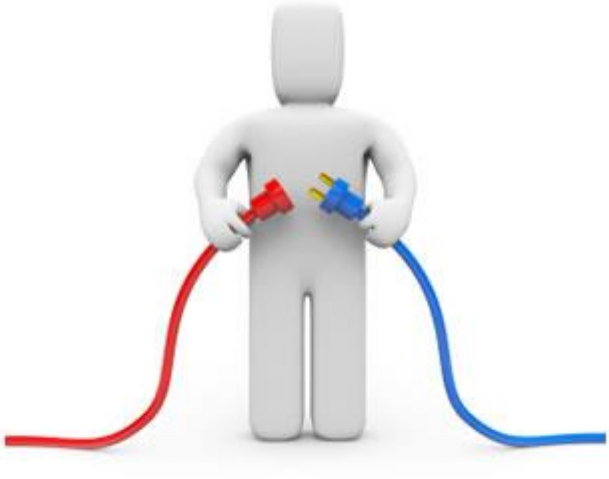
2023

ProLAN



DC Connector 1.4
32 & 64 bit

Руководство разработчика

Документ предназначен для разработчиков программного обеспечения и описывает методику использования компонента DcConnector.

Оглавление

Введение

Компонент ProLAN DC Connector (далее **DcConnector**) является составной частью распределенной архитектуры решений компании ProLAN использующих Концентраторы Данных (далее **DC**).
DC – компьютеры локальной сети под Windows, на которых установлено программное обеспечение EPM-Agent Plus.
Например, в решениях Кнопка Лояльности, Кнопка Активности Продавцов, Кнопка Анкетер и других решениях, к Концентраторам Данных подключаются проводные и беспроводные пульты. Концентраторы Данных также управляют Планшетными Пультами при проведении опросов.

Компонент DcConnector представляет собой 32-х или 64-х разрядный COM (ActiveX) компонент, который устанавливается на любой компьютер локальной сети, и позволяющий из любого Windows приложения выполнить нажатие одной из виртуальных кнопок на DC, либо инициировать управляемый опрос на одном из Планшетных Пультов, подключенных к DC. При этом приложение может дополнительно передать произвольный набор параметров (идентификатор клиента, номер чека и т.п.), который будет далее сохранен вместе с результатами опросов или нажатиями кнопок на пультах.
В версии 1.2 дополнительно появилась возможностью назначать и отменять владельцев (сотрудников компании) для планшетных и кнопочных пультов на DC.

Начиная с версии 1.3 в методах SetPanelOwnerNoWait (назначить владельца пульта) и SetOwnerForAllNoWait (назначить владельца для всех пультов) используются произвольные данные.

Установка компонента

Компонент может быть установлен на компьютер с ОС Windows XP и выше, включая Windows 10. В зависимости от разрядности кода приложений, использующих компонент необходимо устанавливать 32-х либо 64-х разрядную версию компонента. В 64-ч разрядной ОС Windows могут быть установлены обе версии компонента.

Дистрибутивы установки компонента DcConnector вы можете загрузить по ссылкам:
http://www.prolan.ru/files/freetools/DcConnectorSetup.exe
http://www.prolan.ru/files/freetools/DcConnector64Setup.exe

Скачайте дистрибутив и запустите на выполнение файл DcConnectorSetup.exe или DcConnector64Setup.exe (потребуются права администратора).

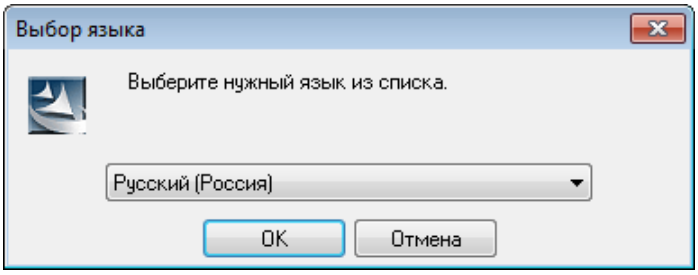


Рис.1. Выберите язык установки

На странице Сведений о пользователе (рис.2), для опции **Установить приложение для**, выберите пункт **всех пользователей данного компьютера**. На установку COM компонента это не влияет, но позволит использовать утилиты настойки компонента для всех пользователей компьютера.

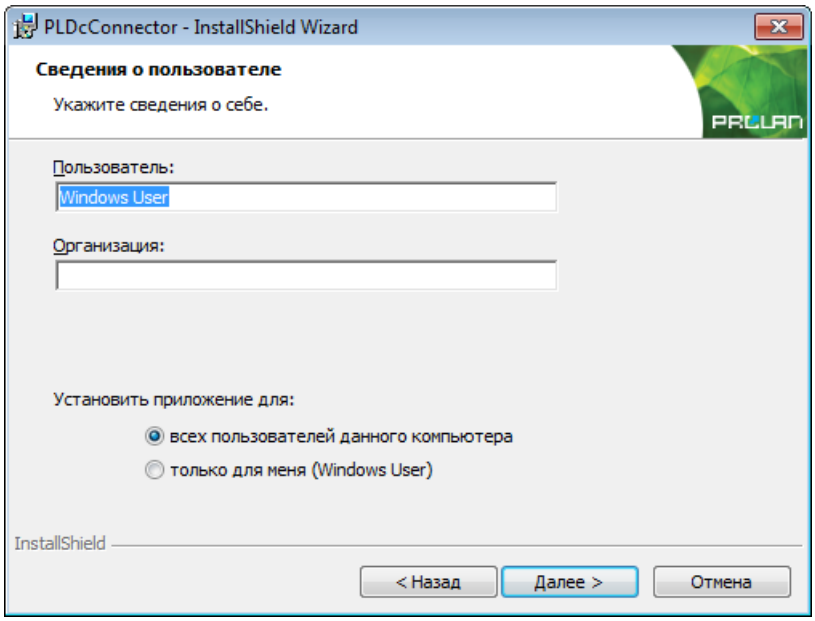


Рис.2. Сведения о пользователе

На следующей странице установки Вид установки (рис.3), вы можете выбрать **Полную** ли **Выборочную** установку дистрибутива компонента. Для разработчиков рекомендуется выбирать **полную** установку.

Дистрибутив включает 3 компонента установки:

1. **ActiveX компонент**. Устанавливает собственно сам COM компонент. Обязателен для установки.
2. **Настройка компонента**. Устанавливает утилиты настройки параметров компонента для текущего пользователя и локального компьютера. Рекомендуется устанавливать данный компонент.
3. **Разработка**. Руководство разработчика и включаемые файлы для компиляции в среде Visual Studio. Установка не требуется для компьютера конечного пользователя.

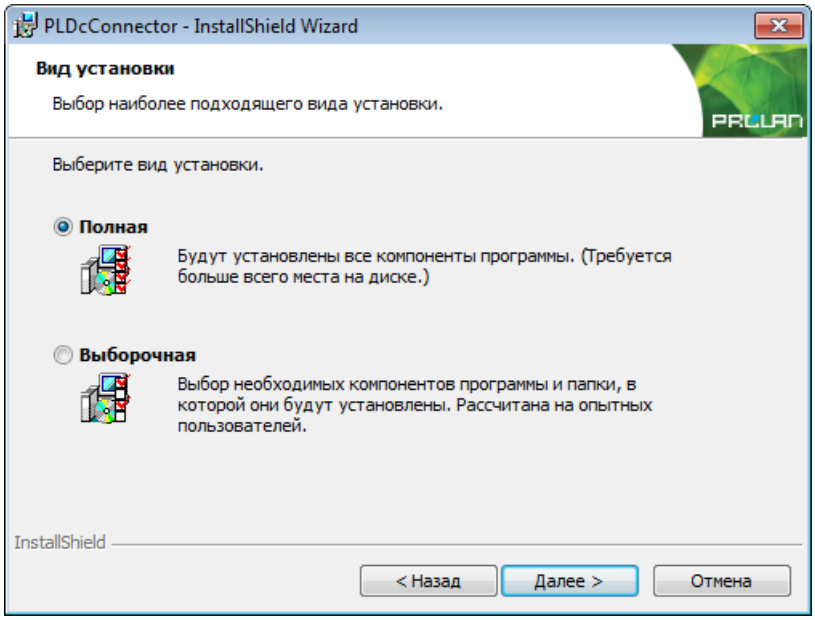


Рис.3. Вид установки

Если вы устанавливаете продукт на компьютере конечного пользователя, или хотите изменить папку установки по умолчанию, то выберите **Выборочная**, и на странице Выборочная установка (рис.4) произведите необходимые действия.

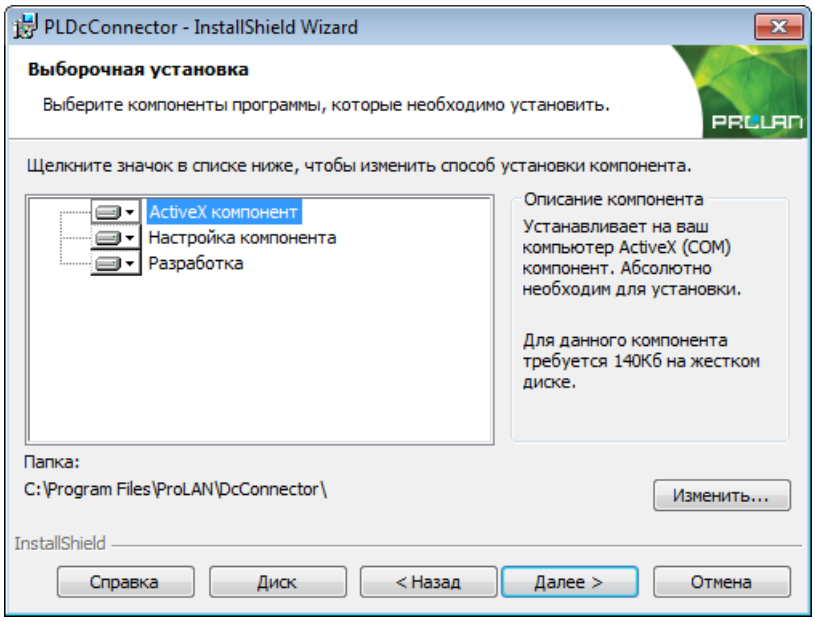


Рис.4. Выборочная установка

На рисунке 4 показана папка по умолчанию C:\Program Files\ProLAN\DcConnector, в которую будет производится установка файлов из состава компонентов **Настройка компонента** и **Разработка**. Для 64-х разрядных операционных систем, при установке 32-ч битной версии компонента папка будет отображаться как C:\Program Files (x86)\ProLAN\DcConnector
Если это необходимо, то нажав кнопку **Изменить…**, вы можете задать другую папку установки.

Внимание! Независимо от выбора папки установки, сам COM компонент DcConnector устанавливается и регистрируется в системе в папке C:\Program Files\Common Files\ProLAN\DcConnector либо C:\Program Files (x86)\Common Files\ProLAN\DcConnector в зависимости от разрядности компонента и операционной системы.

Нажмите на кнопки **Далее** и **Установить** для начала процесса установки. В процессе установки, если в ОС включен контроль учетных записей (UAC), то система запросит подтверждение на внесение изменений на данном компьютере. Разрешите внесение изменений. По окончании установки в меню Windows **Пуск → Все программы** будет добавлена папка **ProLAN → DcConnector** или **DcConnector64** со значками запуска программ настройки COM компонента и на просмотра данного руководство разработчика.

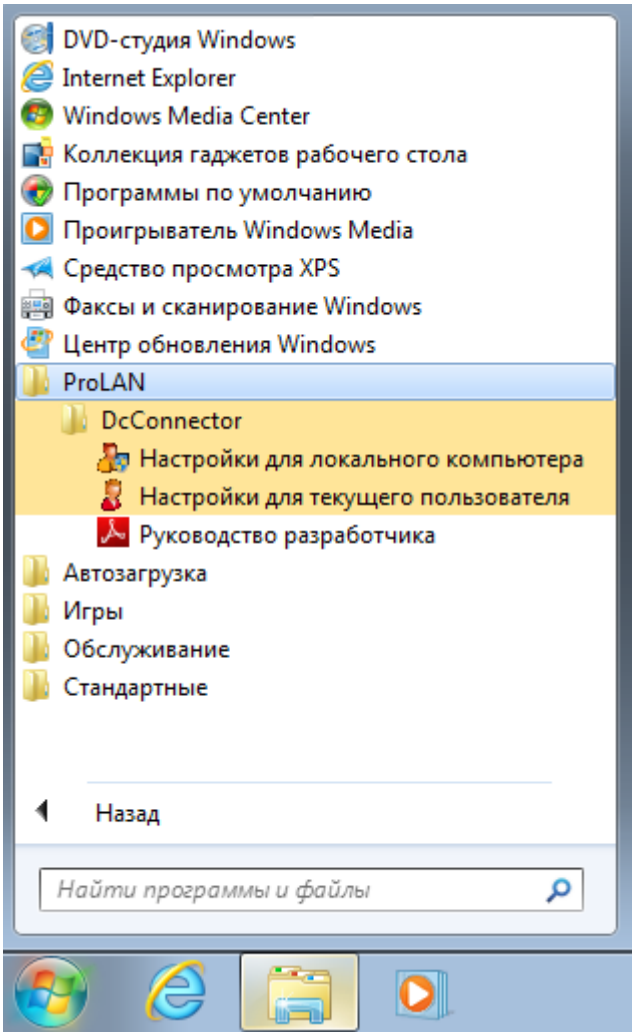


Рис.5. Пункты меню Window для запуска программ настройки компонента.

Настройки компонента

Компонент DcConnector и Концентратор Данных (DC), как правило (но не обязательно), находятся на разных компьютерах локальной сети. Между собой они общаются по протоколу UDP, обмениваясь пакетами (Datagram). Компонент DcConnector имеет интерфейс с именем UDPChannel, объекту которого до вызова основных методов необходимо сообщить:

- На каком компьютере сети находится DC, и по какому номеру порта UDP он принимает пакеты от DcConnector;
- Значение таймаута ожидания от DC подтверждения о приеме пакета, посланного DcConnector;
- Число повторов посылки пакета в случае ошибки посылки или неполучении подтверждения от DC приема пакета;
- Значение паузы между повторами посылки пакета;
- Имя виртуальной кнопки, которую «нажимает» DcConnector на DC, когда использует соответствующий метод;
- Псевдоним опроса и имя планшетного пульта DC, когда используется метод запуска управляемого опроса или метод назначения/отмены владельца пульта.

Вообще говоря, все перечисленные параметры могут быть сообщены объекту непосредственно в коде приложения, использующего компонент. Но передача значений настроек в коде приложения приводит к дополнительным затратам разработчика, хотя непосредственно к основному выполняемому действию (нажатие виртуальной кнопки, запуск опроса либо назначение/отмена владельца пульта) конкретные значения настроек отношения не имеют. Поэтому, в большинстве случаев удобно выполнить предварительные настройки COM компонента, значения которых будут сохранены в реестре компьютера. При создании объекта, значения из настроек автоматически будут переданы объекту. Таким образом, разработчик может не заботиться о начальной настройке свойств объекта.

Уровни хранения настроек

Настройки компонента могут храниться в реестре системы на двух уровнях:

- **Уровень текущего пользователя**. Значение любого параметра настроек либо значения всех параметров могут быть заданы и сохранены на уровне текущего пользователя системы. При создании объекта, компонент присваивает значения настроек, сохраненных на уровне данного пользователя свойствам объекта.
- **Уровень локального компьютера**. Значения любых параметров настроек могут быть заданы и сохранены на уровне локального компьютера. Если при создании объекта какие-либо значения параметров на уровне текущего пользователя не заданы, то используются значения соответствующих параметров уровня локального компьютера. Таким образом, настройки уровня локального компьютера могут быть использованы для любого пользователя системы, если настройки уровня текущего пользователя для него отсутствуют.

При эксплуатации программного обеспечения, использующего компонент DcConnector нужно выбрать вариант хранения настроек. Например, если пользователи компьютера меняются (имеют различные учетные записи), но для всех пользователей настройки компонента не отличаются, то

их целесообразно создавать на уровне локального компьютера. Если настройки (все или частично отличаются), то их необходимо создавать и сохранять на уровне каждого текущего пользователя отдельно. При этом можно использовать смешанный тип. Например, параметры концентратора (хост, порт, таймаут и повторы) сохранить на уровне локального компьютера, а имя нажимаемой виртуальной кнопки и/или псевдоним опроса, пульт и имя владельца пульта задать для каждого пользователя отдельно.

Настройки компонента для текущего пользователя

Запустите утилиту **Настройки для локального пользователя**. На рисунке 6 показано окно диалога программы сразу после запуска.

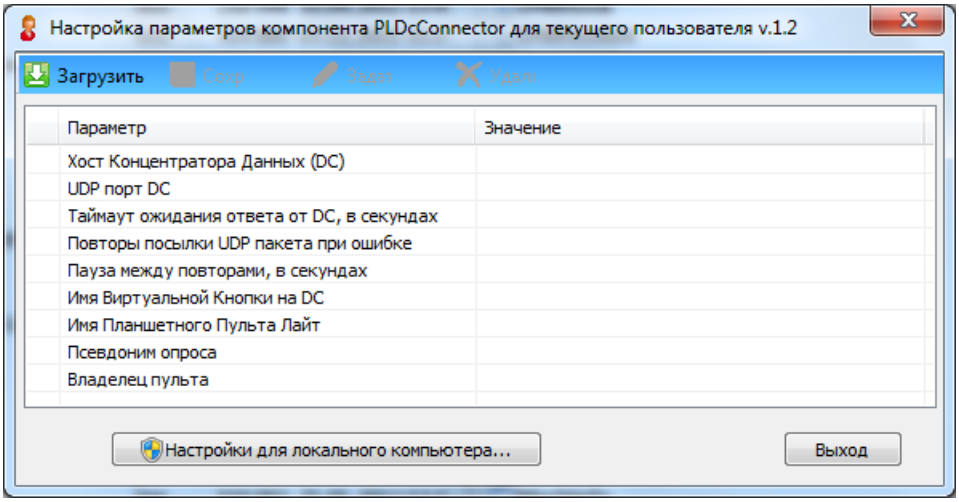


Рис.6. Окно диалога программы настроек для текущего пользователя.

Процесс задания настроек прост и интуитивно понятен, поэтому не нуждается в описании. При вводе значений числовых параметров (например, **Таймаут ожидания ответа от DC, в секундах**) программа также выдает информацию о диапазоне допустимых значений и контролирует правильность ввода.

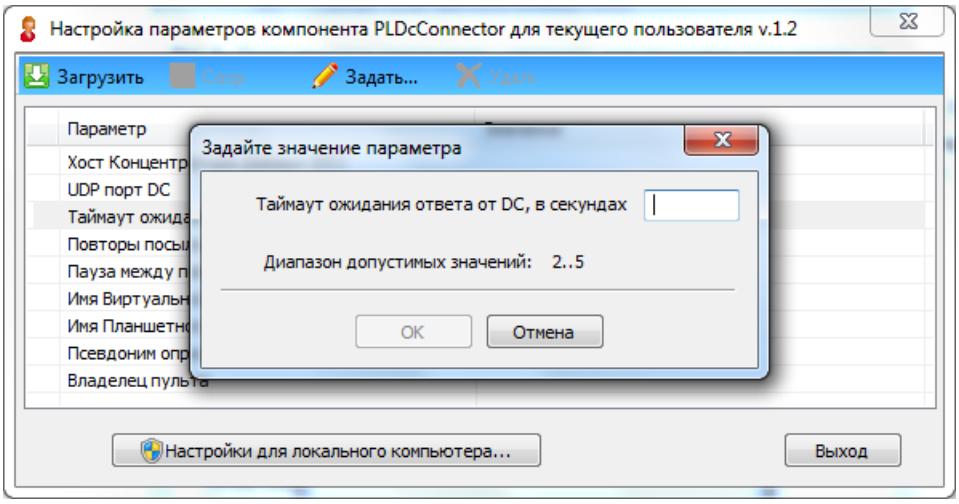


Рис.7. Задание значения числового параметра.

Список параметров:

- **Хост концентратора данных (DC)**. Может содержать IP-адрес, плоское (короткое) или доменное имя компьютера локальной сети;
- **UDP порт DC**. Номер порта Концентратора Данных, по которому он принимает UDP пакеты от компонента DcConnector. Если ни в настройках текущего пользователя, ни в настройках локального компьютера значение порта не будет задано, то будет использовано значение порта по умолчанию 7575;
- **Таймаут ожидания ответа от DC, в секундах.** Значение таймаута ожидания от DC подтверждения о приеме пакета. Диапазон значений от 2 до 5 секунд. Если значение не будет задано ни в одном из уровней настроек, будет использовано значение 2 секунды;
- **Повторы посылки UDP пакета при ошибке**. Задает число повторов посылки пакета в случае ошибки посылки или неполучении подтверждения от DC приема пакета. Диапазон значений от 0 до 10. Если значение не будет задано ни в одном из уровней настроек, будет использовано значение 0 (нет повторов);
- **Пауза между повторами, в секундах**. Диапазон значений от 1 до 10. Если значение не будет задано ни в одном из уровней настроек, будет использовано значение 1 секунда;
- **Имя виртуальной кнопки на DC**. Задает имя виртуальной кнопки, которую «нажимает» DcConnector на DC, когда использует метод PressVirtualButtonNoWait. Символьная строка, длиной до 31 символа включительно;
- **Имя Планшетного Пульта Лайт**. Задает имя планшетного пульта, управляемого с DC, на котором будет производиться управляемый опрос при вызове метода ExecPollNoWait или пульт для назнаения/отмены владельца при вызове методов SetPanelOwnerNoWait и ReleasePanelOwnerNoWait соответственно. Символьная строка, длиной до 15 символов включительно. Если значение не будет задано ни в одном из уровней настроек, то в качестве имени планшетного пульта при вызове метода ExecPollNoWait будет использоваться короткое имя локального компьютера;
- **Псевдоним опроса.** Задает псевдоним (некоторое символическое имя) управляемого опроса, который будет производиться на одном из планшетных пультов при вызове метода ExecPollNoWait. Символьная строка, длиной до 255 символов включительно.
- **Владелец пульта.** Задает в произвольной форме имя, фамилию и т.п. сотрудника, который будет назначен владельцем пульта при вызове метода SetPanelOwnerNoWait или владельцем всех пультов на DC при вызове метода SetOwnerForAllNoWait. Символьная строка, длиной до 63 символов включительно.

В панели инструментов окна приложения имеются кнопки:

- **Загрузить**. Загружает из реестра и отображает в окне значения сохраненных на данный момент параметров. Можно использовать комбинацию клавиш **Ctrl+L**;
- **Сохранить**. Сохраняет в реестре все изменения параметров сделанные в программе. Комбинация клавиши **Ctrl+S**;
- **Задать**. Позволяет задать или изменить значение выбранного в списке параметра. Клавиша **Пробел** или **Alt+Enter**;
- **Удалить.** Удаляет текущее значение выбранного в списке параметра. После сохранения данный параметр будет не задан (не определен). Клавиша **Delete.**

В нижней части окна программы находится кнопка **Настройка для локального компьютера…** , при нажатии на которую запускается утилита настройки параметров уровня локального компьютера. Обратите внимание, что в отличие от программы настроек для текущего пользователя, для запуска требуются права администратора, т.к. настройки сохраняются в ветке реестра LOCAL_MACHINE.

Настройки компонента для локального компьютера

Утилита может быть запущена через меню Пуск или из окна программы настроек текущего пользователя. Интерфейс пользователя полностью аналогичен описанному выше. На рисунке показано окно программы с введенными значениями параметров.

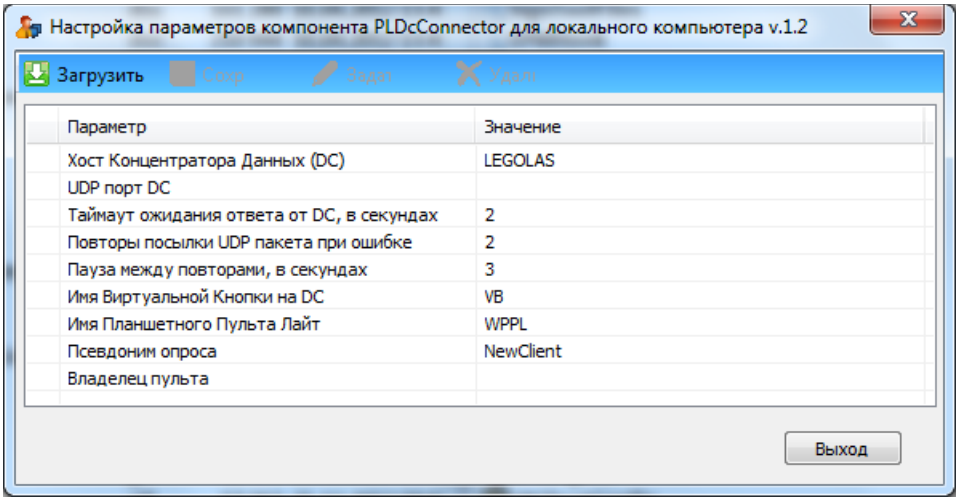


Рис.8. Окно диалога программы настроек для локального компьютера.

Создание объекта в коде приложения

В качестве языка примеров, в данном руководстве будет использоваться VBScript, синтаксис которого похож на множество языков используемых в средах разработки скриптового типа. Если язык разработки вашего продукта не имеет ничего общего с VBScript, но имеет возможность работы с COM компонентами, то используйте документацию и примеры работы с COM в вашей среде разработки. Дополнительно будут приводиться примеры на C++.

Компонент DcConnector имеет единственный класс **CUDPChannel** с интерфейсом **IUDPChannel**.

VBScript

В скриптовых языках объект создается с использованием строкового эквивалента идентификатора интерфейса (ProgID), имеющего значение `PLDcConnector.UDPChannel` либо `PLDcConnector64.UDPChannel` для 32-х и 64-х разрядной версии компонента:

```
...
On Error Resume Next
Err.Clear

'Переменная будет содержать объект
Dim l_objUDPChannel
'Создаем объект для 32-х разрядной версии компонента.
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")
'Для 64-х разрядной версии компонента создание объекта выглядит так
' Set l_objUDPChannel = CreateObject("PLDcConnector64.UDPChannel")

'Объект создан?
If Err.number <> 0 Then
    MsgBox "Ошибка создания объекта 'PLDcConnector.UDPChannel': " & _
            Err.Description, vbOKOnly + vbCritical, "Ошибка"
    Exit Sub
Else
    MsgBox "Объект создан.", vbOKOnly + vbInformation, _
            "Версия компонента " & l_objUDPChannel.SVersion
    'Закрываем объект
    Set l_objUDPChannel = nothing
End If
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"    // для 32-х разрадной версии
// либо
#include "PLDcConnector64_i.c"
#include "PLDcConnector64_i.h"      // для 64-х разрадной версии

...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
```

```
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Объект создан
}
else {
    // Ошибка создания объекта. Описание ошибки можно извлечь из hr
}
// Удаляем объект
l_spIUDPChannel = NULL;
...
```

Класс CUDPChannel является потокобезопасным. Это означает, что создав объект в одном из потоков программы, вы можете его использовать в других потоках, с единственным ограничением – код ошибки последней операции произведенной с объектом не поддерживается для каждого потока отдельно.

Свойства

Объект поддерживает несколько свойств (Properties), часть из которых дублирует настройки компонента, рассмотренные выше.

LastErrorCode

Код ошибки последней операции. Только чтение. Возвращает числовой код ошибки последней операции произведенной с объектом, к которым относятся задание любых свойств и вызов любых методов. При задании свойства и вызове метода, значение кода ошибки устанавливается в 0. Если при выполнении кода свойства или метода случается ошибка, то код ошибки принимает одно из значений, соответствующий контексту ошибки.

Коды ошибок:

	Число	Описание
_LE_NO_ERROR	0	Нет ошибки.
_LE_MEMORY_ALLOCATION_ERROR	1	Система не смогла выделить память.
_LE_CREATE_THREAD_ERROR	2	Ошибка создания потока в методе.
_LE_INCORRECT_UDP_PORT	3	Некорректный номер порта UDP.
_LE_INCORRECT_UDP_TIMEOUT	4	Некорректное значение таймаута ожидания подтверждения от DC.
_LE_INCORRECT_UDP_RETRY_COUNT	5	Некорректное значение числа повторов посылки пакета при ошибке.
_LE_INCORRECT_UDP_RETRY_PAUSE	6	Некорректное значение паузы между повторами.
_LE_NAME_IS_TOO_LONG	7	Слишком большая длина строки
_LE_EMPTY_PARAM_NAME	8	Отсутствует имя дополнительного параметра
_LE_VALUE_IS_TOO_LONG	9	Слишком большая длина строки для значения дополнительного параметра
_LE_HOST_NAME_NOT_SET	10	Не задано имя хоста DC
_LE_VIRTUAL_BUTTON_NAME_NOT_SET	11	Не задано имя виртуальной кнопки
_LE_POLL_ALIAS_NOT_SET	12	Не задан псевдоним опроса
_LE_MAX_UDP_THREAD_EXCEEDED	13	Превышено максимальное число потоков, отправляющих пакеты на DC
_LE_RSC_INSUFFICIENT	14	Система не может создать поток из-за нехватки ресурсов.

_LE_OWNER_NAME_IS_TOO_LONG	15	Слишком большая длина строки для владельца пульта
_LE_PPL_NAME_NOT_SET	16	Не задано имя пульта
_LE_OWNER_NAME_NOT_SET	17	Не задан владелец пульта

VBScript

```
Dim l_nLastErrorCode
' Получаем код ошибки последней операции
l_nLastErrorCode = l_objUDPChannel.LastErrorCode
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
LONG l_lLastErrorCode;
hr = l_spIUDPChannel->get_LastErrorCode(&l_lLastErrorCode);
// В l_lLastErrorCode теперь находится код ошибки последней операции
...
```

LastErrorText

Текст ошибки последней операции. Только чтение.

VBScript

```
Dim l_nLastErrorText
' Получаем описание ошибки последней операции
l_nLastErrorText = l_objUDPChannel.LastErrorText
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrErrorText;
hr = l_spIUDPChannel->get_LastErrorText(&l_bstrErrorText);
// В l_bstrErrorText теперь находится описание ошибки
...
```

DcHostName

Получает или задает имя хоста Концентратора Данных. Строка. При задании имени хоста, максимальная длина строки составляет 1023 символа включительно. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции

будет установлен в _LE_NAME_IS_TOO_LONG (7). В качестве имени хоста может быть использована строка, содержащая IP-адрес, имя или доменное имя компьютера.

VBScript

```
Dim l_sOldHostName
' Получаем текущее имя хоста DC
l_sOldHostName = l_objUDPChannel.DcHostName
' Устанавливаем другое имя хоста DC
l_objUDPChannel.DcHostName = "DCHOST.OFFICE.FABRICAM.COM"
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrOldHostName;
hr = l_spIUDPChannel->get_DcHostName(&l_bstrOldHostName);
// В l_bstrOldHostName теперь находится имя хоста
// Задаем другое имя хоста, например по его IP-адресу
hr = l_spIUDPChannel->put_DcHostName(CComBSTR("10.0.3.124"));
...
```

UDPPort

Получает или задает номер UDP порта, по которому DC принимает пакеты от DcConnect. Число. При задании значения, допустимы номера портов от 1 до 65535. В противном случае текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_INCORRECT_UDP_PORT (3). Номер порта по умолчанию 7575, присваивается объекту при его создании, если другой номер порта не был задан явно в настройках компонента.

VBScript

```
Dim l_nOldUDPPort
' Получаем текущий номер порта
l_nOldUDPPort = l_objUDPChannel.UDPPort
' Устанавливаем другой номер порта
l_objUDPChannel.UDPPort = 8008
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
LONG l_nOldPortNumber;
hr = l_spIUDPChannel->get_UDPPort(&l_nOldPortNumber);
// Задаем другой номер порта
```

```
hr = l_spIUDPChannel->put_UDPPort(8008);
...
```

UDPTimeout

Получает или задает значение таймаута ожидания от DC подтверждения о приеме пакета, в секундах. Число. При задании свойства, допустимы значения от 2 до 5 (секунд). В противном случае текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_INCORRECT_UDP_TIMEOUT (4).

VBScript

```
Dim l_nOldUDPTimeout
' Получаем текущее значение таймаута
l_nOldUDPTimeout = l_objUDPChannel.UDPTimeout
' Устанавливаем другое значение таймаута
l_objUDPChannel.UDPTimeout = 5
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
LONG l_nOldUDPTimeout;
hr = l_spIUDPChannel->get_UDPTimeout(&l_nOldUDPTimeout);
// Задаем другое значение таймаута
hr = l_spIUDPChannel->put_UDPTimeout(5);
...
```

UDPRetryCount

Получает или задает число повторов посылки пакета в случае ошибки посылки или неполучении подтверждения от DC приема пакета. При задании свойства, допустимы значения от 0 до 10. В противном случае текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_INCORRECT_UDP_RETRY_PAUSE (6).

VBScript

```
Dim l_nOldUDPRetryCount
' Получаем текущее значение
l_nOldUDPTimeout = l_objUDPChannel.UDPRetryCount
' Устанавливаем другое значение
l_objUDPChannel.UDPRetryCount = 3
```

C++

```
...
```

```
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
LONG l_nOldUDPRetryCount;
hr = l_spIUDPChannel->get_UDPRetryCount(&l_nOldUDPRetryCount);
// Задаем другое значение
hr = l_spIUDPChannel->put_UDPRetryCount(3);
...
```

UDPRetryPauseSeconds

Получает или задает паузу между повторами посылки пакета. При задании свойства, допустимы значения от 1 до 10. В противном случае текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_INCORRECT_UDP_RETRY_COUNT (5).

VBScript

```
Dim l_nOldUDPRetryPauseSeconds
' Получаем текущее значение
l_nOldUDPRetryPauseSeconds = l_objUDPChannel.UDPRetryPauseSeconds
' Устанавливаем другое значение
l_objUDPChannel. UDPRetryPauseSeconds = 10
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
LONG l_nOldUDPRetryPauseSeconds;
hr = l_spIUDPChannel->get_UDPRetryPauseSeconds(&l_nOldUDPRetryPauseSeconds);
// Задаем другое значение
hr = l_spIUDPChannel->put_UDPRetryPauseSeconds(10);
...
```

VirtualButtonName

Получает или задает имя виртуальной кнопки, которую «нажимает» DcConnector на DC, когда использует метод PressVirtualButtonNoWait. Символьная строка, длиной до 31 символа включительно. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_NAME_IS_TOO_LONG (7).

VBScript

```
Dim l_sOldVirtualButtonName
' Получаем текущее имя виртуальной кнопки
l_sOldVirtualButtonName = l_objUDPChannel.VirtualButtonName
' Устанавливаем другое имя виртуальной кнопки
l_objUDPChannel.VirtualButtonName = "MyVirtualButton"
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrOldVirtualButtonName;
hr = l_spIUDPChannel->get_VirtualButtonName(&l_bstrOldVirtualButtonName);
// Задаем другое имя имя виртуальной кнопки
hr = l_spIUDPChannel->put_VirtualButtonName(CComBSTR("MyVirtualButton"));
...
```

PplName

Получает или задает имя планшетного пульта, управляемого с DC, на котором будет выполняться управляемый опрос при вызове метода ExecPollNoWait или пульт для назнаения/отмены владельца при вызове методов SetPanelOwnerNoWait и ReleasePanelOwnerNoWait соответственно. Символьная строка, длиной до 15 символов включительно. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_NAME_IS_TOO_LONG (7). Если до вызова метода ExecPollNoWait значение не будет задано ни в одном из уровней настроек, ни явно, то в качестве имени планшетного пульта будет использоваться короткое имя локального компьютера;

VBScript

```
Dim l_sOldPplName
' Получаем текущее имя планшетного пульта для опроса
l_sOldPplName = l_objUDPChannel.PplName
' Устанавливаем другое имя планшетного пульта
l_objUDPChannel.PplName = "Cassa01"
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrOldPplName;
hr = l_spIUDPChannel->get_PplName(&l_bstrOldPplName);
// Задаем другое имя имя виртуальной кнопки
hr = l_spIUDPChannel->put_PplName(CComBSTR("Cassa01"));
...
```

PollAlias

Получает или задает псевдоним управляемого опроса, который будет выполняться на одном из планшетных пультов при вызове метода ExecPollNoWait. Символьная строка, длиной до 255 символов включительно. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в _LE_NAME_IS_TOO_LONG (7).

VBScript

```
Dim l_sOldPollAlias
' Получаем текущий псевдоним опроса
l_sOldPollAlias = l_objUDPChannel.PollAlias
' Устанавливаем другой псевдоним опроса
l_objUDPChannel.PollAlias = "PollForClient"
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrOldPollAlias;
hr = l_spIUDPChannel->get_PollAlias(&l_bstrOldPollAlias);
// Задаем другое другой псевдоним опроса
hr = l_spIUDPChannel->put_PollAlias(CComBSTR("PollForClient"));
...
```

OwnerName

Получает или задает имя владельца пульта, которое будет использовано при вызове методов SetPanelOwnerNoWait и SetOwnerForAllNoWait.

Символьная строка, длиной до 63 символов включительно. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в `_LE_OWNER_NAME_IS_TOO_LONG` (15).

VBScript

```
Dim l_sOldOwnerName
' Получаем имя владельца пульта
sOldOwnerName = l_objUDPChannel.OwnerName
' Устанавливаем другое имя владельца для последующего использования в методе
l_objUDPChannel.OwnerName = "Смирнов В.Д."
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrOldOwnerName;
hr = l_spIUDPChannel->get_OwnerName(&l_bstrOldOwnerName);
// Устанавливаем другое имя владельца для последующего использования в методе
hr = l_spIUDPChannel->put_OwnerName(CComBSTR("Смирнов В.Д."));
...
```

SendingInProgress

Только для чтения. Возвращает число потоков, выполняющих в данный момент передачу UDP пакетов на Концентратор Lанных, после вызова методов PressVirtualButtonNoWait, ExecPollNoWait, SetPanelOwnerNoWait, ReleasePanelOwnerNoWait, SetOwnerForAllNoWait и ReleaseOwnerForAllNoWait. Перечисленные методы являются асинхронными, т.е. запускают отдельный поток, выполнение которого может продолжаться и после возврата из кода метода в код вызывающего приложения. Поток может выполнять относительно длительные операции по разрешению IP-адресов, ожиданию подтверждения о приеме пакет от DC, повторных посылок в случае ошибки и т.п. Вообще говоря приложение не обязано отслеживать завершение потока посылки пакета на DC перед дальнейшим использованием объекта. Но перед удалением объекта (в однопоточной программе) или завершением программы (в многопоточной программе), будет правильно убедится, что все потоки передачи завершились.
Для этого, проверяйте в цикле значение свойства SendingInProgress, до тех пор, пока оно не будет равным 0.
Значение свойства равно 0, если все потоки передачи завершили работу. В противном случае значение будет больше 0, и показывает число потоков, продолжающих передачу в данный момент.

Примечание! Значение свойства – число выполняющихся потоков передачи, не связано напрямую с конкретным объектом класса `UDPChannel`. Свойство возвращает число выполняющихся потоков передачи для **всего приложения** (процесса в целом), независимо от того, каким конкретным объектом были созданы потоки передачи. Таким образом в многототочных приложениях, где каждый поток работает со своим объектом класса `UDPChannel`, вы можете смело удалять оъект, созданный в конкретном потоке, не проверяя значение свойства SendingInProgress. После удаления объекта, потоки передачи продолжат свою работу. Но перед завершением программы вам необходимо убедится, что все потоки передачи завершились. Для этого проверьте значение свойства SendingInProgress. Так как все объекты класса `UDPChannel` уже удалены, то вы можете создать новый объект и проверять значение свойства SendingInProgress у него.

Существует также возможность завершить работу программы, с принудительным завершением всех потоков передачи, если таковые выполняются в данный менент. Для этого используется метод Cleanup().

Ниже приведены примеры использования свойства SendingInProgress для ожидания завершения потока передачи в однопоточной программе (либо в программе, где только один поток работает с объектами `UDPChannel`). Перед удалением объекта, программа дожидается завершения всех потоков передачи. Так как потоки передачи создаются только в текущем потоке программы, то ожидание заверщения всех потоков передачи являетя совершенно корректным.

VBScript

```
Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что все необходимые параметры заданы в настройках компонента
Dim l_fSuccess
' Нажимаем виртуальную кнопку
```

```
l_fSuccess = l_objUDPChannel.PressVirtualButtonNoWait(0)

If fSuccess Then
    Dim l_InProgress

    Do
        ' Получаем статус работы потока передачи
        l_InProgress = l_objUDPChannel.SendingInProgress
        If l_InProgress = 0 Then
            ' Поток завершился
            Exit Do
        End If
        ' Поток еще работает. Ждем…
    Loop
End If

'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что все необходимые параметры заданы в настройках компонента
    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->PressVirtualButtonNoWait(0, &l_Success);
    if(l_Success) {
        while(1) {
            LONG l_nThreadCount;
            // Получаем статус работы потока передачи
            l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
            if(!l_nThreadCount)
                // Поток завершился
                break;
        }
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

SVersion

Только для чтения. Возвращает строку, содержащую номер версии компонента. Для текущей версии компонента возвращается строка "1.4".

VBScript

```
Dim l_sVersion
' Получаем номер версии компонента
l_sVersion = l_objUDPChannel.SVersion
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
CComBSTR l_bstrVersion;
hr = l_spIUDPChannel->get_SVersion(&l_bstrVersion);
...
```

Методы

Методы (Methods) объекта можно разделить на следующие категории:

- Метод очистки ошибки последней операции **ClearLastError**;
- Методы, выполняющие активные действия, к которым относятся **PressVirtualButtonNoWait**, **ExecPollNoWait**, **SetPanelOwnerNoWait**, **ReleasePanelOwnerNoWait**, **SetOwnerForAllNoWait**, и **ReleaseOwnerForAllNoWait**;
- Методы, управляющие набором произвольных данных, к которым относятся **AddParameterPair** и **ClearParameterPairs**. Произвольные данные передаются на Концентратор Данных при вызове методов **PressVirtualButtonNoWait, ExecPollNoWait, SetPanelOwnerNoWait** и **SetOwnerForAllNoWait**.
- Глобальный метод завершения потоков передачи данных на DC после вызова методов активных действий – **Cleanup**.

ClearLastError

Сбрасывает код ошибки последней операции с объектом в _LE_NO_ERROR (0). Не имеет параметров и не возврашает никаких значений.

VBScript

```
l_objUDPChannel.ClearLastError
```

C++

```
...
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
...
hr = l_spIUDPChannel->ClearLastError();
...
```

PressVirtualButtonNoWait

«Нажимает» виртульную кнопку на Концентраторе Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных;
- Имя виртуальной кнопки.

Набор произвольных данный (пар: имя-значение), если он задан для объекта `UDPChannel`, также пердается концентратору данных.

Метод имеет единственный числовой параметр, который может может принимать значения 0 или любое другое значение отличное от 0. При этом значение параметра 0 означает, что виртуальная

кнопка просто нажимается. Значение отличное от нуля, означает, что виртуальная кнопка нажимается с удержанием. На Концентраторе Данных для этих двух типов нажатий могут быть заданы различные активные действия. Если вы как разработчик не представляете, как это можно использовать с пользой, то вызывайте метод со значением параметра равным 0.

Имя метода не случайно содержит постфикс NoWait. На самом деле метод не выполнят никаких сетевых взаимодействий с DC. Вместо этого он проверяет наличие необходимых параметров и запускает отдельный поток, который и осуществляет передачу UDP пакета на DC, ожидает подтверждения, выполняет необходимые повторы передачи. Таким образом ваше приложение не будет ждать завершения процесса сетевого обмена между локальным компьютером и DC, а может работать дальше.
После запуска потока передачи метод возвращает управление вызывающей программе. Т.к. поток передачи никак не связан с породившим его объектом, то объект после вызова метода может быть удален, что никоим образом не скажется на доставке пакета на DC.

В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.

Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что все необходимые параметры заданы в настройках компонента
Dim l_fSuccess
' Нажимаем виртуальную кнопку
l_fSuccess = l_objUDPChannel.PressVirtualButtonNoWait 0

If Not l_fSuccess Then
    Dim l_nLastErrorCode, l_sLastErrorText
    ' Получаем код и описание ошибки
    l_nLastErrorCode = l_objUDPChannel.LastErrorCode
    l_sLastErrorText = l_objUDPChannel.LastErrorText
    ' Выполняем какие-либо действия
    ...
End If

' Для однопоточного приложения, дожидаемся завершения работы потока передачи

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
```

```
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop

'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что все необходимые параметры заданы в настройках компонента
    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->PressVirtualButtonNoWait(0, &l_Success);
    if(!l_Success) {
        LONG l_lLasrErrorCode;
        l_spIUDPChannel->get_LastErrorCode(&l_lLasrErrorCode);
        CComBSTR l_bstrLastErrorText;
        l_spIUDPChannel->get_LastErrorText(&l_bstrLastErrorText);
        // Выполняем какие-либо действия
        ...
    }

    // Для однопоточного приложения, дожидаемся завершения
    // работы потока передачи

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

ExecPollNoWait

Запускает управляемый опрос на одном из планшетных пультов, управляемых с Концентратора Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных;
- Имя планшетного пульта (опционально);
- Псевдоним опроса.

Набор произвольных данный (пар: имя-значение), если он задан для объекта `UDPChannel`, также пердается концентратору данных.

Метод не имеет параметров .

Как и предыдущий метод, метод ExecPoll NoWait только проверяет наличие необходимых параметров и запускает поток, который производит сетевой обмен между локальным компьютером и DC.
В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.

Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что все необходимые параметры заданы в настройках компонента
Dim l_fSuccess
' Запускаем управляемый опрос
l_fSuccess = l_objUDPChannel.ExecPollNoWait

If Not l_fSuccess Then
    Dim l_nLastErrorCode, l_sLastErrorText
    ' Получаем код и описание ошибки
    l_nLastErrorCode = l_objUDPChannel.LastErrorCode
    l_sLastErrorText = l_objUDPChannel.LastErrorText
    ' Выполняем какие-либо действия
    ...
End If

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop
```

```
'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что все необходимые параметры заданы в настройках компонента
    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->ExecPollNoWait(&l_Success);
    if(!l_Success) {
        LONG l_lLasrErrorCode;
        l_spIUDPChannel->get_LastErrorCode(&l_lLasrErrorCode);
        CComBSTR l_bstrLastErrorText;
        l_spIUDPChannel->get_LastErrorText(&l_bstrLastErrorText);
        // Выполняем какие-либо действия
        ...
    }

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

SetPanelOwnerNoWait

Назначает владельца пульта (планшетного и/или кнопочного) на Концентраторе Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных;
- Имя пульта;
- Имя владельца.

Набор произвольных данный (пар: имя-значение), если он задан для объекта `UDPChannel`, также пердается концентратору данных.

Метод не имеет параметров . Как и предыдущие методы, метод SetPanelOwnerNoWait только проверяет наличие необходимых параметров и запускает поток, который производит сетевой обмен между локальным компьютером и DC.
В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.

Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что концентратор данных задан в настройках компонента
' Задаем имя пульта и имя владельца
l_objUDPChannel.PplName = "WPPL"
l_objUDPChannel.OwnerName = "Сергеев А.К."

Dim l_fSuccess
l_fSuccess = l_objUDPChannel.SetPanelOwnerNoWait

...

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop

'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...
```

```
// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что концентратор данных задан в настройках компонента
    // Задаем имя пульта и имя владельца
    hr = l_spIUDPChannel->put_PplName(CComBSTR("WPPL"));
    hr = l_spIUDPChannel->put_OwnerName(CComBSTR("Смирнов В.Д."));

    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->SetPanelOwnerNoWait(&l_Success);
    ...

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

ReleasePanelOwnerNoWait

Отменяет назначение владельца для пульта (планшетного и/или кнопочного) Концентратора Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных;
- Имя пульта.

Метод не имеет параметров . Метод проверяет наличие необходимых параметров и запускает поток, который производит сетевой обмен между локальным компьютером и DC.
В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.
Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")
```

```
' Считаем, что концентратор данных задан в настройках компонента
' Задаем имя пульта, владельца которого нужно отменить
l_objUDPChannel.PplName = "WPPL"

Dim l_fSuccess
l_fSuccess = l_objUDPChannel.ReleasePanelOwnerNoWait

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop

...
'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что концентратор данных задан в настройках компонента
    // Задаем имя пульта, владельца которого нужно отменить
    hr = l_spIUDPChannel->put_PplName(CComBSTR("WPPL"));

    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->ReleasePanelOwnerNoWait(&l_Success);
    ...

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

SetOwnerForAllNoWait

Назначает владельца для всех пультов (планшетных и кнопочных) на Концентраторе Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных;
- Имя владельца.

Набор произвольных данный (пар: имя-значение), если он задан для объекта `UDPChannel`, также пердается концентратору данных.

Метод не имеет параметров . Метод проверяет наличие необходимых параметров и запускает поток, который производит сетевой обмен между локальным компьютером и DC.
В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.
Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что концентратор данных задан в настройках компонента
' Задаем имя владельца
l_objUDPChannel.OwnerName = "Сергеев А.К."

Dim l_fSuccess
l_fSuccess = l_objUDPChannel.SetOwnerForAllNoWait

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop

...
'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что концентратор данных задан в настройках компонента
    // Задаем имя владельца
    hr = l_spIUDPChannel->put_OwnerName(CComBSTR("Смирнов В.Д."));

    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->SetOwnerForAllNoWait(&l_Success);
    ...

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

ReleaseOwnerForAllNoWait

Отменяет назначение владельца для всех пультов Концентратора Данных. До вызова метода, в объекте должны быть заданы (явно или в настройках компонента) параметры, необходимые для передачи UDP пакета на DC, включая:

- Имя хоста Концентратора данных.

Метод не имеет параметров . Метод проверяет наличие необходимых параметров и запускает поток, который производит сетевой обмен между локальным компьютером и DC.
В случае успеха метод возвращает TRUE, в случае ошибки FALSE. Если метод вернул FALSE, то дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.
Для однопоточного приложения (или приложения, в котором с объектом работает только один поток), после вызова метода, и перед удалением объекта, убедитесь, что поток передачи данных, созданный при вызове метода, завершил свою работу. Для этого, в цикле проверяйте значение свойства SendingInProgress, до тех пор, пока оно не станет равным 0.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что концентратор данных задан в настройках компонента

Dim l_fSuccess
l_fSuccess = l_objUDPChannel.ReleaseOwnerForAllNoWait

...

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop


'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...
#include <comdef.h>
#include <atlbase.h>

#include "PLDcConnector_i.c"
#include "PLDcConnector.h"
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что концентратор данных задан в настройках компонента

    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->ReleaseOwnerForAllNoWait(&l_Success);
    ...

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
```

```
    l_spIUDPChannel = NULL;
}
...
```

Произвольные данные

Очень важным свойством методов выполняющих активные действия, является возможность передачи на DC дополнительных данных, которые ваше приложение считает необходимым передать вместе с командой нажатия виртуальной кнопки, запуском опроса или назначением владельцев пультов.

Набор произвольных данных формируется из парных параметров. Парный параметр имеет символьное имя параметра и символьное значение параметра. Например, вызывая метод запуска управляемого опроса, вы можете дополнительно передать на DC номер чека пробитого на кассе. Парный параметр в этом случае будет иметь имя `"Номер чека"` и значение `"12345"`. Значение может быть пустой строкой, но имя не должно иметь нулевую длину. Добавляя в объект парные параметры, вы тем самым формирует набор произвольных данных, которые используют при передаче методы **PressVirtualButtonNoWait**, **ExecPollNoWait, SetPanelOwnerNoWait** и **SetOwnerForAllNoWait**.

Имеется ограничение в 511 байт на общий размер произвольных данных, которые могут быть переданы на DC. Общий размер вычислется как сумма размеров парных параметров. Размер парного параметра, в свою очередь, вычислется как сумма длин (число символов) строк имени и значения, плюс два байта.

Имеются предопределенные (специальные) имена парных параметров:

- **Client_ID**. Означает идентификатор клинта;
- **Document_ID**. Идентификатор документа;
- **Price**. Некоторое стоимостное значение;
- **Service**. Выполняемая операция, вид услуги и т.п.

Вы можете использовать как предопределенные, так и любые другие имена параметров.

AddParameterPair

Добавляет в набор произвольных данных парный параметр. Параметрами метода являются две строки – имя параметра и значение параметра. Значение может быть пустой строкой. Максимальная длина каждой строки не более 255 символов включительно.

В случае успеха, метод возвращает индекс (начиная от 0) парного параметра добавленного в набор произвольных данных. В случае ошибки, возвращается значение -1. В этом случае дополнительная информация о причине ошибки может быть возвращена через свойства LastErrorCode и LastErrorText объекта.

VBScript

```
...
Dim l_objUDPChannel
```

```
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что все необходимые параметры заданы в настройках компонента
' Добавляем произвольные данные
Dim l_Index
l_Index = l_objUDPChannel.AddParameterPair "Client_ID", "4534/35"
l_Index = l_objUDPChannel.AddParameterPair "Client_Name", "Сидоров К.Н."
l_Index = l_objUDPChannel.AddParameterPair "Document_ID", "0102346"
l_Index = l_objUDPChannel.AddParameterPair "Price", "4500.00"
l_Index = l_objUDPChannel.AddParameterPair "Service", "Платеж наличными"
l_Index = l_objUDPChannel.AddParameterPair "Касса", "01"

Dim l_fSuccess
' Запускаем управляемый опрос
l_fSuccess = l_objUDPChannel.ExecPollNoWait

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
    End If
    ' Поток еще работает. Ждем…
Loop

'Закрываем объект
Set l_objUDPChannel = nothing
```

C++

```
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что все необходимые параметры заданы в настройках компонента
    // Добавляем произвольные данные
    LONG l_Index;
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Client_ID"),
                            CComBSTR("4534/35"), &l_Index);
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Client_Name"),
                             CComBSTR("Сидоров К.Н."), &l_Index);
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Document_ID"),
                             CComBSTR("0102346"), &l_Index);
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Price"),
                             CComBSTR("4500.00"), &l_Index);
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Service"),
                             CComBSTR("Платеж наличными"), &l_Index);
    hr = l_spIUDPChannel->AddParameterPair(CComBSTR("Касса"),
                             CComBSTR("01"), &l_Index);

    // Запускаем управляемый опрос
    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->ExecPollNoWait(&l_Success);

    while(1) {
```

```
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
...
```

ClearParameterPairs

Чистит набор произвольных данных объекта. После вызова метода набор не содержит ни одного парного параметра.

Метод не имеет параметров и ничего не возвращает.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")

' Считаем, что все необходимые параметры заданы в настройках компонента
' Добавляем произвольные данные
Dim l_Index
l_Index = l_objUDPChannel.AddParameterPair "Client_ID", "4534/35"
l_Index = l_objUDPChannel.AddParameterPair "Client_Name", "Сидоров К.Н."
l_Index = l_objUDPChannel.AddParameterPair "Document_ID", "0102346"
l_Index = l_objUDPChannel.AddParameterPair "Price", "4500.00"
l_Index = l_objUDPChannel.AddParameterPair "Service", "Платеж наличными"
l_Index = l_objUDPChannel.AddParameterPair "Касса", "01"

Dim l_fSuccess
' Запускаем управляемый опрос
l_fSuccess = l_objUDPChannel.ExecPollNoWait

' Чистим произвольные данные
l_objUDPChannel.ClearParameterPairs

' Добавляем другие произвольные данные
l_Index = l_objUDPChannel.AddParameterPair "Номер чека", "0102346"
l_Index = l_objUDPChannel.AddParameterPair "Касса", "01"

' Нажимаем виртуальную кнопку
l_fSuccess = l_objUDPChannel.PressVirtualButtonNoWait 0

Dim l_InProgress

Do
    ' Получаем статус работы потока
    l_InProgress = l_objUDPChannel.SendingInProgress
    If l_InProgress = 0 Then
        ' Поток завершился
        Exit Do
```

```
    End If
    ' Поток еще работает. Ждем…
Loop

'Удаляем объект
Set l_objUDPChannel = nothing
```

C++

```
...

// SMART указатель на объект с интерфейсом IUDPChannel
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
if(hr == S_OK) {
    // Считаем, что все необходимые параметры заданы в настройках компонента
    // Добавляем произвольные данные
    LONG l_Index;
    l_spIUDPChannel->AddParameterPair(CComBSTR("Client_ID"),
                            CComBSTR("4534/35"), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Client_Name"),
                             CComBSTR("Сидоров К.Н."), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Document_ID"),
                             CComBSTR("0102346"), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Price"),
                             CComBSTR("4500.00"), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Service"),
                             CComBSTR("Платеж наличными"), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Касса"),
                             CComBSTR("01"), &l_Index);

    // Запускаем управляемый опрос
    VARIANT_BOOL l_Success;
    hr = l_spIUDPChannel->ExecPollNoWait(&l_Success);

    // Чистим произвольные данные
    l_spIUDPChannel->ClearParameterPairs();

    //Добавляем другие произвольные данные
    l_spIUDPChannel->AddParameterPair(CComBSTR("Номер чека"),
                            CComBSTR("0102346"), &l_Index);
    l_spIUDPChannel->AddParameterPair(CComBSTR("Касса"),
                            CComBSTR("01"), &l_Index);

    // Нажимаем виртуальную кнопку
    hr = l_spIUDPChannel->PressVirtualButtonNoWait(0, &l_Success);

    while(1) {
        LONG l_nThreadCount;
        // Получаем статус работы потока
        l_spIUDPChannel->get_SendingInProgress(&l_nThreadCount);
        if(!l_nThreadCount)
            // Поток завершился
            break;
    }

    // Удаляем объект
    l_spIUDPChannel = NULL;
}
```

Cleanup

Завершает работу потоков сетевого обмена с DC, которые были созданы объектами при вызове методов активных действий. Если объект после вызова любого метода активных действий не дожидался завершения работы потока взаимодействия с DC, используя свойство **SendingInProgress**, то нельзя гарантировать, что на некоторый момент поток уже завершился. После вызова метода **Cleanup** будут гарантированно завершены **ВСЕ потоки** взаимодействия с DC, **ВСЕХ объектов** (существующих на данный момент и уже уничтоженных).

Правилом «хорошего тона» будет вызвать метод **Cleanup** непосредственно перед завершением работы приложения, если перед удалением объектов не проверяется значение свойства SendingInProgress. При этом для вызова метода может быть использован любой экземпляр объекта, существующий в коде приложения постоянно, либо созданный отдельно для вызова метода.

Метод не имеет параметров и ничего не возвращает.

VBScript

```
...

Dim l_objUDPChannel
'Создаем объект
Set l_objUDPChannel = CreateObject("PLDcConnector.UDPChannel")
…

'Перед завершение работы приложения. Используем существующий объект
l_objUDPChannel.Cleanup
```

C++

```
...

// Перед завершение работы приложения.
// В качестве примера создадим новый объект, специально для вызова метода
CComPtr<IUDPChannel> l_spIUDPChannel;
HRESULT hr = l_spIUDPChannel.CoCreateInstance(CLSID_UDPChannel);
l_spIUDPChannel->Cleanup();
l_spIUDPChannel = NULL;
...
```

Время жизни модуля и объектов

Модуль COM-компонента (PLDcConnector.dll либо PLDcConnector64.dll) загружается процессом в момент попытки создания в коде приложения первого объекта, и не выгружается до момента завершения приложения. Временем жизни объектов в приложении можно управлять. Локально созданный объект будет существовать до тех пор, пока не наступит одно из событий:

- Явное удаление (деструкция). Техника зависит от языка средства разработки;
- Уход из «области видимости» смарт указателей на объект. В скриптовых языках это переменные ссылающиеся на объект;

- Завершение приложения.

Глобально созданный приложением объект живет до завершения приложения (или явного удаления) и может использоваться повторно в любых фрагментах и модулях кода.
В описании методов активных действий (**PressVirtualButtonNoWait, ExecPollNoWait, SetPanelOwnerNoWait, ReleasePanelOwnerNoWait, SetOwnerForAllNoWait** и **ReleaseOwnerForAllNoWait**) говорилось, что они создают отдельные потоки передачи, которые взаимодействуют с DC. Потоки не связаны с объектами, породившими их, и выполняются независимо от того существует ли объект в данный момент или нет. Потоки завершаются в случае:

- Успеха передачи пакета UDP на DC и получения подтверждения;
- В случае безуспешного выполнения заданного числа повторов передачи;
- Вызове метода Cleanup из любого экземпляра объекта;
- При завершении работы приложения.

На последний случай необходимо обратить внимание.
Неработоспособной будет архитектура, когда приложение будет запускать отдельный процесс, создающий объект, выполняющий любой активный метод, и после этого сразу завершающий свою работу. В этом случае созданный методом поток **может не закончить** обмен пакетами с DC (и скорее всего так и произойдет).
В этом случае процессу необходимо использовать циклическую проверку свойства SendingInProgress, до тех пор пока SendingInProgress не вернет 0 (потоки передачи завершены).

В следующих версиях компонента возможно будут добавлены синхронные эквиваленты активных методов.

Тестирование кода, использующего компонент

На первом этапе разработки, вы можете просто создавать код, использующий объекты компонента и отправляющий UDP пакеты «в пустоту». Для этого можно задать любое имя хоста DC (существующее или нет), любое имя виртуальной кнопки DC, любой псевдоним опрос и т.д.
Так как методы активных действий реально не взаимодействуют с DC, то вам и не важно, что процесс передачи на DC будет безуспешен.

В принципе, на этом можно и остановиться. Если же есть желание проверить работу с использованием реального DC, то в минимальном варианте можно сделать следующее:

1. Скачайте по ссылке http://www.prolan.ru/files/freetools/ProLAN EPM-Agent Plus.exe дистрибутив установки программы EPM-Agent Plus и установите его на любой компьютер локальной сети или на виртуальную машину;
2. Запустите на выполнение программу через меню **Пуск → Все программы → ProLAN → EPM-Agent → ProLAN EPM-Agent Plus**;
3. При первом запуске, в диалоге **Функциональность программы** включите опцию **Краская Кнопка**.

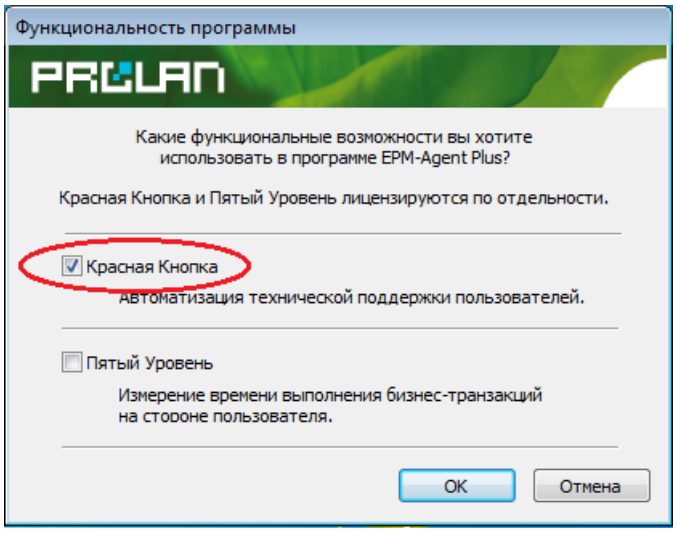


Рис.9. Выбор функционала **Красная Кнопка** при первом запуске.
Другие диалоги при первом запуске можно игнорировать.

4. Откройте окно программы, дважды щелкнув мышью на значке в области уведомлений панели инструментов. Убедитесь, что версия программы не ниже чем 2. 27 (версия видна в заголовке окна).

   Откройте диалог настроек программы через меню программы **Файл → Настройки…**

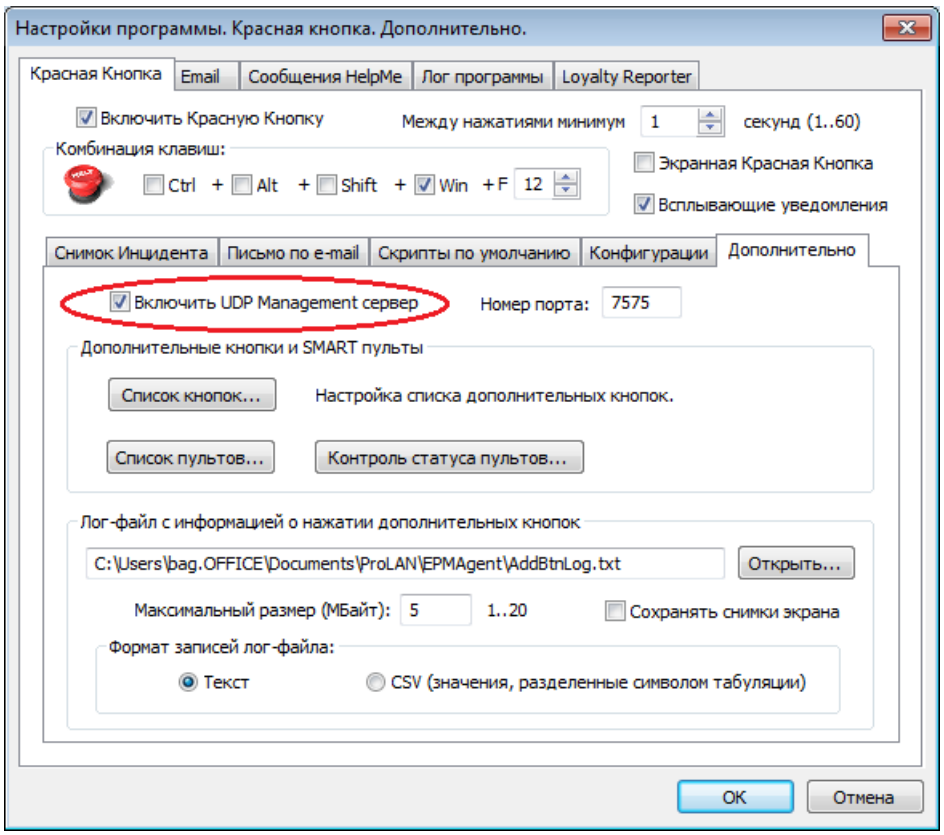


Рис.10. Включение UDP сервера.

На закладке **Красная Кнопка, Дополнительно** включите опцию **UDP Management сервер**. Это обеспечит прием пакетов от DcConnector.

5. Закройте диалог настроек кнопкой **OK**.
6. Отключите на компьютере с DC брандмауэр, т.к. он может блокировать прием UDP пакетов.
7. В настройках компонента DcConnector на вашем компьютере задайте имя или IP-адрес компютера куда вы установили программу EPM-Agent Plus.
8. Выполните ваш код, с использованием метода активных действий.
9. На компьютере с DC вновь откройте диалог настроек программы и перейдите на закладку Лог программы.

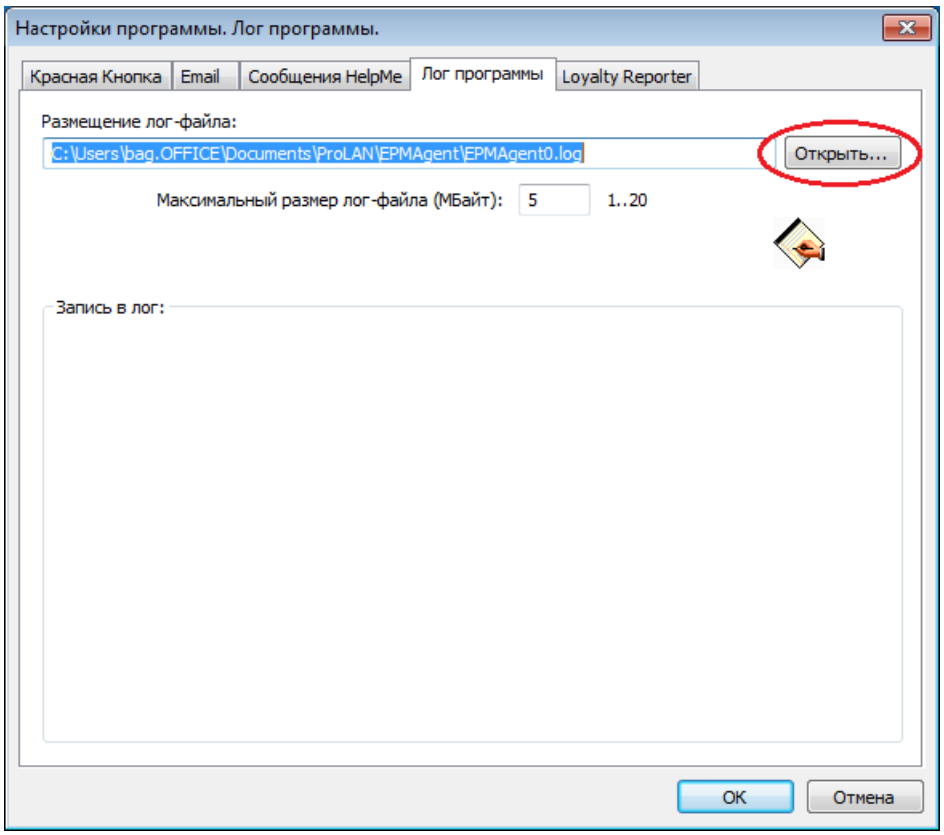


Рис.11. Просмотр Лога программы.

Нажмите на кнопку **Открыть…** В окне редактора будет показано содержимое лога. Если UDP пакет от DcConnector дошел до DC, то в конце текста лога вы найдете запись приблизительно такого содержания:

11/06/2015 17:59:29 INFO! С IP-адреса 10.0.3.122 принят UDP пакет _COM_2_DC_EXEC_POLL.
11/06/2015 17:59:29 Параметры пакета _COM_2_DC_EXEC_POLL: Client_ID=4534/35; Client_Name=Сидоров К.Н.; Document_ID=0102346; Price=4500.00; Service=Платеж наличными; Касса=01

Возможные проблемы и способы их разрешения

Ниже перечислены типичные проблемы которые могут возникнуть при использовании компонента.

Экземпляр объекта не создается, хотя установка компонента выполнена

Возможные причины:

- Разрядность установленного компонента не совпадает с разрядностью приложения. Например, в 64-х битной операционной системе была установлена 32-х разрядная версия компонента. При этом код приложения, создающего объект является 64-х битным.
    - Убедитесь, что в системе установлен компонент с разрадностью, совпадающей с разрядностью приложения;
    - Убедитесь, что приложение правильно использует имя интерфейса компонента. Для кода C++, включаемые файлы описания интерфейса должны соответствовать разрядности компонента и приложения. Для скриптовых языков, при создании объекта символьный эквивалент интерфейса (ProgID) должен быть указан правильно: PLDcConnector.UDPChannel для 32-х разрядного кода и PLDcConnector64.UDPChannel для 64-х разрядного кода.
- Пользователь, с правами которого производится создание экземпляра объекта не имеет на это прав. Используйте системную утилиту **dcomcnfg.exe** для проверки и, при необходимости, настройки прав доступа пользователя на COM-компонент.