponse = response.Detach();

int res = dc->InitResources();
if ( res != 0 )
{
    CString mes;
    mes.Format( _T("Init result: %d - %s"), res, (LPTSTR)dc->ErrorDescription );
    MessageBox( mes, _T("Error!"), MB_OK );
}

pRequest->Amount = "1";
pRequest->CurrencyCode = "643";
pRequest->OperationCode = 1;
pRequest->TerminalID = "00000003";

exchangeResult = dc->Exchange( &pRequest, &pResponse, timeout );

dc->FreeResources();
long Status = pResponse->Status;
CString mes;
mes.Format( _T("Exchange result: %d, Status: %d"), exchangeResult, Status );
if ( pResponse->TextResponse.length() )
{
    mes.AppendFormat( _T(", Text: %s"), (LPCWSTR)pResponse->TextResponse );
}
MessageBox( mes, _T("OK!"), MB_OK );

// Пример преобразования строкового представления параметра в массив байт
// Инициализация StringConverter
DualConnector::IStringConverterPtr converter;
converter.CreateInstance(__uuidof(DualConnector::StringConverter));
// Определение длины массива, достаточной для сохранения байтового представления строки
int arrayLength = converter->Get1251Bytes(pResponse->ReceiptData, NULL, 0);
// Создание SAFEARRAY нужной длины
SAFEARRAY* safeArray = SafeArrayCreateVector(VT_UI1, 0, arrayLength);
// Заполнение SAFEARRAY
converter->Get1251Bytes(pResponse->ReceiptData, safeArray, arrayLength);

// TODO: safeArray содержит теперь байтовое представление строки

// Освобождаем safearray
```

```
SafeArrayDestroy(safeArray);
safeArray = NULL;

dc->Release();
pRequest->Release();
pResponse->Release();
CoUninitialize();
```

Также существует асинхронный способ вызова Exchange. Для этого необходимо подписаться на событие COM объекта с DISPID = 0x01. В этом случае параметр Timeout в вызове Exchange должен быть равен 0.

4.2 C#

```
DualConnector.DCLink dclink = new DualConnector.DCLink();
DualConnector.ISAPacket query = new DualConnector.SAPacket();
DualConnector.ISAPacket response = new DualConnector.SAPacket();
query.Amount = "001";
query.CurrencyCode = "643";
query.OperationCode = 1;
query.TerminalID = "00000003";
//dclink.OnExchange += new DualConnector.OnExchangeHandler(dclink_OnExchange);
int res = dclink.InitResources();
if ( res != 0 )
{
    MessageBox.Show(string.Format("Init resource:{0}-{1}",res,dclink.ErrorDescription) );
}
res = dclink.Exchange( ref query, ref response, 180000 );
if ( res != 0 )
{
    MessageBox.Show(string.Format("Exchange:{0}-{1}",res,dclink.ErrorDescription));
}
dclink.Dispose();
```

Для асинхронного вызова Exchange, необходимо подписаться на событие OnExchange: `dclink.OnExchange += new DualConnector.OnExchangeHandler(dclink_OnExchange);`
В этом случае параметр Timeout в вызове Exchange должен быть равен 0.

4.3 Взаимодействие с оператором

DualConnector имеет возможность транслирования запросов терминала для отображения сообщений кассиру.

Взаимодействие может быть организовано основным или альтернативным способом.

Основным является использование ПО DC PosGUI. Для его использования необходимо убедиться в том, что модуль установлен (выбрать галочку при установке) и указать настройки соединения, которые находятся в файлах конфигурации DualConnector.xml и DC PosGUI.xml. Подробная информация и пример файла конфигурации представлены в разделе **“Настройка параметров Dual Connector”**. Данные в DC PosGUI 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 2 способа 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.

- Использовать событие OnShowWindow из API DualConnector (см. раздел **“Описание API”**). При подписке на это событие, DualConnector не будет использовать DC PosGUI для вывода окон. При необходимости отобразить то или иное диалоговое окно, будет вызвано событие. В аргументах события будет содержаться пакет SA, разобрав который, можно выяснить какое окно и с каким содержимым требуется отобразить.

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:

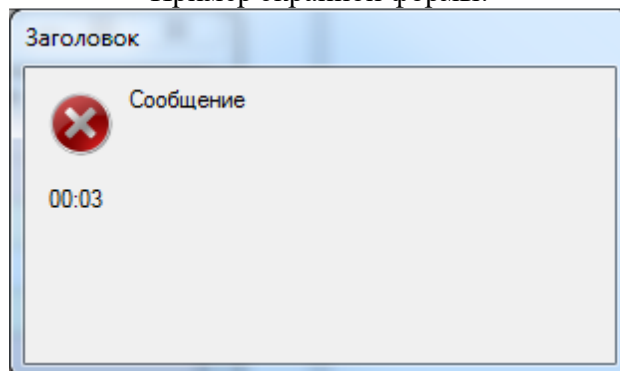
type - 1 – информационное сообщение;

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>
```

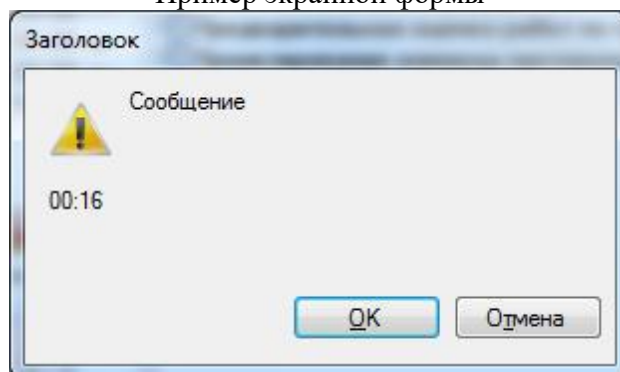
Где:

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:

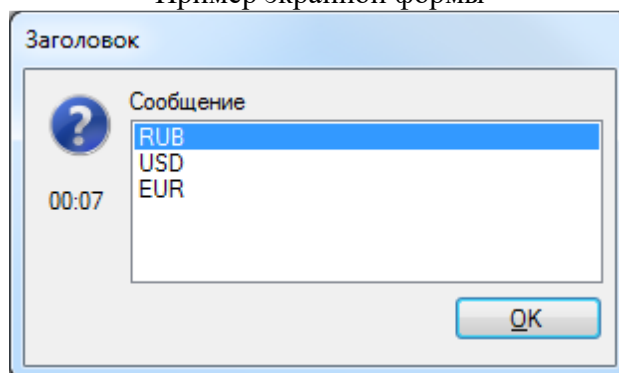
```
<response>
  <type>3</type>
  <data>1</data>
  <adata>4</adata>
</response>
```

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 $N=2^m$, 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>
```

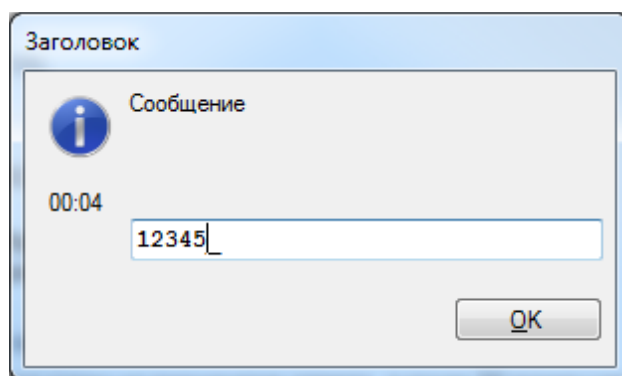
Формат ответа:

```
<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 	О	n1
^	Разделитель между элементами данных внутри поля.	М	
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> <p>0x04 – Cancel MB_CANCEL</p> <p>0x08 – No MB_NO</p>	О	n..3
^	Разделитель между элементами данных внутри поля.	М	

Заголовок сообщения	Заголовок окна, выводимого на ККМ	О	an..40
^	Разделитель между элементами данных внутри поля.	М	
Сообщение кассиру	Строка (или набор строк, разделенных символом «\n» или «;»), содержащая текст информационного сообщения для кассира	М	an..950

Примечание:

М – обязательный элемент (mandatory);

О – опциональный элемент, может отсутствовать (optional).

Например:

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