

Dual Connector

Руководство программиста

Версия 2.2

История изменений документа

Версия	Дата	Перечень изменений
1.0	10.07.2018	Первоначальная версия
1.1	05.03.2019	Актуализирован пункт 1.3.1 Описание API
1.2	29.05.2019	Добавлена сноска №4 для поля 89 в пункт 1.3.1 «Объект «SAPacket»
1.3	16.01.2020	Изменен пункт 3 и 3.1
1.4	14.09.2021	Изменены пункты 1.3.1, 2.1, 2.4
1.5	04.05.2022	В п. 1.2 удалено ограничение по ОС x32
1.6.	18.10.2022	В п. 1.3.1 для 89 поля обновлена сноска №4
1.7	03.10.2023	Изменены пункты 1.3.1, 2.1, 2.3, 3.4. Добавлено описание способов создания файлов в п 2.3, 2.3.1, 2.3.3 Добавлен пункт 2.5
1.8	01.11.2023	Обновлен пункт 2.5 «Параметры настройки DC PosGUI» Актуализирован пункт 2.1 «Основные параметры» Добавлен пункт 4 «Приложение 1. Подключение нескольких терминалов к ККМ.»
1.9	06.02.2024	В пункте «1.2 Системные требования» изменены требование к установленным пакетам «.NET Framework 4.0», исключена поддержка работы на Windows XP Добавлено примечание к методу «SetChannelTerminalParam» в пункте «1.3 Порядок использования API» Добавлена «Обработка исключений» в пункте «1.3 Порядок использования API». Добавлено примечание к уровням логирования в пункте «2.1 Основные параметры»
2.0	24.09.2024	Добавлен новый параметр USEMUTEX в пункте «2.1 Основные параметры»
2.1	09.12.2024	Обновлен п. «1.2 Системные требования»: Выполнена доработка библиотеки DC для совместимости с проектами на .NET 5.0 В п. 1.3.1 « <u>Описание API</u> »: Добавлены сноски к методу «SetField» и «GetField»; Добавлено примечание в описании метода «Cancel». Добавлен п. 5 <u>Приложение 2. Пример работы Dual Connector на C#.</u>
2.2	11.09.2025	В п. «1.3 Порядок использования API» уточнён алгоритм инициализации библиотеки: инициализация выполняется один раз за время работы приложения,

Версия	Дата	Перечень изменений
		<p>метод Exchange может вызываться многократно без повторной инициализации.</p> <p>В п. «1.3.1 Описание API»:</p> <ul style="list-style-type: none"> к методу InitResources добавлены особенности работы (повторный вызов, отключение FREERESOURCE_AUTO, освобождение ресурсов вручную, поведение объектов ISAPacket). добавлено описание события OnExchange. для событий OnExchange и OnShowWindow добавлено примечание о проверке множественной подписки обработчиков. добавлено описание дополнительных свойств объекта SAPacket для передачи служебной информации: VersionDC, VersionTerminalDriver, VersionJVMonKMM. <p>В п. «2.1 Основные параметры» уточнено описание параметра FREERESOURCE_AUTO, рекомендуемое значение «OFF».</p> <p>В п. «3 Рекомендации по использованию библиотеки «DualConnector»» добавлено предупреждение о необходимости создавать новый экземпляр ISAPacket (Request/Response) для каждой операции или очищать поля перед повторным использованием.</p> <p>Добавлен п. «3.3 Примеры для режима с автоматической очисткой ресурсов (FREERESOURCE_AUTO=ON)» с примерами использования API при инициализации перед каждой операцией.</p>

Содержание

Правовая информация и сведения о поддержке продукта	5
1. Введение	6
1.1. Назначение.....	6
1.2. Системные требования	6
1.3. Порядок использования API	6
1.3.1. Описание API.....	7
2. Настройка параметров «DualConnector».....	14
2.1. Основные параметры.....	14
2.2. Дополнительные параметры.....	16
2.3. Создание файла параметров «DualConnector.xml» в директории установки.....	16
2.3.1. Создание и настройка файла параметров при помощи «XML Generator»	16
2.3.2. Копирование заранее созданного файла параметров во время инсталляции.	16
2.3.3. Создание нового файла конфигурации при помощи ключей параметров во время инсталляции.	17
2.4. Инсталляция в системе	18
2.5. Параметры настройки DC PosGUI.....	18
3. Рекомендации по использованию библиотеки «DualConnector».....	20
3.1. C++.....	20
3.2. C#.....	22
3.3. Примеры для режима с автоматической очисткой ресурсов (FREERESOURCE_AUTO=ON)	22
3.3.1. C++	22
3.3.2. C#	24
3.4. Взаимодействие с оператором	24
4. Приложение 1. Подключение нескольких терминалов к ККМ.	30
5. Приложение 2. Пример работы Dual Connector на C#.	32

Правовая информация и сведения о поддержке продукта

Dual Connector. Версия 2.2 Руководство программиста: М.: ООО "Лаборатория платежных решений", 2025 — 33с.

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. 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 "Лаборатория платежных решений".

Dual Connector Версия 2.2 © ООО "Лаборатория платежных решений" 2025

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 ООО "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».

1.1. 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, то отпадает необходимость иметь на терминале коммуникационные модули.

Позволяет устанавливать SSL соединение, используя предоставляемые терминалом сертификаты.

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

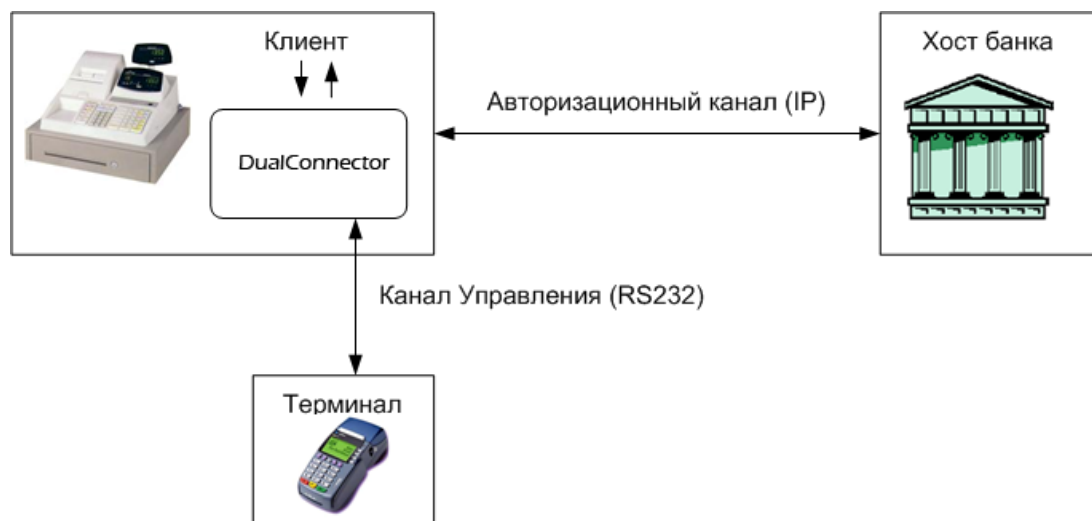


Рисунок 1. Организация взаимодействия с DualConnector

«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.

1.2. Системные требования

- Версия ОС: Windows 7/8/10;
- Установленный пакет «.NET Framework 4.0» и выше или «.NET 5+»;
- Права доступа «Администратор».

1.3. 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**». Рекомендуемый вариант — выполнять инициализацию один раз за время работы приложения (**FREERESOURCE_AUTO=OFF**). Для обратной совместимости поддерживается также схема инициализации перед каждой операцией (**FREERESOURCE_AUTO=ON**).

Примечание. Параметр **FREERESOURCE_AUTO** настраивается в файле **DualConnector.xml** (см. п. [2.1 Основные параметры](#)) или через утилиту **DC Control** на вкладке **XML Generator** (параметр «Очистка ресурсов»).

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;

5. Подготовить данные для проведения транзакции, заполнив соответствующие поля объекта «**Request**»;
6. Послать запрос на проведение транзакции, вызвав метод «**Exchange**» объекта «**DCLink**». Метод может вызываться многократно после единственной инициализации;
7. Обработать результат проведения транзакции, прочитав данные из объекта «**Response**»;
8. Освободить ресурсы, выделенные под объекты «**Request**» и «**Response**», используя для этого метод «**Release**» данных объектов;
9. 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.

1.3.1. Описание API

Объект «DCLink»

Метод «InitResources»:

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

Особенности метода «InitResources»:

- Метод вызывается один раз за время работы приложения.
- При повторном вызове возвращает ОК, если библиотека уже инициализирована. В случае ошибки выполняется повторная попытка инициализации.
- Для корректной работы рекомендуется отключить параметр **<FREERESOURCE_AUTO>** в конфигурации (значение **OFF**). Освобождение ресурсов выполняется вручную вызовом **FreeResources** в конце работы.
- Объекты **ISAPacket** необходимо создавать заново или очищать перед повторным использованием.

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 (используются значения из файла параметров «DualConnector.xml», при его наличии.)
- 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

Примечание. Значение параметров IPADDR, WAITACK, WAITPACKET копируются из файла параметров «DualConnector.xml», при его наличии, из первого настроенного устройства (DEVICE).

Метод «Cancel»:

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

Примечание. Использование метода «Cancel» возможно только при работе в асинхронном режиме.

Метод «SetField»¹:

- 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

¹ Поля для которых используется метод «SetField»: 59*, 91*, 94, 95, 96*, 97, 98, 99, 100, 101, 102, 103, 104, 105;

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»²:

- 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.

² Поля для которых используется метод «GetField»: 59*, 91*, 94, 95, 96*, 97, 98, 99, 100, 101, 102, 103, 104, 105;

* - поля требуется передавать в hex-формате

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](#)»).

Событие «OnExchange»

- Вызывается после завершения операции Exchange. Используется для обработки результата в асинхронном режиме. Если подписка на данное событие не осуществлена, обработка результата выполняется синхронно в месте вызова метода Exchange.

Примечание. Для событий **OnExchange** и **OnShowWindow** реализована проверка на множественную подписку обработчиков. Повторная подписка одним и тем же методом игнорируется.

Для обратной совместимости при прямой подписке через COM-интерфейс проверка не выполняется.

Объект «SAPacket»

Свойства объекта:

Свойства	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 ³	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

³ Если в ответе для **ОДОБРЕННОЙ** транзакции значение данного свойства равно 1 или 2, то это означает, что транзакция была подтверждена PIN-кодом.

Свойства	Поле протокола SA	Описание	Тип значения
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 ⁴	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	Статус получения Dual Connector результата операции от терминала (при обмене с кассовым ПО не используется) ⁵	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 ⁶
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

⁴ Описание свойства **Status** смотри ниже в данном пункте

⁵ При включенном параметре CONFIRM_OPERATION (п.3.1) поля используются для подтверждения операции на стороне DualConnector.

⁶ Для Elecsnet: в поле 70 DualConnector возвращает на кассу строка в hex-формате.

Свойства	Поле протокола SA	Описание	Тип значения
AdditionalData	86	Дополнительные данные транзакции	String
ModelNo ⁷	89	Наименование модели ВУ	String
ReceiptData	90	Данные для печати на чеке	String

Дополнительные свойства объекта «SAPacket» (передаются в составе 89 поля SA-протокола)

В библиотеке DualConnector реализованы дополнительные свойства, которые не относятся к стандартным полям протокола SA, но используются для передачи в кассовое ПО служебной информации:

Свойства	Поле протокола SA	Описание	Тип значения
VersionDC	89	Версия используемой библиотеки DualConnector	String
VersionTerminalDriver	89	Версия установленного драйвера терминала (PAX, VeriFone)	String
VersionJVMonKKM	89	Версия JVM на кассе. В текущей реализации Dual Connector значение всегда равно 0. Используется только в Dual Connector 2.0	String

Особенности:

- данные свойства являются только для чтения;
- по умолчанию их значения равны 0, пока не выполнена операция «Сверка итогов»;
- при уровне логирования ADVANCED и выше значения фиксируются в лог-файле DualConnector.

Пример записи в лог:

```
[89] = 'CON:SW:DC;1.3.19.0;SA;2;USB:VeriFone Unified USB Driver 5.0.2.1;0;'
```

где:

DC;1.3.19.0 — версия библиотеки DualConnector;

USB:VeriFone Unified USB Driver 5.0.2.1 — версия установленного драйвера терминала;

0 — значение для JVM (в текущей реализации Dual Connector всегда 0)

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

⁷ В данное поле может добавить информацию о своей версии касса, Dual Connector, и данные о времени ожидания ответа от терминала.

Значение	Описание
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

Обработка исключений

- Вывод информации об ошибке в случае неудачного соединения: терминал разрывает соединение с кассой, не дожидаясь разрыва соединения со стороны кассы.

2. Настройка параметров «DualConnector»

2.1. Основные параметры

Основной файл параметров должен находиться в директории с «DualConnector.dll» и называться «DualConnector.xml». Можно создать вручную.

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>
  <BINARY_FORMAT_FIELD70>ON</BINARY_FORMAT_FIELD70>
  <DEVICES>
    <DEVICE>
      <TYPE>TERMINAL</TYPE>
      <TERMINAL_ID>12345678</TERMINAL_ID >
      <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>
    <CONNECT_TIMEOUT>20</ CONNECT_TIMEOUT >
    <SSL>ON</SSL>
    <SSLTYPE>TLS</SSLTYPE>
    <CHECKNAME>OFF</CHECKNAME>
    <CERTNAME></CERTNAME>
    <USEMUTEX>OFF<\USEMUTEX>
  </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**», «**VERBOSE**», «**DEBUG**». Наличие необязательно, по умолчанию «**NONE**».

Примечание. В режиме ПКХ (использование коммуникаций кассы для взаимодействия POS-терминала с хостом банка) в лог-файл записывается информация о подключении, отправке и получении данных, в том числе IP-адресе и порт хоста.

- 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. Рекомендуемое значение «OFF» (работа в режиме единичной инициализации библиотеки и последовательного выполнения операций). При отсутствии секции параметра метод «FreeResources» не вызывается автоматически. Значение «ON» поддерживается для обратной совместимости и используется, если кассовое ПО работает по схеме инициализации перед каждой транзакцией.
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. **BINARY_FORMAT_FIELD70** — включает конвертацию поля 70 из текста в бинарный формат. Если значение параметра отличается от «ON» или отсутствует, то конвертация производиться не будет.
7. **DEVICES** — описание подключённых терминалов. Наличие обязательно.
 - 7.1. **DEVICE** — описание подключенного терминала или пинпада
 - 7.1.1. **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».
 - 7.1.2. **TERMINAL_ID** — хранит значение ID терминала. Для каждого ID должна быть создана секция **DEVICE**. Если значение **TERMINAL_ID** совпадает с **TID** принятым от кассы (27 поле, нормализованный вид, без спец. символов), то запрос будет отправлен по коммуникация указанным в данной секции **DEVICE** (см. п. 4). Если принятое значение **TID** не совпадает с **TERMINAL_ID**, то запрос будет отправлен по всем коммуникациям указанных в секциях **DEVICE**.
 - 7.1.3. **CONNECTION** — описание подключения к терминалу. Наличие обязательно
 - 7.1.3.1. **TYPE** — тип подключения. Допустимые значения «COM», «IP».
 - 7.1.3.2. **PORT** — номер COM порта. Наличие обязательно при соединении по COM
 - 7.1.3.3. **BAUDERATE** — скорость обмена. Наличие необязательно. По умолчанию 115200.
 - 7.1.3.4. **IPADDR** — IP адрес терминала. Наличие обязательно при соединении по IP
 - 7.1.4. **SA** — Описание параметров протокола. Наличие необязательно.
 - 7.1.4.1. **WAITACK** — время ожидания сигнала подтверждения получения пакета в секундах (или миллисекундах при значениях выше 300). Значение по умолчанию 5 секунд. Наличие необязательно
 - 7.1.4.2. **WAITPACKET** — время ожидания ответного пакета в секундах (или миллисекундах при значениях выше 300). Значение по умолчанию 45 секунд. Наличие необязательно
 - 7.1.5. **GUI** — сервер отображения запросов терминала. Наличие необязательно
 - 7.1.5.1. **IPADDR** — IP адрес сервера, на который адресуются запросы терминала. Описание протокола смотрите в соответствующем разделе данного документа.
8. **OUTPUT** — секция выходных параметров. Наличие необязательно.

- 8.1. **CONNECT_TIMEOUT** — Механизм прерывания установки соединения по истечению времени. Указывается время ожидания соединения с сервером в секундах. Значение по умолчанию — 20 секунд.
- 8.2. **SSL** — Признак обработки данной секции параметров
- 8.2.1. **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

- 8.3. **CHECKNAME** — Только при наличии параметра и значения «OFF» не осуществляется проверка имени серверного сертификата.
- 8.4. **CERTNAME** — Имя клиентского сертификата (подставляется вместо имени из самого сертификата) при хранении сертификатов на кассе, или имя серверного сертификата, при получении сертификатов с терминала.
- 8.5. **USEMUTEX** — включает протокол Mutex. Значения «OFF» — протокол Mutex не используется. Значения «ON» — протокол Mutex используется.

2.2. 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%».

2.3. Создание файла параметров «DualConnector.xml» в директории установки

Файла параметров «DualConnector.xml» можно создать несколькими способами.

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

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

Далее инсталляция происходит как в пункте 2.4.

2.3.2. Копирование заранее созданного файла параметров во время инсталляции.

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

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

Запустить инсталлятор с параметром:

```
COPYCONFIG=1
```

Далее инсталляция происходит как в пункте 2.4.

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

Ключи, с помощью которых можно создать файла параметров DualConnector.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» или кассового ПО. Применима 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 на повторное подключение/соединение 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 Installer.msi») — запуск инсталляционного файла.

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

Далее инсталляция происходит как в пункте 2.4.

2.4. 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 подробного режима ведения лога «VERBOSE». Без лицензии режим «VERBOSE» будет переключен в системный режим «SYSTEM».

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.*

2.5. Параметры настройки DC PosGUI

ПО «DC PosGUI». модуль ПО «DualConnector» — сервер, который слушает запросы терминала и выводит терминальные диалоговые окна на ККИ.

Для использования ПО «DC PosGUI» необходимо убедиться в том, что модуль установлен (при установке отметить чекбокс PosGUI).

ВНИМАНИЕ! ПО «DC PosGUI» — это приложение, которое работает в фоне (не служба).

Для корректной работы ПО «DC PosGUI» перед стартом работы с «DualConnector» необходим запуск «PosGui.exe».

Примечание. В случае если не запущено ПО «PosGUI.exe», то диалоговые окна не будут выводиться на кассу.

Настройка ПО «DC PosGUI» осуществляется при помощи файла «DC PosGUI.xml», располагающимся в папку с установленным «DualConnector». Стандартный путь расположения файла конфигурации «DC PosGUI.xml»: «C:\Program Files (x86)\INPAS\DualConnector\DC PosGUI.xml»

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

Примечание. Если производилась настройка ПО «DualConnector» в «DC Control», то обязательно перезапустить (или запустить) «PosGUI.exe».

Пример файла настройки конфигурации диалогового окна «DC PosGUI.xml»:

```
<root>
  <listen>127.0.0.1:6000</listen>
  <codepage>1251</codepage>
  <readtimeoutms>30</readtimeoutms>
  <closeable>true</closeable>
```

```
<fontsize>15</fontsize>
<width>500</width>
<height>280</height>
<maxwidth>500<maxwidth>
<maxheight>280<maxheight>
<autosize>true</autosize>
</root>
```

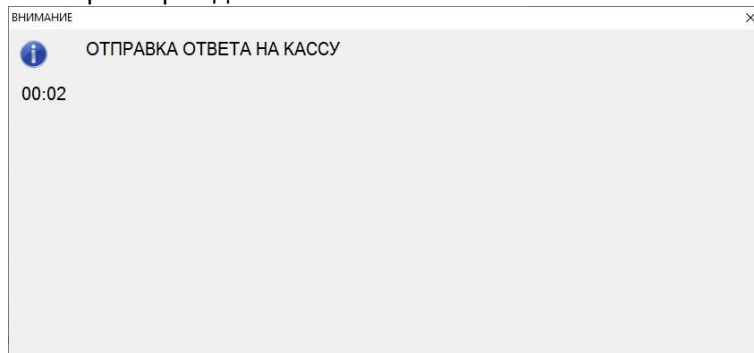
Параметры настройки DC PosGUI:

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 «DualConnector.xml» в поле **IPADDR**.
3. **codepage** — кодировка символов запроса. Числовое значение по умолчанию — 1251, что соответствует кодировке Windows-1251.
4. **readtimeoutms** — время считывания пакета запроса в миллисекундах.
5. **closeable** — Отображение кнопки закрытия окна «X». По умолчанию «не отображается», что соответствует значению «**lie**». Включению отображения кнопки закрытия соответствует значение «**true**».
6. **fontsize** — Размер шрифта текста окна. Значение по умолчанию: 15.
7. **width** — Размер диалогового окна в пикселях по горизонтали. Значение по умолчанию: 500
8. **height** — Размер диалогового окна в пикселях по вертикали. Значение по умолчанию: 500
9. **maxwidth** — Максимальный размер диалогового окна в пикселях по горизонтали. Значение по умолчанию соответствует размеру, в пикселях, подключенного экрана к кассе по горизонтали. Параметр может отсутствовать.
10. **maxheight** — Максимальный размер диалогового окна пикселях по вертикали. Значение по умолчанию соответствует размеру, в пикселях, подключенного экрана к кассе по вертикали. Параметр может отсутствовать.
11. **autosize** — Явное включение автоматического изменения окна под размеры контента. Принимает значения «**true**» или «**false**». Значение по умолчанию: «**false**». Параметр может отсутствовать. Если параметр «**autosize**» отсутствует или принимает значение «**false**», то размер диалогового окна будет подстраиваться подстраивается под размер контента используя параметры «**width**» и «**height**». Если параметр «**autosize**» принимает значение «**true**», то размер диалогового окна автоматически подстроится под размер контента под указанные параметры «**maxwidth**» и «**maxheight**».

Примечание.

Если параметры «**maxwidth**» и «**maxheight**» и «**autosize**» отсутствуют, то размер диалогового окна изменится под размеры контента используя параметры «**width**» и «**height**».

Пример изменённых параметров диалогового окна.



3. Рекомендации по использованию библиотеки «DualConnector»

В разделе приведены примеры для режимов работы:

- режим единой инициализации (рекомендуемый, FREERESOURCE_AUTO=OFF) — см. п. [3.1 C++](#) и п. [3.2 C#](#);
- режим с автоматической очисткой ресурсов (FREERESOURCE_AUTO=ON) — см. п. [3.3.1 C++](#) и п. [3.3.2 C#](#).

Примеры не являются готовым руководством к действию. Они приведены исключительно как ориентир для интеграции и могут требовать адаптации под конкретное кассовое ПО.

Внимание!

Модули «DualConnector» (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.

Примечание. РАЗРАБОТЧИКУ КАССОВОГО ПО ВАЖНО УЧИТЫВАТЬ ОСОБЕННОСТИ ОПЕРАЦИОННОЙ СИСТЕМЫ КАССЫ ПРИ ИСПОЛЬЗОВАНИИ НИЖЕ ПРИВЕДЕННЫХ ПРИМЕРОВ.

Внимание!

При работе в режиме единой инициализации библиотеки (FREERESOURCE_AUTO=OFF) для каждой операции используйте новый экземпляр **ISAPacket** (Request/Response) или очищайте поля перед повторным использованием.

Это не заменяет вызов метода Release. Release применяется в конце работы для окончательного освобождения памяти, а «новый/очищенный ISAPacket» нужен именно между последовательными операциями. Несоблюдение данного требования может привести к переносу данных между операциями и некорректной работе приложения.

3.1. C++

Ниже, представлена ссылка (запуск/закрытие «COM Library»), которая используется для помощи разработчику кассового ПО, а именно, верное использование опций, внутри кассового ПО.

//Initializes the COM Library — <https://docs.microsoft.com/en-us/windows/win32/com/the-com-library>

```
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;
}

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 );
        return;
    }

    // Метод Exchange может вызываться многократно после единственной инициализации.
    // Для каждой операции используйте новый ISAPacket (Request/Response) или очищайте поля
    // перед повторным использованием.

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

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

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

    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;
    // освобождение ресурсов один раз в конце работы
    dc->FreeResources();
    dc->Release();
    pRequest->Release();
    pResponse->Release();
    //Closes the COM library — https://docs.microsoft.com/en-us/windows/win32/com/the-com-library

```

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.

3.2. C#

```
DualConnector.DCLink dclink = new DualConnector.DCLink();
// Для каждой операции используйте новый SAPacket (Request/Response) или очищайте поля
// перед повторным использованием.
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) );
    return;
}
// Метод Exchange может вызываться многократно после единственной инициализации.
res = dclink.Exchange( ref query, ref response, 180000 );
if ( res != 0 )
{
    MessageBox.Show(string.Format("Exchange:{0}-{1}",res,dclink.ErrorDescription));
}
// освобождение ресурсов один раз в конце работы
dclink.FreeResources();
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.

3.3. Примеры для режима с автоматической очисткой ресурсов (FREERESOURCE_AUTO=ON)

Ниже приведены примеры для режима работы с автоматической очисткой ресурсов (FREERESOURCE_AUTO=ON). В этом режиме инициализация выполняется перед каждой операцией, а освобождение ресурсов вызывается автоматически.

3.3.1. C++

Данный пример нельзя расценивать, как руководство к действию. Пример является ориентиром при внесении изменений (встраивании его) в код кассового ПО.

Ниже, представлена ссылка (запуск/закрытие «COM Library»), которая используется для помощи разработчику кассового ПО, а именно, верное использование опций, внутри кассового ПО.

//Initializes the COM Library — <https://docs.microsoft.com/en-us/windows/win32/com/the-com-library>

```

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();
```

```
//Closes the COM library — https://docs.microsoft.com/en-us/windows/win32/com/the-com-library
```

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.

3.3.2. 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.

3.4. 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 «**DC PosGUI**» 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, пример файла представлен в разделе «Параметры настройки DC PosGUI».

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.
2. 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.

ПО «DualConnector» обеспечивает поддержку функционала «Синхронизация диалоговых окон»: синхронизацию отображения всех действий терминала в диалоговых окнах на кассе. Так же имеется возможность настроить параметры отображения диалогового окна (см «Параметры настройки DC PosGUI»).

Внимание! Во время взаимодействия DC с VPOS следует обратить внимание на, что когда пользователь производит оплату в момент выполнения деактивации кассовой ссылки (например, при нажатии в диалоговом окне «отмена»), то такая последовательность действий может привести к различным ошибкам. Рекомендуется пользоваться кнопками отвечающие за отмену только в том случае, когда клиент НЕ осуществил оплату.

В случае возникновения ошибок при деактивации платежной ссылки СБП: дополнительно запросить с кассы статус последней операции.

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:

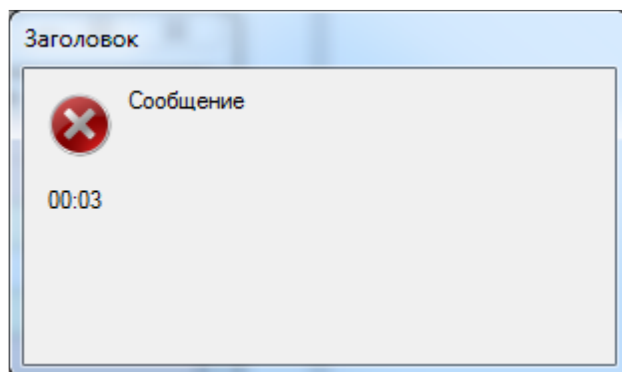
```
<response>
  <type>1</type>
  <data>0</data>
</response>
```

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>
```

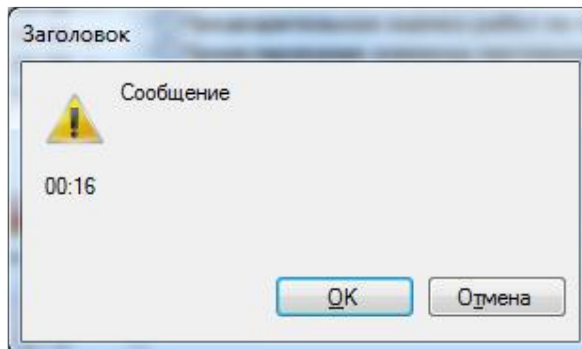
Где:

- 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>
```

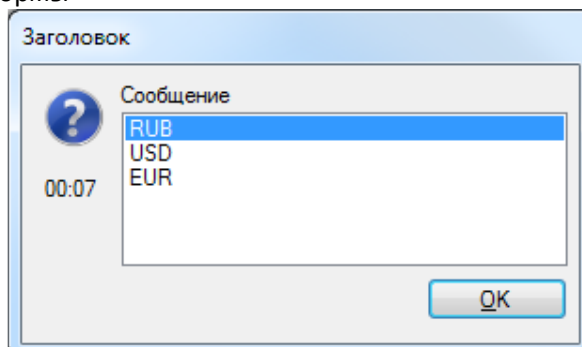
Формат ответа:

```
<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>
```

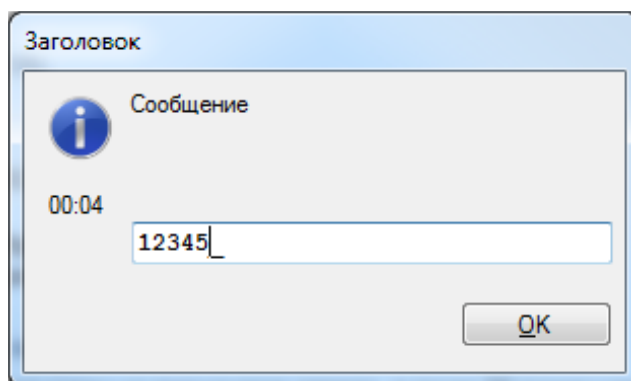
Формат ответа:

```
<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:





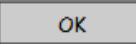
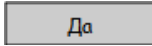


```
<response>
```

```
<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> <p>0x04 — Cancel  MB_CANCEL</p> <p>0x08 — No  MB_NO</p>	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	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

4. Приложение 1. Подключение нескольких терминалов к ККМ.

При работе с несколькими терминалами на одной ККМ нужно создать в файле параметров «DualConnector.xml» столько секций **DEVICE**, сколько подключено к ККМ терминалов.

В каждой секции **DEVICE**, настраивается способ подключения терминала к ККМ. Например, в секции с **TERMINAL_ID** «00000001» терминал настроен на соединение по COM-порту, а в секции с **TERMINAL_ID** «00000002» терминал настроен на соединение по IP-адресу.

Описания значений параметров предоставлены в пункте 2.1.

Пример файл параметров «DualConnector.xml» с двумя подключенными терминалами:

```
<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>
  <BINARY_FORMAT_FIELD70>ON</BINARY_FORMAT_FIELD70>
  <DEVICES>
    <DEVICE>
      <TYPE>TERMINAL</TYPE>
      <TERMINAL_ID>00000001</TERMINAL_ID >
      <CONNECTION>
        <TYPE>COM</TYPE>
        <PORT>COM10</PORT>
        <BAUDRATE>115200</BAUDRATE>
      </CONNECTION>
      <SA>
        <WAITACK>6</WAITACK>
        <WAITPACKET>45</WAITPACKET>
      </SA>
      <GUI>
        <IPADDR>127.0.0.1:6060</IPADDR>
      </GUI>
    </DEVICE>
    <DEVICE>
      <TYPE>TERMINAL</TYPE>
      <TERMINAL_ID>00000002</TERMINAL_ID >
      <CONNECTION>
        <TYPE>IP</TYPE>
        <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>
    <CONNECT_TIMEOUT>20</CONNECT_TIMEOUT>
    <SSL>ON</SSL>
    <SSLTYPE>TLS</SSLTYPE>
    <CHECKNAME>OFF</CHECKNAME>
    <CERTNAME></CERTNAME>
</OUTPUT>
</ROOT>
```

5. Приложение 2. Пример работы Dual Connector на C#.

Пример файла «Program.cs»:

```
using System;
using System.Diagnostics;
namespace ConsoleApplication1
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            var timeout = 180000;

            Pay(timeout);

            Pay(timeout);
        }

        public static void Pay(int timeout)
        {
            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 = "L100025P";

            int res = dclink.InitResources();
            Console.Out.WriteLine("init = " + res);

            if (res == 0)
                res = dclink.Exchange(ref query, ref response, timeout);

            Console.Out.WriteLine("exchange = " + res);

            dclink.Dispose();
        }
    }
}
```


Пример файла «AssemblyInfo.cs»:

```
using System.Reflection;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("ConsoleApplication1")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("ConsoleApplication1")]
[assembly: AssemblyCopyright("Copyright © 2024")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("1E359BF3-4584-4820-B8CF-68809FF77A86")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```